TIBCO® Object Service Broker for z/OS

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

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 alphabetical 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.

Topics

Related Documentation

This section lists documentation resources you may find useful.

TIBCO Object Service Broker Documentation

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

Fundamental Information

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

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

Application Development and Management

The following manuals provide information about application development and management:

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

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

System Administration on the z/OS Platform

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

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

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

System Administration on Open Systems

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

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

External Database Gateways

The following manuals describe external database gateways:

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

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

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions

Convention	Use
`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*
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.

Table 1 General Typographical Conventions (Cont'd)

Convention	Use
	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`

`{ }`	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.

All TIBCO Object Service Broker for z/OS utilities whose names were starting with HRN have been renamed and their names now start with S6B. If your business processes require the use of utility names starting with HRN, those utility names are valid aliases.

Topics

S6BBRBAL (Segment Balance)

The S6BBRBAL (Segment Balance) utility reads a TIBCO Object Service Broker segment archive (backup) and creates a new archive that matches the new specifications provided for the segment. This utility works at the page level only. It accepts archives from any of the supported TIBCO Object Service Broker platforms, and can be used against Release 3.2 and above archives.

Purpose

S6BBRBAL 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 S6BTLBPS 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?
- Do you use point-in-time recovery?

Steps to Take Before Executing S6BBRBAL

Step 1. Take a Journal Spin

Do either of the following procedures:

1. Vary offline the segments on which you are to run S6BBRBAL.

   This segment is here referred to as the target segment.
2. Force a journal spin.

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

1. Suspend all users.
2. Force a journal spin.
3. Shut down the Data Object Broker.

Step 2: 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 S6BTLBPS (Backup Page Data Sets) to take a backup of each target segment.
2. Run S6BBRPTR (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 S6BBRBAL

1. Modify, as required, the JCL shown in member S6BBRBAL in the JCL data set.
2. Run S6BBRBAL (Segment Balance) for each target segment.

   Ensure you are using the appropriate backup data set for each segment.
3. Run S6BBRPTR (Batch Pointer Check) against each newly balanced segment to verify its contents.

   Ensure you are using the appropriate newly balanced data set for each segment.
4. 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 S6BBRBAL

When the new archive is created:

1. If the number of data sets in a target segment changed, modify the ACBS statement for that segment in your JCL member DBJCL to equal the new number of data sets in the segment.
2. Delete the existing data sets in the target segments.
3. Define the new data sets in the target segments.
4. Run S6BTLFPS (Format Page Data Sets) to format each new segment definition.
5. Run S6BTLRPS (Restore Pagestore) to restore the newly balanced archive to the segment, using the output from S6BBRBAL as input to S6BTLRPS.
6. If you are using continuous backup, restart it from this point for each target segment using the output of S6BBRBAL to create a base backup for each segment. Alternatively, you can run S6BTLBPS (Backup Page Data Sets) to create the base backup.

OR
If you are not using continuous backup, run S6BTLBPS (Backup Page Data Sets) to take a full backup of each target segment.

7. 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 step 7 can result in a corrupted target segments and loss of data. You can use a sort job using the following sort control statement to accomplish step 7:

```
SORT FIELDS=(5,6,BI,A,21,6,BI,D,30,3,BI,D,27,3,BI,D),EQUALS
 MODS
E15=(S6BSPX15,100000,EXITLIB,N),E35=(S6BSPX35,110000,EXITLIB,N)
*
------------------------------------------------------------------
-
* MODIFY THE FOLLOWING STATEMENT TO SPECIFY EACH SEGMENT
*      THAT IS BEING REBALANCED WITH S6BBRBAL. IN THE EXAMPLE,
*      SEGMENTS '00' AND '99' ARE BEING EXCLUDED FROM THE NEW
BACKUP.
*
----------------------V-----------------V------------------------
-
OMIT COND=((5,2,FI,EQ,+00),OR,(5,2,FI,EQ,+99))
RECORD TYPE=V
```

or use the S6BSPSEP (Separate Spun Journal by Segment) utility.

8. Restart or recycle the Data Object Broker.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for detailed information about the backup and restore procedures.

Invoking S6BBRBAL

The S6BBRBAL member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.

This sample is provided as a reference only; modify the JCL for your needs.

Note the following about the sample JCL:

- In the PARM= statement, the following can be specified:

F=	The number of data sets in the newly defined segment.
S=	The number of the offline segment. If this option is omitted, the segment number defaults to zero (that is, the base segment is expected).

P=	The number of defined pages in the input segment archive. 400000 is the default.
V=	Y produces a verbose audit report; N produces a terse audit report. A verbose audit report gives complete details of all modifications made to the segment, for example, the old and new numbers of all pages.

- `OLDARC` represents the name of the segment archive to use (input)
- `NEWARC` represents the name of the new segment archive (output)

Return Codes

Return Code	Meaning
0	No errors.
4	Warnings issued.

A non-recoverable error results in an abend.

Example Report

```
S6BBRBAL    Balance Segment number:   001      2009 Apr 22    12:50     v1.2
Balance Archive S6B.PRD.BACKUP.S001.G0011V00
        into Redefined Archive S6B.AP.OSBNEW.ARCHIVE
Number of files in redefined segment: 001


PHASE 1 - Analyze Old Archive:


Segment: 0001; Old File: 0; Extent: 0; Page# 00000000
# pages available: 31744; Bit map size: 0F80
Segment: 0001; Old File: 0; Extent: 1; Page# 00007C00
# pages available: 31744; Bit map size: 0F80
Segment: 0001; Old File: 0; Extent: 2; Page# 0000F800
# pages available: 26512; Bit map size: 0CF2

File 0 High Used Page: 0000B3B6; # Used Pages: 44593; # Pages Ignored: 3
OLD ARCHIVE - Read: 44596; Processed: 44596; Used: 44593; Ignored: 3


PHASE 2 - Re-assign Page Numbers:


Pages to be assigned to 1st file only: 44593
New File 0 Summary;
High Used Page: 0000AE32; # Assigned Pages: 44593; # Control Pages  2


PHASE 3 - Create New Archive:

New Control Page; 00000000
Prev: FFFFFFFF; Next: FFFFFFFF; Type: x'F0'
Number of Pages: 31744; Bit Map Size: 0F80; File High Page: 0000AE32

New Control Page; 00007C00
Prev: FFFFFFFF; Next: FFFFFFFF; Type: x'F0'
Number of Pages: 12851; Bit Map Size: 0647; File High Page: 0000AE32

Bytes: Input 144020059; Output - old 144007867; New 5767
OLD Pages - Read: 44596; Processed: 44596; Used: 44593; Ignored: 3
```

S6BBRCFC (Coupling Facility Size)

The S6BBRCFC program estimates the size of the Coupling Facility (CF) structure required to support the XCF recovery option of TIBCO Object Service Broker.

Invocation The JCL is as follows:

```
//STEP0 EXEC PGM=S6BBRCFC,
//PARM='CFNAME=RACE,COMM=4335,CMXU=630,LOCK=2416,LMXU=10'
//STEPLIB  DD  DISP=SHR,DSN=$HLQNONV$.$INSTVER$.AUTH
//SYSPRINT DD  SYSOUT=*
```

where

CFNAME	The name of the CF where the structure will be stored, and where the calculation is performed.
COMM	The number of COMMAREA buffers defined.
CMXU	The maximum number of COMMAREA buffers used.
LOCK	The number of LOCK buffers defined.
LMXU	The maximum number of LOCK buffers used.

Some input parameters are taken from the output of the S6BTLADM (Administration Menu) utility, option D shown below (figures should be recorded after peak load, such as at the end of a busy day):

```
S6BADMD1  S6RMDOBA            BUFFER POOL STATISTICS          2009APR16 08:16:21

POOL                            IN-USE         QUEUED
NAME      # GETS     # FREES    CURR    MAX    CURR    MAX    WAITS   ERRORS   BUFFERS
BUFQ                                                                        1200
CLOG                                                                         220
COMM         315          22     293    630                                4335
LOCK        3662        3657       5     10                                2416
MSGI           1                  1      1                                 150
NAMW                                                                          75
PAGE      118607      118607              24                               16384
SESS       35213       35213              10                                 304
VRPL        6312        6312               5                               32767
WORK         668         668               2                                1324
XTAB          30          30               5                                 256
```

Output The output is written to SYSPRINT as follows:

```
 MAXIMUM DATA SIZE CALCULATION
 =============================
COMM BUFFERS=            4335
LOCK BUFFERS=            2416

  CF MAXSIZE=           20736 KB

 INITIAL DATA SIZE CALCULATION
 =============================
CMXU BUFFERS=             630
LMXU BUFFERS=              10

 CF INITSIZE=            7424 KB
```

INITSIZE and MAXSIZE should be used as input to the structure definition.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for more information about the options on the Administration menu.

S6BBRCLR (Batch Table Clear)

S6BBRCLR deletes all data from a TDS table. This offline utility performs the same function as the $CLRTAB tool.

S6BBRCLR deletes all data in the table specified in the PSTABLE parameter. This batch utility operates in the same way as the online table clear; however, with this utility you can delete and load a table within a single job stream, provided that the segment containing the specified table is offline.

To ensure the integrity of your continuous backup after you run the S6BBRCLR utility, you must back up the segment containing the new data. For information about backing up a segment, refer to *TIBCO Object Service Broker for z/OS Managing Backup and Recovery*.

Invocation

The S6BBRCLR member of the JCL data set as distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Constraints

- The segment containing the specified table must be offline.
- S6BBRCLR cannot be used to delete data from a selected table instance of a parameterized table. To delete data from a selected table instance, you should use the $CLRTAB tool.

Online transactions are often subject to your site's chosen transaction limits. If you exceed your online transaction limit, you must use S6BBRCLR to clear the table while it is offline.

Sample Report

The following illustrates the S6BBRCLR report:

```
S6BBRCLR          BATCH CLEAR       MONTHLY_SALES      DATE 2009 APR 23 TIME 11:49

                                TABLE CLEAR SUMMARY

            TOTAL PAGES FREED                    4

            TOTAL PAGES UPDATED                  1

            START TIME  11:49:05     END TIME  11:49:05
```

See Also

TIBCO Object Service Broker Messages With Identifiers for messages associated with this utility and *TIBCO Object Service Broker Shareable Tools* for details on $CLRTAB.

S6BBREXT (Extract Selected Pages)

S6BBREXT is used to extract selected pages from a pagestore to assist with operational maintenance . It should only be used under instruction from TIBCO Support.

Invocation

1. The segment must be offline.
2. The output data set pointed to by the PAGES DD statement must be a sequential file with RECFM=VB, LRECL=4096
3. The input list of pages to be extracted will be read from the 80 bytes card images on the SYSIN DD in columns 1 to 8 - in Hex format.

Sample for Extracting Selected Pages

The following is a sample JCL to extract two pages from the segment 0:

```
//EXTRACT   EXEC PGM=S6BBREXT,PARM='SEG=00'
//STEPLIB   DD DSN=$HLQNONV$.$INSTVER$.LOAD.DISP=SHR
//DBDLIB   DD DSN=$HLQVSAM$.$SLQ$.DBDLIB.DISP=SHR
//PAGES   DD DSN=$HLQNONV$.$SLQ$.EXTRACT(PAGES).DISP=OLD
//SYSPRINT    DD SYSOUT=*
//SYSIN    DD *
02001609
020016D8
/*
```

Use **PARM='SEG-##'** to indicate the segment number.

S6BBRFRU (Reformat TIBCO Object Service Broker Files Transferred with FTP)

S6BBRFRU reformats TIBCO Object Service Broker files transferred between z/OS and Open Systems using FTP. S6BBRFRU can be used to reformat on z/OS files received from Windows or Solaris or to reformat files on z/OS in preparation for transfer to Windows or Solaris.

Prerequisites

- The TIBCO Object Service Broker files must be transferred using FTP. S6BBRFRU does not work with other file transfer tools.
- The files must be transferred in binary mode.
- Files created on z/OS must use the FTP LOCSITE sub command with the parameters RDW or NORDW as required. Use of these parameters is explained in the following sections.

File or Data Set Types That Can Be Reformatted

You can use S6BBRFRU to reformat the following types of files or data sets that are to be transferred with FTP:

- TIBCO Object Service Broker segment archives
- Import/export table files
- Missing BDW Variable Format files
- Workbench unload files
- TIBCO Object Service Broker offline batch utility unload files

The procedures for transferring and reformatting each of these file types are explained in the following sections.

Transferring TIBCO Object Service Broker Segment Archives

Reformatting Requirements for Segment Archives

Archives Created on z/OS and Transferred to Windows or Solaris

Segment archives created by the backup utility S6BTLBPS on z/OS can be transferred without intermediate processing to Windows or Solaris for use by TIBCO Object Service Broker. Transfer the files using binary mode and the z/OS FTP LOCSITE sub command with NORDW parameter specified.

Archives Created on Windows or Solaris and Transferred to z/OS

Segment archives created on Windows or Solaris must be reformatted before they can be used on z/OS.

To transfer archive files created by the backup utility hrntlbps on Windows or Solaris, complete the following steps:

1. Transfer the archive to z/OS using FTP in binary mode.

   The z/OS target file must have a LRECL greater than 36 bytes and a format of either FB or VB. We recommend that FTP be permitted to allocate the file, except in the case of a very large file where you should preallocate a sufficiently large file.

2. Run the JCL.

   Refer to the following sample JCL.

You should always verify backups created on Windows or Solaris using the hrnbrptr (Batch Pointer Check) utility before using them as archives. Re-formatted archives on z/OS should also be checked with S6BBRPTR (Batch Pointer Check) before they are used.

JCL to Reformat Archives

The JCL must have a statement similar to the following:

```
PARM='REFORMAT=ARCHIVE,SEG=segnum,PAGES=numofpages,BYTES=numofbytes'
```

This statement contains the following parameters:

REFORMAT	The reformat type, which is ARCHIVE.
SEG	The segment number that is to be reformatted. If it is not supplied, S6BBRFRU assumes segment 0.
PAGES	(Optional) The number of pages in the archive. Provide S6BBRFRU with the number of pages in the archive if you want to check if the archive is complete. This parameter cannot be used if there is more than one segment in the input data set.

BYTES	(Optional) The number of bytes in the archive. Provide S6BBRFRU with the number of bytes in the archive if you want to check if the archive is complete. This parameter cannot be used if there is more than one segment in the input data set.

Sample JCL

The BRFRUARC member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility to reformat an archive. This sample is provided as a reference only; modify the JCL for your needs.

Transferring TIBCO Object Service Broker Export/Import Table Files

- To avoid unpredictable changes to file data, the Default Import/Export Type on Open Systems must be set to the LENGTH_PREFIXED_EBCDIC format if the file is to be transferred. This applies even if the file transfer is occurring within the same operating system.
- You must know how the file is going to be used on Windows or Solaris before you use S6BBRFRU. Variable length (V, VB) files or data sets used for import/export tables and workbench unloaded files or data sets are processed identically by S6BBRFRU, since their formatting requirements are identical. However, format requirements for z/OS VB data sets that are to be used by offline batch utilities on Windows or Solaris have different formatting requirements (see below).

Reformatting Requirements for Import/Export Table Files

Files Created on z/OS and Transferred to Windows or Solaris

Fixed Length (F, FB) files	No reformatting is required provided the file is transferred in binary mode and the correct record size is entered in the file's *filespec.dsn* entry prior to importing.

Variable Length (V, VB) files	These must be reformatted before they are transferred to Windows or Solaris, after which they can be used as import tables. After reformatting the files using S6BBRFRU (refer to JCL to Reformat an Export Table Created in z/OS on page 17), transfer the files using binary mode and the z/OS FTP LOCSITE sub command with NORDW parameter specified.

Files Created on z/OS and Transferred to z/OS

Fixed Length (F, FB) files	No reformatting is required provided the file is transferred in binary mode.
Variable Length (V, VB) files	No reformatting is required provided the file is transferred in block (not stream) mode.

Files Created on Windows or Solaris and Transferred to z/OS

Fixed Length (F, FB) files	No reformatting is required provided the file is transferred in binary mode and the correct file characteristics are specified for z/OS (FB, LRECL, and so on).
Variable Length (V, VB) files	These files must be reformatted using S6BBRFRU after they have been transferred to z/OS (refer to JCL to Reformat an Export Table Created in z/OS on page 17). The files must be created in LENGTH_PREFIXED_EBCDIC form and transferred in binary mode. The format of the z/OS target file can be FB or VB. We recommend that you allow FTP to allocate the file, except for a very large file when you should preallocate a sufficiently large file.

Files Created on Windows or Solaris and Transferred to Windows or Solaris

No reformatting is required provided the file is transferred in binary mode.

JCL to Reformat an Export Table Created in z/OS

The JCL must contain a statement similar to the following:

```
PARM='REFORMAT=platform'
```

where *platform* is one of the following:

MVSUNLD	The file was exported/unloaded on z/OS.
MVSLOAD	The file was exported/unloaded on Windows or Solaris.

Sample JCL

The BRFRUEXP member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility to reformat an export table. This sample is provided as a reference only; modify the JCL for your needs.

Transferring Missing BDW Variable Format Files

Missing BDW (Block Descriptor Word) Variable Format files are created when z/OS VB files are transferred to either Windows or Solaris using the z/OS FTP LOCSITE sub command with RDW parameter specified. FTP strips all block descriptor words during the transfer, so the file produced on the target platform is a non-z/OS-standard variable format file.

Reformatting Requirements for BDW Variable Format Files

If the file is to be transferred back to z/OS, it must be reformatted as follows:

1. Transfer the file back to z/OS in binary mode using FTP.

   The z/OS target file must have a VB format. If the format is FB, there could be pad bytes appended to the last record in the file that could be taken as extra records. If there are pad bytes, a warning is issued by S6BBRFRU.

2. Run JCL to reformat the z/OS file.

   Refer to the following sample JCL.

JCL to Reformat a Missing BDW Variable-Format File

The JCL must contain the following statement:

```
PARM='REFORMAT=NODBW'
```

Sample JCL

The BRFRUBDW member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility to reformat a missing BDW variable-format file. This sample is provided as a reference only; modify the JCL for your needs.

Transferring Workbench Unload Files

To avoid unpredictable changes to file data, the Default Read/Write File Type on Open Systems for the UNLOAD tool must be set to LENGTH_PREFIXED_EBCDIC if the file is to be transferred. This applies even if the file transfer is occurring within the same operating system.

Reformatting Requirements for Workbench Unload Files

Unload Files Created on z/OS and Transferred to Windows or Solaris

These files must be reformatted by S6BBRFRU before they are transferred to Windows or Solaris. Complete the following steps:

1. Run JCL to reformat the file.

   Refer to the following sample JCL.

2. Transfer the file to Windows or Solaris using binary mode and the z/OS FTP LOCSITE sub command with NORDW parameter specified.

3. Load the file on Windows or Solaris.

   Use the TIBCO Object Service Broker administrator's workbench LOAD option or the LOAD tool from the developer's workbench.

Unload Files Created on z/OS and Transferred to z/OS

No formatting is required provided the file is transferred in FTP block (not stream) mode.

Unload Files Created on Windows or Solaris and Transferred to z/OS

These files must be reformatted before they are loaded on z/OS. Complete the following steps:

1. Unload the file from Windows or Solaris in LENGTH_PREFIXED_EBCDIC form.

2. Transfer the file to z/OS using binary FTP.

The z/OS target file must have a format of FB or VB. We recommend that FTP be permitted to allocate the file, except in the case of a very large file where you should preallocate a sufficiently large file.

3. Run JCL to reformat the file.

   Refer to the following sample JCL.

4. Load the file on z/OS.

   Use the TIBCO Object Service Broker administrator's workbench LOAD option or the LOAD tool from the developer's workbench.

Unload Files Created on Windows or Solaris and Transferred to Windows or Solaris

No reformatting is required provided the file is transferred using binary FTP.

JCL to Reformat Workbench Unload Files

The JCL must contain a statement similar to the following:

```
PARM='REFORMAT=platform'
```

where platform is one of the following:

MVSUNLD	File was exported/unloaded on z/OS.
MVSLOAD	File was exported/unloaded on Windows or Solaris.

Sample JCL to Reformat an Unload File Created from the z/OS Workbench

The BRFRUUNL member of the JCL data set contains sample JCL required to run this utility to reformat an unload file created in z/OS. This sample is provided as a reference only; modify the JCL for your needs.

Sample JCL to Reformat an Unload File Created in Windows or Solaris

The BRFRULOD member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility to reformat an unload file created in Windows or Solaris. This sample is provided as a reference only; modify the JCL for your needs.

Transferring TIBCO Object Service Broker Offline Batch Utility Unload Files

Offline batch utility unload files are created by the utilities S6BBRULB, S6BBRULH, or S6BBRULA.

Reformatting Requirements for Offline Batch Utility Unload Files

Files Created on z/OS and Transferred to Windows or Solaris

No formatting is required provided the files are transferred to Windows or Solaris using binary mode and the z/OS FTP LOCSITE sub command with RDW parameter specified. After the files are transferred, they can be loaded using hrnbrtbl or translated using hrnbrnls.

- On Open Systems, when loading using hrnbrtbl or translating using hrnbrnls, ensure that the control file (-c) indicates variable format and *do not specify the -m command line option*. Although the transferred files are VB format on z/OS, after the transfer they no longer contain block descriptor words and are therefore not valid VB files.
- TIBCO Object Service Broker format requirements for z/OS-created VB data sets to be used by offline batch utilities on Windows or Solaris are different from the requirements of Import/Export table data sets. You must therefore know what the intended use of the file on Windows or Solaris is before using S6BBRFRU.

Files Created on Windows or Solaris and Transferred to z/OS

These files must be reformatted on z/OS before they can be loaded or translated. Complete the following steps:

1. Transfer the files to z/OS using binary FTP.

   The z/OS target file must have a VB format. If the format is FB, there could be pad bytes appended to the last record in the file that could be taken as extra records. If there are pad bytes, a warning is issued by S6BBRFRU.

2. Run JCL to reformat the file.

   Refer to the following sample JCL.

3. Load the file using S6BBRTBL or translate it using S6BBRNLS.

JCL for Reformatting Offline Batch Utility Unload Files

The JCL must contain the statement `PARM='REFORMAT=OFFLINE'`.

Sample JCL

The BRFRUOFF member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility to reformat an offline unload file from the Batch utility. This sample is provided as a reference only; modify the JCL for your needs.

S6BBRHDW (CPU Capabilities Report)

S6BBRHDW analyzes the machine where you run it, reporting on the availability of the hardware instructions required to run TIBCO Object Service Broker for z/OS.

You should run this utility on all mainframe systems where you plan to run TIBCO Object Service Broker. You do this to find out if there are restrictions to running TIBCO Object Service Broker on those systems.

Invocation

The S6BBRHDW member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.

This sample is provided as a reference only; modify the JCL for your needs.

Usage Notes

S6BBRHDW issues various machine code instructions and reports that the instruction ran and whether the corresponding facility is installed. In some cases, such as for the STSI instruction, the machine could run the instruction and not perform all the function when doing so. In this case, the instruction is reported with asterisks (***) under the INSTALLED column, with an explanation at the end of the report stating that the full function is not available.

If S6BBRHDW is not APF authorized, some instructions, such as STSI, cannot be tested because the instruction is privileged. In this case, the literal “N/A” appears under the INSTALLED heading. To get the full report, APF-authorize the utility and rerun the report.

Note on the JCL

The SYSPRINT output data set contains the utility report. For saving the report to file, the characteristics of the report data set are: RECFM=VBA and LRECL=137. The block size is unspecified.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	All instructions executed successfully and are implemented on this machine.
4	Some instructions are not completely implemented or are unavailable.
16	SYSPRINT DD statement failed to open successfully.

Return Code	Meaning
20	SYSPRINT DD statement failed to close successfully.
24	S6BBRHDW is not APF authorized.

Sample Report

The following shows the report that comes out of this utility. The specific information contained on your report can vary depending on the capabilities of your machine, the operating system, and the authorization environment available at runtime:

```
DATE: APR 02, 2009                S6BBRHDW -- CPU Capabilities Report                    TIME: 11:41:16
_____________________________________________________________________________________________________

TIBCO(r) Object Service Broker for z/OS may require additional hardware instruction capabilities. S6BBRHDW is
designed to test specific instructions to ensure that your machine will support the required facility.

    OS Level Detected:  z/OS V1 R8
    System Name:        STAR
    S6BBRHDW level:     V520E043

  -----------     ---------     ---------     -------------------------------------
  INSTRUCTION     INSTALLED     MANDATORY             D E S C R I P T I O N
  -----------     ---------     ---------     -------------------------------------
     CLST            YES           YES        STRING INSTRUCTION FACILITY
     JNE             YES           YES        RELATIVE BRANCH FACILITY
     TMH             YES           YES        EXTENDED IMMEDIATE FACILITY
     CLCLE           YES           YES        COMPARE/MOVE EXTENDED FACILITY
     CKSM            YES           YES        CHECKSUM FACILITY
     PLO             YES           YES        LOCKED OPERATION
     TBDR            YES           YES        FLOATING POINT EXTENSIONS
     STCKE           YES           YES        EXTENDED TOD CLOCK FACILITY
     STSI            YES           YES        STORE SYSTEM INFORMATION
                                                Manufacturer:            IBM
                                                Machine type:            2098
                                                Model number:            J04
                                                Sequence Code:           00000000000DF1A3
                                                Plant of Manufacture:    02
     TRE             YES           NO         EXTENDED TRANSLATION-1 TRANSLATE
     CLCLU           YES           NO         EXTENDED TRANSLATION-2 UNICODE
     SRSTU           YES           NO         EXTENDED TRANSLATION-3 UNICODE

All TIBCO(r) Object Service Broker required hardware facilities are installed on this machine. No further
action is required.

S6BBRHDW: Step Return Code is 0000
```

S6BBRIAL (Move ACCESSLOG)

S6BBRIAL moves the ACCESSLOG table, which stores the security audit log, from the MetaStor (segment 0) to another segment. The ACCESSLOG table is located in segment 0 when distributed with TIBCO Object Service Broker. If this utility is used, it should be run immediately after archiving or purging all current data, and successfully shutting down the Data Object Broker.

Prerequisites

Before invoking S6BBRIAL, ensure that the following prerequisites are met:

- Since this utility directly modifies both segment 0 and a target segment, the user invoking it must have write authority to both segments.
- The JCL execution parameter SEGMENT= indicates a valid Pagestore segment number.
- A valid ACCESSLOG occurrence exists on segment 0, including an entry in TABLES where TABLES.SEG is not NULL and an RTIX entry where primary path = a segment 0 page number.
- There is no ACCESSLOG entry on the target segment.
- 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.
- The ACCESSLOG table is archived or purged.
- The Data Object Broker is shut down normally.

Invocation

The JCL required to run S6BBRIAL is as follows (where ## is the target segment):

```
//MOVLOG   EXEC PGM=S6BBRIAL,PARM='SEGMENT=##'
//DBDLIB   DD   DISP=SHR,DSN=dbdlib_data_set_name
//AUDIT    DD   SYSOUT=*
//SYSUDUMP DD   SYSOUT=*
```

Considerations

Note the following when using S6BBRIAL:

- The S6BBRIAL utility moves the ACCESSLOG table from segment 0 to a non-zero segment only. ACCESSLOG can be moved only once and after it is moved off segment 0 it cannot be moved again. If possible, move it to its own segment where it can be kept separate from other data.
- Since this utility updates the segment offline and does not normally perform journaling, you must back up any changes you make to the target segment.

Sample Audit Log

The following illustrates a sample audit log for the S6BBRIAL utility. The audit log must be checked after the run; the message: `S6BBC915I ACCESSLOG INSTALLATION COMPLETE` indicates success:

```
S6BBRIAL              INSTALL ACCESSLOG              DATE 2009 APR 09 TIME 07:41 V520E049

INSTALLING ACCESSLOG ON SEGMENT #:  99

S6BBC910I SEARCH METASTOR FOR ACCESSLOG
S6BBC911I CHECKING METASTOR POINTERS
S6BBC912I INSTALLATION COMMENCED - TARGET SEGMENT
S6BBC913I INSTALLATION COMPLETED - TARGET SEGMENT
S6BBC914I MODIFYING METASTOR POINTERS
S6BBC915I ACCESSLOG INSTALLATION COMPLETE
```

See Also

TIBCO Object Service Broker for z/OS Installing and Operating for information about setting up the ACCESSLOG table.

TIBCO Object Service Broker Managing Security about purging the ACCESSLOG (audit log) table.

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for information about backing up a segment.

S6BBRNLS (Translate File Between Code Pages)

The S6BBRNLS utility is used to translate files between different code pages.

- S6BBRNLS cannot be used to convert files from ASCII to EBCDIC.

Invocation

The S6BBRNLS member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.

This sample is provided as a reference only; modify the JCL for your needs.

If you want to overwrite the input file, the INFILE and OUTFILE statements can be replaced with an INPLACE statement:

```
//INPLACE    DD DISP=OLD,DSN=input_file_name
```

The code page parameters INCP and OUTCP are optional. If you do not specify code pages, these values default to 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 (CNTRL DD statement). This file is identical to the control file used by S6BBRTBL (offline batch load utility) and can be created using BATCHLOAD_CARDS or manually with the editor of your choice. For complete information, refer to the documentation for BATCHLOAD_CARDS in *TIBCO Object Service Broker Shareable Tools*.

When creating the control file, consider the following:

- S6BBRNLS is designed to translate table data unloaded from a TIBCO Object Service Broker table using one of the offline batch unloads (S6BBRULB, S6BBRULH, S6BBRULA).
- S6BBRNLS is designed to translate unloaded table data before it is loaded to a TIBCO Object Service Broker table using the offline batch load (S6BBRTBL).
- The S6BBRNLS input file (CNTRL DD statement) 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.
- S6BBRNLS 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. This could yield an unacceptable result.

- Static values are not eligible for translation.
- If a field is eligible for translation and is also a parameter or primary key field, consider sorting the output file after translation.
- The CHARSET value in the control file has no effect on translation.

Eligible File Fields

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.

Input File (INFILE)

This unload file must be in the correct sequence (ascending parameter 1 through *n* and primary key 1 through *n)*.

Output File (OUTFILE)

The output file must be allocated with the same characteristics as those of the input file (DCB=*.INFILE). The 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 therefore requires sorting prior to the load, a message is issued in the audit file.

Translate In Place (INPLACE)

S6BBRNLS is designed to translate table data unloaded from a TIBCO Object Service Broker table using one of the offline batch unloads (S6BBRULB, S6BBRULH, S6BBRULA). 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 changed with translation, and therefore requires sorting prior to the load, a message is issued on the audit file.

For Translate In Place (INPLACE), if a problem occurs during the S6BBRNLS process, it is possible that your file is no longer usable.

Audit File (AUDIT)

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. If you encounter problems about which you need to contact TIBCO Support, have this file available.

Return Codes

Return Code	Meaning
0	Translation completed successfully.
4	At least one warning is issued.
1210	Transaction terminated abnormally—ABEND.

Sample Audit Log

The following illustrates a sample translation audit log for the S6BBRNLS utility:

```
*S6BBRNLS   NLS CODE PAGE TRANSLATION UTILITY      0   DATE 2009 Apr 11 TIME 08:59

-------------------- TABLE -----------------------
CONTROL - file: S6B.TST.NLS.CNTL                          (TC006001)
INPUT - file: S6B.TST.NLS.IBM278                        (IN1     );
              Data type: Huron; Record format: Variable
HURON TDS Table: SVEN_1,   Character Set: SVEN
OUTPUT - file: SYS07101.T085856.RA000.1234T.NLS.H03       ;
              Data type: Huron; Record  format: Variable

#fields - 650;   #records - 100000;   #parameters - 0;   #secondaries - 0
Input/Output sizes: Record - 34; Minimum Record - 34 Buffer - 65536


FIELD: FIELD1 ---> FIELD1 (PRIMARY KEY)
IN:  type=  ; syn=    C; len= 30; dec= 0; offset=0
OUT: type= S; syn=    C; len= 30; dec= 0; fld= 1; key= 1; prm= 0

END OF TABLE
         IBM-278 ->          IBM-277    S6BBC205I Translation requested

2    S6BBC224I Records Translated

0    S6BBC225I Records Copied

2    S6BBC207E Total Records Processed

2    S6BBC226I Values Changed

2    S6BBC227I Records Changed
```

****

Related Utilities

- S6BBRULB
- S6BBRULH
- S6BBRULA
- S6BBRTBL

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

S6BBRPGC (Pagestore Correction)

You can use S6BBRPGC to reclaim orphan pages detected by S6BBRPTR (the Batch Pointer Check utility). Orphan pages are identified by S6BBRPTR as having been marked as allocated, yet have no connection to any table structure within the segment.

Invocation

Complete the following steps:

1. Run S6BBRPTR.

   Refer to S6BBRPTR (Batch Pointer Check) for sample JCL. An ERRLOG data set is created by the S6BBRPTR utility. S6BBRPTR writes messages to the ERRLOG DD statement indicating which pages are orphaned.

2. Review the output to ensure that the only errors indicated are for orphan pages.

   If other errors are indicated, they must be corrected and S6BBRPTR rerun before proceeding to the next step.

3. Run S6BBRPGC.

   The S6BBRPGC member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.

   This sample is provided as a reference only; modify the JCL for your needs.

   S6BBRPGC reads the error log from S6BBRPTR and designates the orphan pages as free and available for use. For more information on the error log, refer to Error Log (ERRLOG) on page 35.

Since this utility updates the segment offline and does not normally 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 z/OS Managing Backup and Recovery.*

S6BBRPTR (Batch Pointer Check)

You can use S6BBRPTR to validate the integrity of page images. S6BBRPTR reads in an archive file or offline segment and validates the horizontal and vertical pointers. If multiple segments are contained within the same archive data set, you must run this utility against each segment individually.

We strongly recommend that you always run this utility following a standalone backup and after TIBCO Object Service Broker runs a continuous backup.

Input File

The input to the S6BBRPTR utility can be either an archive file from tape or disk or an offline segment. The archive file that you input to this utility must be generated by doing one of the following:

- Using S6BTLBPS (as in a full system backup)
- Merging page data set backups created by S6BTLUPS into one complete segment backup
- Merging your journal data sets with a previous backup (as in a continuous backup)

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for information about backing up a segment.

Invocation

The S6BBRPTR member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility to check an archive.

This sample is provided as a reference only; modify the JCL for your needs.

Note the following about this JCL:

- The segment parameter specifies that segment 01 is to be checked.
- The backup input data set is allocated by DD NAME BACKUP.
- The JCL includes the optional HDR parameter and its corresponding PGHDR DD statement.
- To pointer-check an offline segment, replace the BACKUP DD statement with a DBDLIB DD statement as follows:

```
//DBDLIB    DD DSN=$HLQNONV$.$INSTVER$.DBDLIB.DISP=SHR
```

- The NOTIFY parameter on the JOB card specifies the user ID to be notified of the success or failure of the job. The user ID must be logged on to receive the notification.

In case the user ID specified in the NOTIFY parameter is not logged on at the time the Batch Pointer Check job is run, you should always check the ERRLOG file to ascertain the status of the job.

If Batch Pointer Check detects a segment corruption, rerun the HDR option before passing the output to Support.

Parameters

DATAPG	Optional parameter to request the validation of the data page header entry size. This checks that the entry size is consistent with data page rows on the page.
HDR	Optional parameter to include header information. If included, the PGHDR DD statement must also included.
NOINVS	Optional parameter that requests the secondary index invalidated warning message be suppressed.
SEGMENT	The segment number to be checked.

File Definitions

This JCL references seven different files as output or work files. These are:

PAGES	This internal work file stores selected pages in their entirety.
INDEX	This internal work file is used to retain indexing information.
AUDIT	This report lists information pertaining to the segment processed. It includes table name, errors, and page counts.
ORPHAN	This file contains a list of pages where the system usage indicators did not agree with reference information detected by Pointer Check processing.
REFLOG	This log identifies all referenced pages. It is intended for TIBCO Support use.

PGHDR	This optional file lists all segment Page Header information. It is created only if HDR is included as a parameter in the input parameter string to the utility. Page Header information is intended for TIBCO Support use.
ERRLOG	This log lists error messages detected by the validation process. It provides an easy reference to help you determine if errors occurred. All output is written to the ERRLOG DD statement data set and to the AUDIT DD statement. ERRLOG output can also be used as input to the reclaim orphan pages utility (S6BBRPGC).

The use of each of these files is discussed in detail.

Page Image File (PAGES)

This work file holds a small number of pages that must be retained in their entirety. These pages are saved for a select group of page types that are required during processing. It is unlikely that you need to retain more than 50 of these pages for the pointer validation process. This file contains fixed length 4096 byte records. We recommend that you use VIO space for this file.

Index Information File (INDEX)

The index information file is a work file used to retain indexing information during the pointer validation process. The file is a variable length blocked file and should be defined with a maximum record length of 4800 bytes. We recommend that a VIO type unit be used. As a general guideline, allow about 20 records per cylinder of space.

Audit Log (AUDIT)

The audit log 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.

The following illustrates a compressed version of an audit log. Note that the Pagestore was intentionally modified for this example.

```
S6BBRPTR          POINTER VALIDATION FOR SEGMENT 1    DATE 2009 APR 22 TIME 13:02 V520E049

TABLE/SIX NAME         I/1      D/B/R      H/G        S/6 F/X/0/?       TOTAL    ERRORS


BACKUP  RECORDS READ          55;         55 ACCEPTED;          0 UNUSED PAGES IGNORED


**** SYSTEM ***           1          1        0          0        0          2
@RULESLIBRARY             0          1        0          0        0          1
EVENTRULES                0          1        0          0        0          1
FIELDS                    0         10        1          0        0         11
ORDERING                  0          2        1          0        0          3
PARMS                     0          8        1          0        0          9
SCREENFIELDS              0          1        0          0        0          1
SCREENS                   0          1        0          0        0          1
SCREENTABLES              0          1        0          0        0          1
SELECTION                 0          1        0          0        0          1
TABLES                    1          1        0          0        0          2

S6BBRPTR          POINTER VALIDATION FOR SEGMENT 1    DATE 2009 APR 22 TIME 13:02 V520E049

TABLE/SIX NAME         I/1      D/B/R      H/G        S/6 F/X/0/?       TOTAL    ERRORS

********** ORPHAN PROCESSING INITIATED **********

S6BBRPTR          POINTER VALIDATION FOR SEGMENT 1    DATE 2009 APR 22 TIME 13:02 V520E049

TABLE/SIX NAME         I/1      D/B/R      H/G        S/6 F/X/0/?       TOTAL    ERRORS


**** SYSTEM ***           0          0        0          0       22         22

GRAND TOTALS              2         28        3          0       22         55         0


          START TIME  13:02:41     END TIME  13:02:41
DATASET              READ      REFERENCED  OLDEST TIMESTAMP     NEWEST TIMESTAMP
PAGE1                  53              53 2009APR09 14:40:27  2009APR09 18:12:51
PAGE2                   1               1 2009APR09 18:12:51  2009APR09 18:12:51
PAGE3                   1               1 2009APR09 18:12:51  2009APR09 18:12:51
```

Entries prefixed with a hyphen (-) indicate a secondary index with that secondary key field name.

The output for non-segment 0 analysis shows several tables, including @RULESLIBRARY, EVENTRULES, FIELD, TABLES, and others that are actually on segment 0. They are automatically inserted in segment 1 due to reserved page formatting and do not indicate an error.

Orphan List (ORPHAN)

The orphan file lists all pages where there are discrepancies between the system and the actual availability and usage indicators.

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 must be a sequential file with a record length of 80 bytes, fixed block. Since orphan pages are an operating exception, a static allocation of five tracks should suffice. Retain the orphan list file so you can compare the results with those generated by a future run. Contact TIBCO Support if the number of orphan pages continues to increase. The utility S6BBRPGC can be used to reclaim orphan pages.

If your report shows `REFERENCED BUT INDICATED FREE` pages, stop using the segment and contact the TIBCO Support immediately. If processing continues on the segment, further data corruption can occur which may result in loss of user data in multiple tables..

Output Reference Log File (REFLOG)

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 identify how the page fits in the Pagestore. The reference log has the following format:

Column 01-08	Page number.
Column 13-20	Previous page number in the chain.
Column 25-32	Next page number in the chain.
Column 37-37	Page type code.
Column 42-57	Name of the table referencing the page.

Column 62-69	Page number of parent page.
Column 73-77	Entry number on parent page.

The REFLOG file must be a sequential file with an 80 byte record length, fixed block. As a guideline, you should allocate one track of REFLOG for every four cylinders of space. The information in the reference log should be retained. It is intended to help the TIBCO Support identify problems should they arise.

Page Header List (PGHDR)

This file contains a list of the header information for each page read. This information is often needed when investigating problems with the structure of the Pagestore.

Error Log (ERRLOG)

The error log lists tables with errors. Refer to Audit Log (AUDIT) on page 32 for more information about how errors are reported. If ERRLOG is to be used as input to S6BBRPGC to reclaim orphan pages, it should be created as a 132-byte FB file. Otherwise, it can be directed to SYSOUT.

S6BBRPTR extracts the timestamp from page 1 and puts it in the ERRLOG in the form `ID=xxxxxxxxxxxxxxx`. S6BBRPGC checks this timestamp against the then current page 1 before acting on any 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
```

Return Code Settings

To identify validation errors more easily, the return code is set according to the most severe error encountered. A non-recoverable error results in an abend.

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.

See Also

TIBCO Object Service Broker Messages With Identifiers for details on abend codes.

S6BBRSET (Batch Segment Re-initialization)

S6BBRSET re-registers empty tables into a TDS segment.

S6BBRSET 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 do the following:

- Rebuild the resident table index
- Create the empty tables
- Rebuild the secondary index structures for the deleted tables

Having used S6BBRSET to build the empty table and index structures, you can reload the data using either the LOAD tool or Batch Load utility (S6BBRTBL).

Invocation

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

1. Delete and redefine the TIBCO Object Service Broker segment (for example, using IDCAMS).
2. Format the segment using the S6BTLFPS 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 z/OS Managing Backup and Recovery.*
3. Run S6BBRSET.

   The S6BBRSET member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.

   This sample is provided as a reference only; modify the JCL for your needs.

The S6BBRSET utility is passed four parameters:

BROWSE	Indicates whether this execution is a test (Y) or production (N) run.
SEGMENT	The segment number you want to re-initialize. The segment number is a one, two, or three digit decimal number. The maximum segment number is 255. It must be offline.

TDS	The Data Object Broker communications identifier of an active TIBCO Object Service Broker system (one to eight characters).
MDL	The MDL Execution Environment communications identifier (one to eight characters).

If the tables were unloaded with the UNLOAD tool, the empty tables and secondary indexes can be reloaded using the LOAD tool after they are in place.

Constraints

The following constraints apply:

- The target can be any segment other than segment 0.
- A TIBCO Object Service Broker system with the same DBDLIB 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 normally 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 z/OS Managing Backup and Recovery.*

Sample Report

The execution report provides information about the resident tables and the secondary indexes that are rebuilt on the reformatted segment. A sample report for a TDS segment follows.

```
S6BBRSET  REBUILD EMPTY TABLES IN FORMATTED TDS SEGMENT 001    DATE: 2009 JUN 18
TIME: 08:31:48      V520E049      PAGE:     1

   TABLE NAME     ACTION      SECONDARY INDEX FIELDNAMES

$TABWITHLOC       ADDED
$TABWITHLOCDATA   ADDED
$TABWITHNOLOC     ADDED
@RULESLIBRARY     KEPT
DZYP4             ADDED
DZYTDS1           ADDED
DZYTEST           ADDED
DZYTESTTDS        ADDED
EVENTRULES        KEPT
FIELDS            KEPT
ORDERING          KEPT
PARMS             KEPT
RAJTD_C#7QU8B9    ADDED
SCREENFIELDS      KEPT
SCREENS           KEPT
SCREENTABLES      KEPT
SELECTION         KEPT
TABLES            KEPT
TABWITHLOC        ADDED
```

```
TABWITHLOCDATA     ADDED
TABWITHNOLOC       ADDED
TEST_3PAR          ADDED
T7FZDYH            ADDED
T7FZDYHB           ADDED
T76UAJ9            ADDED

****************** TARGET SEGMENT UPDATED - RTIX-IX **********************

****************** TARGET SEGMENT UPDATED - RTIX    **********************

PAGES CREATED:
RTIX-IX PAGES:     1   RTIX PAGES:     1 ALLOC PAGES:     5 SECIX PAGES:          0
```

If an error condition is encountered, S6BBRSET execution terminates. You should investigate the cause of the error, correct it, and run S6BBRSET again.

See Also *TIBCO Object Service Broker Messages With Identifiers* for messages associated with this utility.

S6BBRSIX (Batch Secondary Index Build for TDS Tables)

S6BBRSIX is a batch utility that provides a fast method for building one or more secondary indexes on a large, existing TDS table.

S6BBRSIX versus SIXBUILD

S6BBRSIX offers significant performance improvements over the online secondary index build tool (SIXBUILD) because it builds multiple secondary indexes from just one pass through the data. The segment must be offline. If the table occupies more than 3000 pages or exceeds site limits for online processing, you must use S6BBRSIX.

For tuning tips and information helpful when processing very large tables, refer to Appendix B, S6BBRTBL/S6BBRSIX Tuning, on page 143.

Invocation

Complete the following steps to run S6BBRSIX:

Step 1: Describe Input and Output

Describe the input and output to the utility by coding control cards. You can use the SIXBUILD_CARDS utility to define a data set containing the necessary control cards.

The SIXBUILD_CARDS utility is described in detail in SIXBUILD_CARDS (Prepare Cards for Batch Secondary Index Build).

Step 2: Define and Run JCL to run S6BBRSIX

The S6BBRSIX member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.

This sample is provided as a reference only; modify the JCL for your needs.

Step 3: Identify the Secondary Index Fields

When you use the Batch Secondary Index Build utilities, the segment containing the table must be offline. The utility cannot access the table definition to record the secondary indexes that were created because the definition is stored in the MetaStor. Consequently, after the batch utility is run, the secondary indexes are built and usable but you must edit the table's definition to make the new secondary index fields available to the Table Definer.

To mark the fields that have secondary indexes, edit the FIELDS table using the Table Editor. The table is parameterized by table name, so if you had a table called EMPLOYEE, you would edit the FIELDS table from the workbench as follows:

```
ED edit table ==> FIELDS(EMPLOYEE)
```

Enter an uppercase S in the KEYTYPE column next to the field names of the fields that have secondary indexes. If the KEYTYPE column already contains a P and that field is also a secondary index, enter an uppercase Q. The next time you look at the table definition through the Table Definer, you see an S or Q next to these fields in the KEY field.

Step 4: Back up the Segment

To ensure the integrity of your continuous backup after you run the S6BBRSIX utility, you must back up the segment containing the new data.

See Also

TIBCO Object Service Broker Shareable Tools for more information about the SIXBUILD tool.

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for information about backing up a segment.

TIBCO Object Service Broker Messages With Identifiers for messages associated with this utility.

S6BBRTBL (Batch Load)

The S6BBRTBL utility performs the following functions:

- Load large volumes of data into a predefined TIBCO Object Service Broker table in the shortest time possible
- Accept input from a sequential or VSAM 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

- If you are loading data from an unloaded table, you must unload each table to a separate file and run S6BBRTBL individually for each of the unloaded tables.
- For tuning tips and information helpful when loading very large tables, refer to Appendix B, S6BBRTBL/S6BBRSIX Tuning, on page 143.
- Refer to Appendix C, Null Handling, on page 151 for a description of how null values are treated by the batch load utilities. S6BBRTBL considers certain field values to be *null equivalents*, whether or not they came from TIBCO Object Service Broker tables.

Invocation

These are the tasks in using the batch load utility:

- Define the Table, page 42
- Build Secondary Indexes, if Required, page 42
- Define your Input and Output with Control Cards, page 42
- Program Data Validation or Adjustments if Required, page 42
- Define Your JCL and Run the Batch Load Utility, page 45
- Back Up the Segment and Run Batch Pointer Check, page 46

These tasks are described in detail in the following sections.

You should also consider the effect on your continuous backup.

Task A Define the Table

Define the table in TIBCO Object Service Broker if it is not already defined. For information on defining a table, refer to *TIBCO Object Service Broker Managing Data*.

Task B Build Secondary Indexes, if Required

If the table is new and you require secondary indexes, run the SIXBUILD tool against the empty table for each field on which you require a secondary index. For information about the SIXBUILD tool, refer to *TIBCO Object Service Broker Shareable Tools*.

If the table already contains data and you want to add secondary indexes, you should use the S6BBRSIX (Batch Secondary Index Build for TDS Tables) utility.

Task C Define your Input and Output with Control Cards

Define your input and output data with control cards. You can define these control cards in one of two ways:

- TIBCO Object Service Broker provides the BATCHLOAD_CARDS utility to help you prepare the control cards.
- You can also define your control cards manually. For complete specifications, refer to Appendix A, Defining Batch Load Control Cards Manually, on page 133.

Task D Program Data Validation or Adjustments if Required

With the batch load utilities, you can program data validation or adjustments using the three exit points provided. These exit programs must be written in assembler. Informational messages are returned if these exit points are not used.

You can use the three exit points to tailor your data during the load process. These exit points occur:

- Immediately after an input record is read (post read exit)
- After the data is organized into fixed length format (post fixed length formatting)
- Just before the TIBCO Object Service Broker occurrence is formatted (pre-TIBCO Object Service Broker formatting)

These exit points are discussed in detail in the following sections.

Post Read Exit

You can use this exit to validate records or to adjust certain field values immediately after an input record is read.

The post read exit gains control immediately following the read of the input record. The following three parameters are passed in a parameter list pointed to by register 1:

- Input record I/O area address
- Internal definition table start address
- Work area address.

  An 80-byte work area is provided to the exit for its own use. This work space is exclusive to this exit point.

Your program can manipulate a field value in any way, provided that you stay within the semantic and syntax type defined for that field.

Upon return from the exit, register 15 must be zero to continue processing the record or non-zero to ignore it. The exit must be assembled and linked in the step library; its name and entry point must be S6BBRTL1.

Post Fixed Length Formatting

You can use this exit to validate records or to adjust certain field values after the data is organized into fixed length format. The record is in sequence as indicated by the parameter and field index tables. The exit passes the following parameters in a parameter list pointed to by register 1:

- Internal fixed length record area
- Internal definition table start address (refer to Internal Table Definition on page 44 for details)
- Parameter index list (pointers to definition table in TIBCO Object Service Broker parameter order)
- Field index list (pointers to definition table in TIBCO Object Service Broker field order)
- Work area address

  An 80 byte work area is provided to the exit for its own use. This work area is exclusive to this exit point.

Upon return from the exit, register 15 must be zero to continue processing the record or non-zero to ignore it. The exit must be assembled and linked in the step library. Its name and entry point must be S6BBRTL2.

Pre-TIBCO Object Service Broker Formatting

You use this exit when you load tables with an IDgen specification (that is, a system generated primary key). This exit gains control after the primary key is generated and just before the fixed length internal record is compressed into TIBCO Object Service Broker format. You can use this exit to manipulate the primary key value.

The definition table entry for the primary key is the first pointer in the field index table.

The following parameters are passed in a parameter list pointed to by register 1:

- Internal fixed length record area
- Internal definition table start address
- Field index list (pointers to definition table in TIBCO Object Service Broker field order)
- Work area address

  An 80 byte work area is provided to the exit for its own use. This work space is exclusive to this exit point.

Upon return from the exit, register 15 must be zero to continue processing the record or non-zero to ignore it. The exit must be assembled and linked in the step library. Its name and entry point must be S6BBRTL3.

Internal Table Definition

The following contains the definition table DSECT used by the batch load program. This assembler code is in the TIBCO Object Service Broker release package in the HRNBRDT member of the MACRO data set. The table start address is passed in the three user exits described earlier in this section.

```
DTDSECT  DSECT
*
*        DEFINITION CONTROL CARD TRANSLATION DSECT
*
DTINAME  DS    CL36               INPUT FIELD NAME
DTHNAME  DS    CL16               NAME OF FIELD OR PARM
*
DTILEN   DS    H                  INPUT FIELD LENGTH
DTIDEC   DS    H                  INPUT DECIMAL POSITION
DTIOFFST DS    H                  OFFSET IF FIXED LENGTH
DTISYNTX DS    CL1                INPUT FIELD SYNTAX
*
DTHTPSYN DS    0CL2               TYPE/SYNTAX COMBINED
DTHTYPE  DS    CL1                SEMANTIC TYPE
DTHSYNTX DS    CL1                SYNTAX
DTHKYTYP DS    CL1                KEY TYPE
DTHLEN   DS    H                  OUTPUT LENGTH
DTHDEC   DS    H                  OUTPUT DECIMAL POSITION
```

```
*
DTHFLD#  DS    H                FIELD NUMBER
DTHPRM#  DS    H                PARM NUMBER
DTWOFFST DS    H                OFFSET WITHIN FL DATAROW
DTXOFFST DS    H                OFFSET WITHIN INDEX RECORD
         DS    H                UNUSED
DTTEST@  DS    AL4              TEST CONVERSION VALUE ADDRESS
         DS    F                UNUSED
DTKDESC@ DS    A                ADDR OF KDESC OF THIS KEY
DTHNULLF DS    CL1              NULL FLAG; NON-ZERO = NULL
         DS    CL3              UNUSED
DTNULVAL DS    CL16             USER NULL REPRESENTATION
DTNULCNT DS    F                COUNT # DEFAULT NULLS
         DS    A                UNUSED
DTLENGTH EQU   *-DTDSECT        LENGTH OF ENTRY
```

Task E Define Your JCL and Run the Batch Load Utility

After assembling and linking your option exit point programs, you can define your JCL and run the batch load utility.

Execution JCL

The S6BBRTBL member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility for a TDS table.

This sample is provided as a reference only; modify the JCL for your needs.

Following are descriptions of the DD names in the sample JCL:

CNTRL	Fixed block 80 bytes. This file contains the control cards describing the input file and the table you want to load. Create this file manually or with the control card preparation facility.
INPUT	Fixed or variable block; a partitioned data set member, a sequential data set, or a VSAM data set. This file contains the data you want to load into a table.
AUDIT	Record format FBA, 132 bytes. If you specify a data set, make the block size a multiple of 132. This file contains the output execution audit trail.

Task F Back Up the Segment and Run Batch Pointer Check

- To ensure the integrity of your continuous backup after running the batch load utility, you must back up the segment containing the new data. For information about backing up a segment, refer to *TIBCO Object Service Broker for z/OS Managing Backup and Recovery.*
- If S6BBRTBL fails, you must run the S6BBRPTR (Batch Pointer Check) utility against the segment or its backup to ensure the integrity of your data.

Messages Batch load messages are documented in *TIBCO Object Service Broker Messages With Identifiers*.

S6BBRULA (Recover TDS Table From Archive)

S6BBRULA is a TIBCO Object Service Broker recovery utility that restores one or more TDS tables from an archive (backup) file. You can:

- Recover TDS data from an archive file in unload format
- Select and unload table instances from parameterized tables or entire tables
- Present the recovered data in fixed format to allow modification and further selection

When the data is recovered to a file, the file can be used as input to S6BBRTBL (batch load utility) or it can be reloaded as an import file. If loaded as an import file, you must account for null handling in your rules.

Refer to Appendix C, Null Handling, on page 151 for information on preserving null values in data fields. The unload and subsequent load assume specific values in the unloaded file that represent nulls.

Constraints

The following constraints apply:

- If you use either of the batch unload utilities to unload data, you can reload the data by using the batch load utility or by treating the data as a TIBCO Object Service Broker import file. However, you cannot treat the data as an import file when binary or packed fields are to contain numeric syntax null values. When using the import server, you must use a special value string to represent nulls (null equivalent field values) in the unload file or revert to the oldnull option. Refer to Appendix C, Null Handling, on page 151 for more information.
- All the tables to be recovered must reside on the same segment.
- Each table requested must be defined by control records that represent the table layout at the time the archive was created.
- Multi-volume tape files must be cataloged.
- Only TDS tables can be unloaded.
- If you are using FTP to transfer unloaded data between z/OS and Windows or Solaris, transfer the files in binary and specify the z/OS FTP LOCSITE subcommand with the RDW parameter.

Processing Overview

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

1. The first phase scans the archive file and extracts page chain information, primary, and group index chain information.

   Group index pages are written to a temporary data set.

2. The second phase identifies which data pages contain the information for all the unloaded tables.

3. The third phase extracts the data occurrences from the required pages and produces a work file that is then sorted into appropriate order based upon parameter values within the table name.

   The output from the third phase is the RECOVER file.

Invocation

The following sections:

- Explain how to use this utility
- Take you through a sample situation using S6BBRULA
- Explain how to recover selected table instances of a parameterized table
- Explain how to restore the table under a new name and segment

The sample situation assumes that one table on segment 1 is lost or corrupted. If you are unloading all the table instances from a parameterized table, the process would be the same. For information about recovering particular table instances, review the sample and refer to "Selecting Records During Processing".

Before using S6BBRULA, ensure that the backup level is appropriate for your requirements. In the sample situation, S6BBRULA is used to recover the table from the most recent backup. This backup can be merged with outstanding journals to create a more current backup. For more information about this procedure, refer to *TIBCO Object Service Broker for z/OS Managing Backup and Recovery.*

Step 1: Define the Table to Recover

To define the tables that you want to recover, code control cards as described in this section. Your definition must match that of the table when it was archived. You can use the BATCHUNLD_CARDS utility, which provides a screen interface for you to specify the table or table instances to unload from the archive, and creates the control cards and selection file for you. Refer to the documentation for BATCHUNLD_CARDS in *TIBCO Object Service Broker Shareable Tools*.

If you cannot use BATCHUNLD_CARDS because TIBCO Object Service Broker is not available, you must define the control cards manually according to the format described below.

Each record in the control cards is 80 characters long and has a fixed format. The fields are spaced apart to conform to the layout used by a number of TIBCO Object Service Broker batch utilities such as the batch load utility (S6BBRTBL).

Table Control Card

The first card in the control card file is a table control card. This control card specifies the number of tables specified in the control card file and must be the first entry in the file. The layout is as follows:

Field	Position	Description
Type	07 – 07	Control card indicator asterisk (*).
Count	60 – 63	Number of tables present in the control card file.

Table Definition Cards

These cards define each of the tables to be recovered as follows:

Field	Position	Position & Description
Card type	01 – 01	H – TIBCO Object Service Broker definition card identifier.
Table number	03 – 05	Table number (0 through 999).
Type	07 – 07	Card type (sequence within table T, P, F, R) T – Table header P – Parameter F – Field R – End of table block
Name	09 – 24	Table, parameter, or field name depending upon type
For fields other than raw data (RD) or Unicode (UN):		
Semantic Type	46	Field/parameter semantic type
Syntax	48	Field/parameter syntax
For fields of syntax RD or UN:		
Syntax	45 – 48	Field syntax
Continue for all fields:		

Field	Position	Position & Description
Defined length	49 – 52	Defined length (parm/field type cards) Parameter count (table type cards)
Decimal	54 – 56	Decimal position (parm/field type cards) Field count (table type cards)
Offset	60 – 63	Offset of parameter/field in output text (the 4-byte variable length and 16-byte table name are excluded from the offset). This value is optional; the offset is calculated if left blank.
Null-equivalent value	64 – 79	Null-equivalent value; syntax B or P only

Sample Table Definition

The layout you are describing must match the definition contained in the backup file. The following contains a description of the table definition of the sample table:

```
Table Definition for : SCRIPTCOMMANDS

                Field Definitions
 Field Name    Semantic Type      Syntax      Len    Dec    Key Order Rqd
------------- -------------      ---------   ---    - --   ---- ----- ---
 COMMAND       S String           C Fix Char  16      0      P
 RULE          I Identifier       C Fix Char  16      0
 USAGE         S String           C Fix Char  20      0
 PURPOSE       S String           V Var Char  35      0
```

Sample Control Cards

The following illustrates a sample control card file for this table:

```
         *                                                    0002
   H 000 T SCRIPTCOMMANDS                                  000 004
   H 000 F COMMAND                                     S C 016 000
   H 000 F RULE                                        I C 016 000
   H 000 F USAGE                                       S C 020 000
   H 000 F PURPOSE                                     S V 035 000
   H 000 R
```

Step 2: Customize the Sample JCL

To customize the JCL, specify input files, output files, and runtime parameters. The files and runtime parameters used by S6BBRULA are described below.

Files Used by S6BBRULA

Files used by S6BBRULA are as follows:

Name	Description and Use
CNTRL	Fixed block, 80 bytes. Input table definition control card file.
ARCHIVE	Variable block (sequential) or VSAM. Input TIBCO Object Service Broker backup of segment.
AUDIT	SYSPRINT type file. Output execution audit trail including table definitions and summaries for each processing step.
PAGES	Fixed block 4096 byte. Output/input file retains group index page images between pass 1 and 2. This is a temporary file that you do not need to retain. VIO type device is recommended.
WORK	Variable block, 4100 bytes maximum. The output/input file retains raw extracted data with sequencing fields and the table name at the beginning of the record. This file is used as input to the internal sort that orders the entries and removes the sequencing information. This is a temporary file that you do not need to retain afterwards. VIO type device is recommended.
RECOVER	Variable block, 4100 bytes maximum. This is the final output file with extracted data in proper sequence for loading.

There are additional files if you choose to break during execution. These files and the necessary runtime parameters are described in Selecting Records During Processing on page 54.

Runtime Parameters

There are two runtime parameters that you must specify. These are:

Parameter	Description
Segment	The segment number of the Pagestore that stores the tables to be unloaded. If omitted, the default is segment 0.
Pages	The total number of pages contained in a specified segment, including both allocated and unallocated pages. It is better to overestimate the size than underestimate it. Default: 100,000 Acceptable range: 5400 to 99,999,999 **Note** Do not include commas in the parameter value (for example, specify 100000).

There are additional runtime parameters that you can leave at their default values unless error messages indicate that a particular value should be modified. The additional parameters are listed in Runtime Storage Parameters on page 58.

Sample JCL

The S6BBRULA member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.

This sample is provided as a reference only; modify the JCL for your needs.

Files and parameters used are as described above.

The JCL includes the following entries:

`//ARCHIVE`	Points to your backup file.
`//RECOVER`	Contains the unloaded tables.
`//AUDIT`	Contains the execution report.

Step 3: Run S6BBRULA and Review Results

Submit the JCL and review the execution report. Here is a sample execution report:

```
S6BBRULA            TABLE RECOVERY ARCHIVE EXTRACT   DATE 2009 MAR 17 TIME 1

PROCESSING SEGMENT     1

REQUIREMENT TYPE     % OF PAGES    PAGE ESTIMATES   MEMORY REQUIRED

 DATA                    85               8,500           170,000
 PRIMARY INDEX            3                 300             2,400
 GROUP INDEX             10               1,000            20,000
 GROUP INDEX INDEX        2                 200             1,600
 TABLE INSTANCE           1                 400           105,600
 PAGE STORE/TOTAL         **             10,000           299,600

S6BBRULA            TABLE RECOVERY ARCHIVE EXTRACT   DATE 2009 MAR 17 TIME 1

TABLE NUMBER    0   TABLE NAME SCRIPTCOMMANDS

FIELDS FOR TABLE - SCRIPTCOMMANDS
          NAME         TYPE  SYNTAX  LENGTH  DECIMAL  OFFSET
     ***************    *      *      ***     ***      ****
     COMMAND             S      C       16       0         0
     RULE                I      C       16       0        16
     USAGE               S      C       20       0        32
     PURPOSE             S      V       35       0        52

S6BBRULA            TABLE RECOVERY ARCHIVE EXTRACT   DATE 2009 MAR 17 TIME 1

S6BBA011I PASS ONE PROCESSING INITIATED
S6BBA012I PASS ONE; 000008104 RECORDS READ, 000008104 ACCEPTED,
S6BBA013I PASS TWO PROCESSING INITIATED
S6BBA014I GIX TABLE COMPRESSION COMPLETE INPUT=00043767, OUTPUT=00000071
S6BBA015I DATA TABLE COMPRESSION COMPLETE INPUT=00043839, OUTPUT=00000072
S6BBA019I PASS TWO; GROUP INDEX PAGES READ 000000000, 000000000 PROCESSED
S6BBA020I PASS THREE PROCESSING INITIATED
S6BBA021I PASS THREE; 000000445 RECORDS READ, 000000445 ACCEPTED,

S6BBA022I SCRIPTCOMMANDS   0000000000000055 RECORDS UNLOADED
RECOVERY START TIME 12:01:37, END TIME 12:01:46
```

Browse the RECOVER data set.

Records in the RECOVER data set store the table name as a 16-byte header. You must account for this 16-byte header when you describe the record format to the batch load utility (S6BBRTBL).

Selecting Records During Processing

If you do not need to recover all the instances of a parameterized table, you can select specific parameter values.

Parameterized Table Example

The following illustrates the definition of an example parameterized table. The sample table is parameterized by USERID and is used to explain record selection:

```
COMMAND==>                        TABLE DEFINITION

  Table:EMPLOYEE            Type:TDS      Unit:PER             IDgen:N

  Parameter Name Typ Syn Len Dec    Reference    '  Event Rule    Typ Acc
 ---------------  -   -  --- --  -------------- '---------------- -   -
_ USERID          I   C   16  0                 '
_                                               '
  Field  Name    Typ Syn Len  Dec Key Ord Rqd  Default     Reference
 ---------------  -   -  ----  --  -   -   - ----------- ---------------
_ EMPNO           I   P     3   0  P
_ LNAME           S   C    22   0
_ POSITION        S   C    14   0
_ MGR#            I   P     3   0                        MANAGER
_ DEPTNO          I   B     2   0
_ SALARY          Q   P     4   2            0.00
_ HIREDATE        D   B     4   0
_ ADDRESS         S   V    38   0
_ CITY            S   C    20   0
_ PROV            S   C     3   0
_ P_CODE          S   C     7   0
```

Sample Control Cards

The following example shows the control cards for this table:

```
        *                                                          001
H 000 T EMPLOYEE                          001  011
H 000 P USERID                        I   C  016  000
H 000 F EMPNO                         I   P  003  000
H 000 F LNAME                         S   C  022  000
H 000 F POSITION                      S   C  014  000
H 000 F MGR#                          I   P  003  000
H 000 F DEPTNO                        I   B  002  000
H 000 F SALARY                        Q   P  004  002
H 000 F HIREDATE                      D   B  004  000
H 000 F ADDRESS                       S   V  038  000
H 000 F CITY                          S   C  020  000
H 000 F PROV                          S   C  003  000
H 000 F P_CODE                        S   C  007  000
```

Selecting Parameter Values

Parameter values can be selected internally or externally.

Internal Selection

If you know the parameter values that you want to select before you initiate S6BBRULA execution, you can use a selection file to specify table instances.

External Selection

To select specific parameter value sets externally, you can run S6BBRULA in Break mode to halt processing and allow you to select parameter values according to your criteria. This procedure is described in Defining Table Instances Externally on page 57.

Additional Selection Files

Additional files are also required for selection:

SELECT	Fixed block, 530 bytes. This file contains selection criteria for the parameter values. It is required for internal parameter value selection only.
BRKDATA	Variable block, maximum length 40. This output file is created in phase 2 of processing if the BREAK parameter is specified. It retains data page header and requirement information.

BRKPVS	Variable block, maximum length 300. This output file is created in phase 2 of processing if the BREAK parameter is specified. It retains parameter value information so that selection can be done to eliminate unwanted table instances of a parameterized table.
CONTDATA	Variable block, maximum length 40. This input file is read at initialization when in CONTINUE processing mode (refer to BRKDATA).
CONTPVS	Variable block, maximum length 300. This input file is read at initialization when in CONTINUE processing mode (refer to BRKPVS).

Defining Table Instances Internally

To define files internally, the selection criteria in a SELECT file must be 530 bytes. This can be generated using the BATCHUNLD_CARDS utility. For more information, refer to the documentation for BATCHUNLD_CARDS in *TIBCO Object Service Broker Shareable Tools*.

If you cannot use BATCHUNLD_CARDS, you must define the selection file manually in the following fixed format:

Fields	Position	Description
Table	001 – 016	Table name
Include/exclude	017 – 017	I-include, E-exclude
Operator	018 – 018	Relational operator (=, >, <)
Parm 1	019 – 146	Parameter 1 value
Parm 2	147 – 274	Parameter 2 value
Parm 3	275 – 402	Parameter 3 value
Parm 4	403 – 530	Parameter 4 value

The first record in the selection file is a control record. The control record must have a table name of "**********" (16 asterisks). The first four bytes of the Parm 1 field must contain a count of the number of records in the SELECT file, excluding the control record. The following illustrates an example of the selection file for this table:

```
**********                                        0001
EMPLOYEE                                       I=EDUC
```

Defining Table Instances Externally

The alternate way to extract table instances is to define the instances externally by executing S6BBRULA in Break and Continue modes. To halt S6BBRULA processing, you can specify a runtime parameter of BREAK. To continue with the unload after doing some external processing, you run S6BBRULA with a runtime parameter of CONTINUE. The parameters are described here:

Parameter	Explanation
Break	(Optional) Halt processing after all the required data pages have been tagged and data chain and parameter value files have been produced. To eliminate table instances of specific parameter values you do not want to unload, review the parameter value file (BRKPVS).
Continue	(Optional, after a break) Continue processing after external parameter value selection is complete.

Break Execution JCL

The BRKEXEC member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run the table unload program, halting it after internal parameter value selection (PVS) and data page selection.

This sample is provided as a reference only; modify the JCL for your needs.

The BRKPVS data set contains a header record and a record for every table instance extracted from the archive. The header record contains a count of the number of parameter value records. You can edit this file and delete the parameter value records for the table instances you do not want to recover.

Do *not* edit the header record. We recommend that you make a backup copy of the BRKPVS file before you edit it and use it as input to the Continue process as CONTDATA and CONTPVS.

Continue Execution JCL

The CONTEXEC member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to continue the table unload process after a break run.

This sample is provided as a reference only; modify the JCL for your needs.

Runtime Storage Parameters

The following runtime parameters are used in the calculation and distribution of allocated storage. These parameters allow you to tune the execution of S6BBRULA. With the exception of the PAGES parameter, all these storage parameters should be left at their default values unless a previous execution failure indicates that modification is necessary.

Parameter	Description
Pages	An estimate of the total number of pages on the archive file for the specified segment. It is better to overestimate the size than to underestimate it. Default: 100,000 Range: 5400 to 99,999,999 **Note** Do not embed commas in the parameter value (for example, specify 100000).
Data	Percentage of total pages that are data pages. Default: 85 per cent. Range: 70 to 95 percent. Storage requirement = total pages * data% * entry length.
Index	Default: 3 percent. Range: 1 to 20 percent. Storage requirement = total pages * index% * entry length.

Parameter	Description
GIX	Percentage of total pages that are group index pages. Default: 10 percent. Range: 0 to 30 percent. Storage requirement = total pages * gix% * entry length. **Note** If the tables being unloaded is not parameterized, the GIX percentage can be zero.
HIX	Percentage of total pages that are the first page in a group index chain. Default: 2 percent. Range: 0 to 10 percent. Storage requirement = total pages * hix% * entry length. **Note** If the tables being unloaded is not parameterized, the HIX percentage can be zero.
PVS	Percentage of GIX pages that are required by the tables being unloaded. Default: 5 percent. Range: 0 to 30 percent. Storage requirement = total pages * gix% * pvs% * average GIX entries per page (100) * entry length (264). **Note** If the tables being unloaded is not parameterized, the PVS percentage can be set to zero.

Step 4: Restore the Table Under a New Name and Segment

The following steps describe how to restore an archived file to a new name or segment, or both. This example uses the EMPLOYEE table that resides on segment 2 and restores it as EMPLOYEE_RESTORE on segment 3.

1. Recover the EMPLOYEE table using S6BBRULA.
2. Define a new table called EMPLOYEE_RESTORE and specify that it resides on segment 3.
3. Using the BATCHLOAD_CARDS utility, create control cards for the table EMPLOYEE_RESTORE.
4. Take segment 3 offline.

5. Run S6BBRTBL (batch load) to load the data from the flat file into EMPLOYEE_RESTORE.

These procedures can have an adverse effect on your continuous backup. When possible, take full backups before attempting to restore files from an archive. Refer to *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for more information.

S6BBRULB, S6BBRULH (Batch Unload)

S6BBRULB and S6BBRULH (Batch Unload) provide the following functions:

- Unloading large volumes of data from a single TDS table
- Unload capability on a system that is either active or shut down
- Table instance selection (within parameterized tables)
- Data in a readily usable format (a sequential flat file)

Refer to Appendix C, Null Handling, on page 151 for information on the method used by the unloads to preserve null values in the unload file. The unloads use specific field values to represent nulls in the unload file.

Selecting the Correct Utility

Select the appropriate utility based on whether the data you want to unload is online or offline:

TIBCO Object Service Broker condition…	Condition of Pagestore segment containing data to be unloaded	Use utility…
Running	Online	S6BBRULH
Running	Offline	S6BBRULB
Not running	---	S6BBRULB

S6BBRULH extracts all data in a table or selected table instances of a parameterized table. When unloading data from an online database, consider the effect on the resulting unloaded copy of any updating activity going on in the table being unloaded.

Invocation

To use the batch unload utilities complete the following tasks:

1. Define Input and Output with Control Cards, page 61
2. Define and Submit Execution JCL, page 63
3. Review the Audit Trail Report, page 64

These tasks are described in the following sections.

Task A Define Input and Output with Control Cards

To describe the input and output to the Batch Unload utilities, you need to:

1. Allocate any necessary data sets.

2. Define the control cards using BATCHUNLD_CARDS.

   Refer to the documentation for BATCHUNLD_CARDS in *TIBCO Object Service Broker Shareable Tools*.

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

Input Files

You provide input specifications to either of the batch unload utilities in two files:

CNTRL	The control card file specifies the definition of the table to be unloaded. The CNTRL file should be defined as fixed block, record length 80.
SELECT	The selection file is optional. It contains criteria for selecting table instances from a parameterized table set. If a SELECT file is required, it should be defined as fixed block, record length 530.

Output Files

The following output files are used:

UNLOAD	The unload file contains all the data extracted during the execution of the batch unload process. The layout of the text area within the file is reported in the AUDIT trail. The UNLOAD file should be defined as variable block, with a maximum record length of 4100.
AUDIT	This SYSPRINT file contains the table definition, selection criteria, table instances unloaded, and summary information. Refer to the sample audit report in Review the Audit Trail Report on page 64.

In addition to these files, the S6BBRULB utility (used when TIBCO Object Service Broker is down or the segment is offline) uses the following data set:

DBDLIB	The Pagestore definition. DBDLIB is not referenced when you use S6BBRULH to unload from an active TIBCO Object Service Broker system.

Task B Define and Submit Execution JCL

Define and submit JCL to run on offline or online segments, as shown in the following sections.

Execution When TIBCO Object Service Broker is Down or Segment is Offline

The S6BBRULB member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run a batch unload with TIBCO Object Service Broker shutdown or with the specified Pagestore segment offline.

This sample is provided as a reference only; modify the JCL for your needs.

Execution When TIBCO Object Service Broker is Active

The S6BBRULH member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run an unload with the TIBCO Object Service Broker system up and running.

This sample is provided as a reference only; modify the JCL for your needs.

Online/Offline Differences

There are several differences between the offline and online running modes:

- The utility name is S6BBRULB when running offline and S6BBRULH when running online.
- As parameters for S6BBRULH, you specify the Data Object Broker communications identifier and the pattern used for selecting the Execution Environment communications identifier. If required, you must also provide the user ID and password to obtain security clearance.
- For S6BBRULH, the DBDLIB DD statement is not required.
- For S6BBRULB, you must specify the segment where your table resides.

For compatibility with null handling techniques used in earlier TIBCO Object Service Broker releases, you can use the Execution Environment parameter setting OLDNULL=Y. For more information on this feature, refer to Appendix C, Null Handling, on page 151.

Task C Review the Audit Trail Report

The audit trail report provides information about the execution of the batch unload process. The audit trail for the S6BBRULB and the S6BBRULH processes are identical except for the module name, which appears at the left margin of the page header line.

The following contains a compressed version of a sample AUDIT TRAIL report:

```
S6BBRULB       BATCH - UNLOAD TABLE      DATE 2009 APR 20 TIME 10:48 V520E050


 UNLESS SPECIFIED BELOW (COLS 64-79) PACKED DECIMAL AND BINARY NULLS
 WILL BE CONVERTED AND OUTPUT TO THE FILE AS FOLLOWS:
 BINARY AS "LOWVALUE" ----> LENGTH 1-8  X'80' - X'8000000000000000'
 PACKED AS "LOWVALUE" ----> LENGTH 1-8  X'9B' - X'999999999999999B'


UNLOADING TABLE -  TC4048_DATA

            NAME         TYPE  SYNTAX  LENGTH  DECIMAL  OFFSET

FIELDS DEFINITIONS
     KEY                  I      V       20       0        0
     V1                          V        1       0       20
     V127                        V      127       0       21
     V254                        V      254       0      148
     IV1                  I      V        1       0      402
     IV127                I      V      127       0      403
     IV254                I      V      254       0      530
     SV1                  S      V        1       0      784
     SV127                S      V      127       0      785
     SV254                S      V      254       0      912
     CV1                  C      V        1       0     1166
     CV15                 C      V       15       0     1167
     QV1                  Q      V        1       0     1182
     QV15                 Q      V       15       0     1183

***** SEGMENT FORMAT IS TDS *****

S6BBU001I          5 DATA OCCURRENCES UNLOADED


      START TIME 10:48:49   END TIME 10:48:49
```

Constraints The following constraints apply:

- If you use either of the batch unload utilities to unload data, you can reload the data by using the batch load utility or by treating the data as a TIBCO Object Service Broker import file. However, you cannot treat the data as an import file when binary or packed fields are to contain numeric syntax null

values. Null values are preserved only by using the batch load and unload utilities. Refer to Appendix C, Null Handling, on page 151 for more information.

- You cannot reload the data using the interactive LOAD tool.
- If you are using FTP to transfer unloaded data between z/OS and Windows or Solaris, transfer the files in binary and specify the z/OS FTP LOCSITE subcommand with the RDW parameter.

Security on S6BBRULH

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.

Unload Requestor Identity

The Unload Requestor is identified by:

- The USERID entered in the U=*userid* parameter in the EXEC statement.

  If this parameter is supplied, the USERID is always be considered to be that of the requestor.
- The USERID (owner) of the job step (JOB statement USER=), which is assumed secured by z/OS security. This USERID is used only if no USERID is provided in the EXEC statement PARM.

If a USERID is supplied in the EXEC PARM statement and it differs from the job step's USERID, a TIBCO Object Service Broker password must be provided in the EXEC parameter P=*password*.

EXEC Statement Changes

Note the following changes to the JCL EXEC statement:

- The valid EXEC PARM format is as follows:

  ```
  PARM='ttttttmmmmmmmm'
  ```

  where:

ttttt	Mandatory ID (VTAM) of the Data Object Broker.
mmmmmmmm	Mandatory model ID for the job step.

Using this format, the owner of the job step is considered the requestor.

- An alternate EXEC PARM format can be used:

  ```
  PARM='T=tttttt,M=mmmmmm,U=uuuuuu,P=pppppp
  ```

  where:

ttttt	Mandatory communications identifier of the Data Object Broker.
mmmmmm	Mandatory communications model ID for the job step.
uuuuuu	Optional requestor's TIBCO Object Service Broker user ID.
pppppp	Optional requestor's TIBCO Object Service Broker password.

Do not use password if U=*uuuuuu* is not included.

If this format is used, it is not necessary to include U=*uuuuuu* if the job step user ID is also the TIBCO Object Service Broker user ID.

Error Situations

If the security check determines the request to be unauthorized, the utility writes the following error message to the audit report:

```
S6BBU139E ACCESS FOR UNLOAD DENIED
```

It then writes a corresponding message in the log and finally abends with a code of 202.

S6BSMEJA(B) (SMF Record Summary)

S6BSMEJA and S6BSMEJB are utilities that help in analyzing SMF file contents. The reports from these utilities provide a breakdown of SMF record types, subtypes, counts, and number of bytes of SMF data per job, task, and TSO user.

S6BSMEJB groups together all executions with the same job name.

Invocation The S6BSMEJA and S6BSMEJB members of the JCL data set distributed with TIBCO Object Service Broker contain sample JCL required to run these utilities. This sample is provided as a reference only; modify the JCL for your needs.

See Also *TIBCO Object Service Broker for z/OS Monitoring Performance* for complete information on the performance monitoring functions of these utilities.

Constraint To generate SMF records, a valid SMF record parameter must be in effect.

Sample Report The following illustrates a partial sample S6BSMEJB report:

```
S6BSMEJB      SMF RECORD SUMMARY - SYS1.SMFMAR31.Y2009
                                   2009APR01 03:47:05 -> 2009APR01 07:29:30
 ------------------------------------------------------------------------------

 NOT JOB RELATED
           2(........1)       3(........1)
 -SYSTEM- 2009APR01 073050 -> 2009APR01 073104          2 RECORDS         36 BYTES

 LAYCOCE  2009APR01 055700
      255H50(........1) 255H51(........1) 255H52(........1) 255H60(........1)
      255H61(........1) 255H62(........1) 255H72(.......31)
 LAYCOCE  2009APR01 072733 -> 2009APR01 072830         37 RECORDS      56842 BYTES

 MERRYW1  2009APR01 060502
      255H50(........1) 255H51(........1) 255H52(........1) 255H60(........1)
      255H61(........1) 255H62(........1)
 MERRYW1  2009APR01 060531 -> 2009APR01 072930          6 RECORDS       2002 BYTES

 S6ELDOBA 2009APR01 071942
      255H01(........1) 255H02(........1) 255H06(........1) 255H08(........1)
      255H09(........1) 255H10(........1) 255H11(........5) 255H12(........1)
      255H47(.......45) 255H49(.......45)
 S6ELDOBA 2009APR01 071947 -> 2009APR01 072907        102 RECORDS      17475 BYTES

 S6ELDOBB 2009APR01 071227
      255H01(........1) 255H02(........1) 255H06(........1) 255H08(........1)
      255H09(........1) 255H10(........1) 255H11(........5) 255H12(........1)
      255H13(........2) 255H22(........1) 255H23(........1) 255H47(........8)
      255H49(........8)
 S6ELDOBB 2009APR01 071231 -> 2009APR01 072903         32 RECORDS       5737 BYTES

 S6ELDOBC 2009APR01 071948
      255H01(........1) 255H02(........1) 255H06(........1) 255H08(........1)
      255H09(........1) 255H10(........1) 255H11(........6) 255H12(........1)
      255H47(.......45) 255H49(.......45)
 S6ELDOBC 2009APR01 071951 -> 2009APR01 072911        103 RECORDS      17644 BYTES
```

```
S6ELNEEA 2009APR01 072222
     255H50(........1) 255H51(........1) 255H52(........1) 255H60(........5)
     255H61(........5) 255H62(........5) 255H72(.......19)
S6ELNEEA 2009APR01 072224 -> 2009APR01 072854        37 RECORDS     29150 BYTES

S6ELNEEB 2009APR01 072226
     255H50(........1) 255H51(........1) 255H52(........1) 255H60(........5)
     255H61(........5) 255H62(........5)
S6ELNEEB 2009APR01 072228 -> 2009APR01 072845        18 RECORDS      7938 BYTES

S6ELNEEC 2009APR01 072230
     255H50(........1) 255H51(........1) 255H52(........1) 255H60(........5)
     255H61(........5) 255H62(........5) 255H72(.......19)
S6ELNEEC 2009APR01 072232 -> 2009APR01 072842        37 RECORDS     29150 BYTES

S6H1DOBB 2009MAR20 195523
     255H22(........8) 255H23(........8) 255H24(........2) 255H25(........2)
     255H26(........8) 255H47(........3) 255H49(........1)
S6H1DOBB 2009APR01 035505 -> 2009APR01 072930        32 RECORDS      7136 BYTES

S6H2DOBC 2009MAR20 200556
     255H22(........8) 255H24(........2) 255H25(........2) 255H26(........8)
S6H2DOBC 2009APR01 034806 -> 2009APR01 071806        20 RECORDS      5248 BYTES

S6H5DOBF 2009MAR20 201731
     255H22(........8) 255H23(........8) 255H24(........2) 255H25(........2)
     255H26(........8)
S6H5DOBF 2009APR01 034705 -> 2009APR01 071705        28 RECORDS      6448 BYTES
--------------------------------------------------------------------------------

TOTAL    2009APR01 034705 -> 2009APR01 072930       454 RECORDS    184806 BYTES
```

S6BSMESD (SMF Record Count Report)

S6BSMESD reads, truncates, and sorts data from an SMF record file and produces a report of record counts by system ID, date, and time.

Whenever S6BSMESD finds that two or more concurrent sorted internal records exactly match, there is a strong possibility that these represent duplicate SMF records. These get counted as part of the current system ID, date, and time range, and also as duplicates by record type.

Invocation

The S6BSMESD member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

- File //SORTIN is the file of SMF data records and must be in an appropriate variable length format: VBS, VB or V.
- File //REPORT has record format FBA, with record length 80.
- The PARM= value in the JCL is the default value of 60 minutes. With this setting, a line break is triggered in the report whenever the time difference between consecutive records is over 60 minutes. Changing the PARM value provides control over the number of gaps detected and hence the number of report lines.

Sample Report

```
S6BSMESD     SMF RECORD SUMMARY - SMF.DUMP.DAILY.G0088V00

 SYSID     DATE       START      END        REC.COUNT     BYTE.LENGTH
 ------------------------------------------------------------------
 FN00   2009MAR16 01:00:09  23:59:59         85,017      37,428,094
 FN00   2009MAR17 00:00:00  01:00:44         30,821      14,782,061
 ------------------------------------------------------------------
 SMF-2      25       SMF-3      25           115,888      52,211,055
 SIMILAR OR DUPLICATE                         30,315      14,780,594
 230..27916  50.....75  75.....69  72...2248  40......3 254......3 255. .....1
```

Interpreting the Report

This input file contains data from a 24 hour period split over two days. Neither SMF type 2 nor type 3 records are included in REC.COUNT; their counts are provided separately on the totals line.

Following the SMF-2 and SMF-3 counts are the total record and byte counts for the file (not including the SMF type 2 and 3 records).

This input file contains a number of records that are similar or duplicate. Their total count appears below the overall totals and forms a subset of the overall totals. A breakdown is provided in the form of record_type followed by record_count. In this example, there are only a few different record types, which indicates that these types are being produced in pairs or multiples with similar contents.

If there were a much larger number of different record types, it would indicate that certain time ranges are included twice or more within the file, making it unsuitable for accounting.

In cases where duplicated ranges are suspected and the input file is not re-sorted, S6BSMFCH helps determine the starting record number and counts of duplicated date-time ranges. S6BSMFCH produces a similar report, except that the data is reported in the sequence found in the file (no sort).

Return Codes

S6BSMESD can issue the following return codes:

Return Code	Meaning
0	Normal execution completed.
4	Parameter expression invalid (default is used).
8	OPEN failed - SORTIN or REPORT file missing or invalid.
>8	SORT program failed.

See Also

TIBCO Object Service Broker for z/OS Monitoring Performance for complete information about collecting and reporting on SMF records.

S6BSMETY (SMF Usage Analysis)

The S6BSMETY SMF utility helps you to analyze SMF file contents. The report from this utility tells you what types of SMF records you collected and how much space these records occupy. The report also provides details of individual SMF record subtypes and lists total byte counts and record percentages on a byte basis.

Invocation

The S6BSMETY member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Return Codes:

- 0 – Successfully completion.
- 8 – REPORT DD open error. Check JES job log for further information.
- 16 – SORT Program failure. This may be due to a null input file.

Sample Report

The following illustrates a sample report from S6BSMETY.

```
S6BSMETY       SMF RECORD SUMMARY - SYS1.SMFMAR31.Y2009
                                    2009APR01 03:47:05 -> 2009APR01 07:29:30

 REC - SUB    TYPE      COUNT   AVG.LEN   MIN.LEN   MAX.LEN    TOTAL BYTES    PERCENT
 ------------------------------------------------------------------------------------

  02                        1        18        18        18             18       .01 %

  03                        1        18        18        18             18       .01 %

 255 -  22    H301          8       212       212       212           1696       .92 %
 255 -  24    H301          2       112       112       112            224       .12 %
 255 -  25    H301          2       112       112       112            224       .12 %
 255 -  26    H301          8       388       388       388           3104      1.68 %
 255 -  22    H401          8       212       212       212           1696       .92 %
 255 -  23    H401          8        82        82        82            656       .35 %
 255 -  24    H401          2       112       112       112            224       .12 %
 255 -  25    H401          2       112       112       112            224       .12 %
 255 -  26    H401          8       456       456       456           3648      1.97 %
 255 -  50    H500          5       242       242       242           1210       .65 %
 255 -  51    H500          5       206       206       206           1030       .56 %
 255 -  52    H500          5      1703       498      4606           8514      4.61 %
 255 -  60    H500         17       252       252       252           4284      2.32 %
 255 -  61    H500         17       260       260       260           4420      2.39 %
 255 -  62    H500         17      1080       544      4596          18360      9.93 %
 255 -  72    H500         69      1265      1068      4616          87264     47.22 %
 255 -  01    H505          3        42        42        42            126       .07 %
 255 -  02    H505          3        42        42        42            126       .07 %
 255 -  06    H505          3       212       212       212            636       .34 %
 255 -  08    H505          3       112       112       112            336       .18 %
 255 -  09    H505          3       112       112       112            336       .18 %
```

```
255 -  10    H505          3       456       456       456          1368      .74 %
255 -  11    H505         16       169       169       169          2704     1.46 %
255 -  12    H505          3       174       174       174           522      .28 %
255 -  13    H505          2       342       226       458           684      .37 %
255 -  22    H505          9       212       212       212          1908     1.03 %
255 -  23    H505          9        82        82        82           738      .40 %
255 -  24    H505          2       112       112       112           224      .12 %
255 -  25    H505          2       112       112       112           224      .12 %
255 -  26    H505          8       456       456       456          3648     1.97 %
255 -  47    H505        101       172       172       172         17372     9.40 %
255 -  49    H505         99       173       172       184         17040     9.22 %
-------------------------------------------------------------------------------

TOTAL                    454       408        18      4616        184806   100.00 %
```

See Also *TIBCO Object Service Broker for z/OS Monitoring Performance* for detailed information about the use of this utility in performance monitoring.

S6BSMF12 (Checkpoint Statistics Report)

S6BSMF12 generates a Checkpoint Statistics report from TIBCO Object Service Broker SMF subtype 12 records specified in the //SMFIN JCL statement.

Invocation

The S6BSMF12 member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Notes on Sample JCL

- The input file contains TIBCO Object Service Broker SMF records extracted using S6BSMFEX. For more information, refer to S6BSMFEX (SMF Extract Reports).
- Each record of an appropriate subtype results in a report line (or in some cases multiple lines).
- The report line sequence is the same as the //SMFIN data sequence.
- The number of lines per report page is controlled by EXEC statement PARM=# (for example, PARM=60). The words `LINES PER PAGE` appear as a comment only.

  Specifying PARM=0, or a number less than header lines + 1, results in report lines without headings. This can be useful for reprocessing report information (with //REPORT to a data set, instead of SYSOUT).

Sample Report

The following illustrates a report from S6BSMF12 with an explanation of the columns:

```
S6BSMF12                                  CACHE/CHECKPOINT STATISTICS                                    2009APR02 13:35  PAGE    1
                                                                        TCB    SRB    I/O     ELAPSED MILLISECONDS
    DATE     TIME   DS CHPTID EXCP TRAN PAGES= DATA+IDX+BIT+OTR        MSEC   MSEC    BLK CHKPT CCHIO  JLIO PRPGA     DATA OBJECT BROKER
-------------------------------------------------------------------------------------------------------------------------------------
2009APR01 072900  2    024    1    13     5     2   1   1   1        531     76    553    21     8     8    10 S6ELDOBB 2009APR01 0712

2009APR01 072904  1    029    1    11     4     2   1   1   0        647      2    534    23     9     9     5 S6ELDOBA 2009APR01 0719

2009APR01 072907  2    022    1    12     6     3   1   1   1        644      4    557    19     7     2     2 S6ELDOBC 2009APR01 0719
```

Columns:

DATE	The date the checkpoint was taken (HU12DTE)
TIME	The time of day the checkpoint was taken (HU12TME)
DS	The cache data set number (HU12CID)

CHPTID	The checkpoint sequence number (HU12XSEQ)
EXCP	The number of EXCPs issued (HU12EXCP)
TRAN	The number of transactions processed by the checkpoint (HU12NTRX)
PAGES	The total number of pages cached (HU12PAG)
DATA	The number of data pages cached (HU12DAT)
IDX	The number of index pages cached (HU12IDX)
BIT	The number of bitmap pages cached (HU12BIT)
OTR	The number of other types of pages cached (HU12OTHR)
TCB MSEC	The total TCB time since the last checkpoint, in milliseconds (HU12TCBT)
SRB MSEC	The total SRB time since the last checkpoint, in milliseconds (HU12SRBT)
I/O BLK	The total I/O count since the last checkpoint (HU12IO)
CHKPT	The total checkpoint time, in milliseconds (HU12TIM6-HU12TIM2)
CCHIO	The cache write time, in milliseconds (HU12TIM3-HU12TIM2)
JLIO	The journal write time, in milliseconds (HU12TIM4-HU12TIM3)
PRPGA	The Pagestore write time, in milliseconds (HU12TIM5-HU12TIM3)
DATA OBJECT BROKER	The Data Object Broker jobname (HU12JBN) The Data Object Broker reader start date in YYYYMMMDD format (HU12RSD) The Data Object Broker reader start time, in HHMM format (HU12RST)

See Also *TIBCO Object Service Broker for z/OS Monitoring Performance* for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSMF13 (Pagestore Response Time Report)

S6BSMF13 generates a report on Pagestore response times during checkpoint cycles.

Invocation

The S6BSMF13 member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Notes on Sample JCL

- The input file contains TIBCO Object Service Broker SMF records extracted using S6BSMFEX. For more information, refer to S6BSMFEX (SMF Extract Reports).
- Data should be pre-sorted to ensure proper report ordering.
- The PAGESIZE parameter is optional. It is used to control the number of lines printed per page.
- The SEGMENT parameter is optional. If specified, it produces a report showing the response times by data set for the segment. You can specify either the segment number or the segment name.
- The DATASET parameter is optional and can be specified only if the SEGMENT parameter is also specified. Specifying this parameter produces a report on the response times for a specific data set.
- Sorting input records into a sequence other than this sequence can cause improper groupings of data.

Sample Report

You can produce three types of reports from S6BSMF13:

- A general report (providing no SEGMENT or DATASET parameters)
- A segment report (providing only a SEGMENT parameter)
- A data set report (providing both SEGMENT and DATASET parameters)

The following illustrates a sample report from S6BSMF13.

```
S6BSMF13                                PAGESTORE RESPONSE TIME STATISTICS                    2009APR02 11:04  PAGE    1
                                              SMFDATE=2009APR01

-------------------------------------------------------------------------------------------------------------------------

 START   END    SEG    READS     <6  6-10 11-15 16-20 21-25 26-30   >30       WRITES     <6  6-10 11-15 16-20 21-25 26-30   >30
 -----   -----  ---    -----  ---------------------------------------------    ------  ---------------------------------------------
 07:12   07:27    0       21      9    11     0     1     0     0     0           0      0     0     0     0     0     0     0
                  1        2      0     2     0     0     0     0     0           0      0     0     0     0     0     0     0
```

See Also *TIBCO Object Service Broker for z/OS Monitoring Performance* for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSMF22 (Lock Manager Statistics Report)

S6BSMF22 generates a Lock Manager Statistics report from all TIBCO Object Service Broker SMF records supplied via the statement //SMFIN.

Invocation

The S6BSMF22 member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The input file contains TIBCO Object Service Broker SMF records extracted using S6BSMFEX. For more information, refer to S6BSMFEX (SMF Extract Reports).

Sample Report

The following illustrates the S6BSMF22 report:

```
S6BSMF22                                  L O C K    M A N A G E R    S T A T I S T I C S                2009APR02 11:11  PAGE   1
            REQUESTS      G E T L O C K   AUTO-UPG   TRANSACT   RESOURCE                    MILLISEC      LOCK BUFFERS
  INTERVAL   GETLOCK   GRANTED   UPG-REQ              COUNT      COUNT                        CLOCK      NUMBER  X LENGTH   CONTROL
  END TIME  FREELOCK  ALREADY   BLOCKED  ANCESTOR    MAX_CON    MAX_CON                     LM-CPU    MAX_USE       GETS   REGION
  DURATION   TIMEOUT  NO_NEED  DEADLOCK  CUT_STRG    CURRENT    CURRENT                     CREG %    CUR_USE      FREES
-------------------------------------------------------------------------------------------------------------------------------
2009APR01E        63        53        10                 19         53                          1        384  X    4096    S6ELDOBA
 07:29:04O        19         0         0          10      1         17                          1          7          43   2009APR01
       557J       16         0         0           0      0          0                     .50 %          5          38    07:19:42

2009APR01          3         3         0                  1          3                          0        384  X    4096    S6ELDOBB
 07:27:03          0         0         0           0      1          3                          0          7           7   2009APR01
       872         2         0         0           0      1          1                                     7           0    07:12:27

2009APR01E       130       115        12                 23         78                          3        384  X    4096    S6ELDOBB
 07:29:00O        22         0         0           9      3         60                          2          9          29   2009APR01
       989J       23         3         0           0      1          3                     .58 %          7          22    07:12:27

2009APR01E        81        71        10                 20         30                          1        384  X    4096    S6ELDOBC
 07:29:08O        19         0         0          11      2         20                          1          8          26   2009APR01
       556J       27         0         0           0      1          6                     .52 %          7          19    07:19:48
```

See Also

TIBCO Object Service Broker for z/OS Monitoring Performance for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSMF23 (Query Task Usage Report)

S6BSMF23 generates multiple query task usage statistics, generated whenever an end of SMF interval is detected.

Invocation

The S6BSMF23 member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Sample Report

The following illustrates the S6BSMF23 report:

```
HRNSMF23                          MULTIPLE QUERY TASK UTILIZATION STATISTICS                    2009JUL02 16:14:04   PAGE      1
JOBNAME      RECORD TIME   TCB  TIME USED    EVENTS MESSAGES  MAX-CON    WAITS NO WAITS   QUEUED
HRUNPRD 2009JUL01 04:39:31 1        0.00         0        0        0         0         0        0
                           2        0.00         0        0        0         0         0        0
        2009JUL01 04:54:31 1        0.00         0        0        0         0         0        0
                           2        0.00         0        0        0         0         0        0
        2009JUL01 05:09:31 1        2.73      2514     2514        1      2514         0        0
                           2        0.00         0        0        0         0         0        0
      900         0         0         0         0          0   29954485                                    0         0   14:54:20
```

See Also

TIBCO Object Service Broker for z/OS Monitoring Performance for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSMF24 (Query/Commit Response Time Report)

S6BSMF24 generates a Data Object Broker Query/Commit Response Time report from all TIBCO Object Service Broker SMF records supplied via the statement //SMFIN.

Invocation

The S6BSMF24 member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Notes on Sample JCL

- The input file contains TIBCO Object Service Broker SMF records extracted using S6BSMFEX. For more information, refer to S6BSMFEX (SMF Extract Reports).
- The PARM= parameter controls the number of report lines per page, and also suppresses report headings if set to less than 6.

Sample Report

The following is a sample report:

```
S6BSMF24                                QUERY/COMMIT RESPONSE TIME STATISTICS                                2009APR02 11:18  PAGE    1
                      QUERY COUNTS BY RESPONSE TIME (MS)             COMMIT COUNTS BY RESPONSE TIME (MS)
 INTERVAL END      ...2....4....8...16...32...64..128....> ...2....4....8...16...32...64..128....>              DATA OBJECT BROKER
 ------------------------------------------------------------------------------------------------------------------------------------

 2009APR01 0729T       63                                        60                                          S6ELDOBA 2009APR01 0719

 2009APR01 0729T      535                                        51                                          S6ELDOBB 2009APR01 0712

 2009APR01 0729T      130                                        58                                          S6ELDOBC 2009APR01 0719
```

Considerations

Lines showing an interval end time represent subtype 24 records.

RUN TOTALS lines are produced for subtype 8 records.

See Also

TIBCO Object Service Broker for z/OS Monitoring Performance for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSMF25 (Send/Receive Message Length Report)

S6BSMF25 generates a Data Object Broker Send/Receive Message Length report from all TIBCO Object Service Broker SMF subtype 25 and 09 records supplied via the //SMFIN statement.

Invocation

The S6BSMF25 member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The input file contains TIBCO Object Service Broker SMF records extracted using S6BSMFEX. For more information, refer to S6BSMFEX (SMF Extract Reports).

Sample Report

The following is a sample report:

```
S6BSMF25                                SEND/RECEIVE MESSAGE LENGTH STATISTICS                          2009APR02 11:25  PAGE   1
                    RECEIVE COUNTS BY MESSAGE LENGTH                  SEND COUNTS BY MESSAGE LENGTH
INTERVAL END     ..64..128..256..512...1K...2K...4K....> ..64..128..256..512...1K...2K...4K....>          DATA OBJECT BROKER
-------------------------------------------------------------------------------------------------------------------------------

2009APR01 0729T   130   61    2    8    5   15           136   37         9   24   15                    S6ELDOBA 2009APR01 0719

2009APR01 0729T   237  652  108   15    9   32           533   85   59  143  130   67   34    2          S6ELDOBB 2009APR01 0712

2009APR01 0729T   193   64    3    8    4   17           201   38         9   24   17                    S6ELDOBC 2009APR01 0719
```

Lines showing an interval end time represent subtype 25 records.

RUN TOTALS lines are produced for TIBCO Object Service Broker subtype 9 records.

See Also

TIBCO Object Service Broker for z/OS Monitoring Performance for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSMF26 (Data Object Broker General Statistics Report)

S6BSMF26 generates a Data Object Broker General Statistics report from all TIBCO Object Service Broker SMF subtype 26 and 10 records supplied via the statement //SMFIN.

Invocation

The S6BSMF26 member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The input file contains TIBCO Object Service Broker SMF records extracted using S6BSMFEX. For more information, refer to S6BSMFEX (SMF Extract Reports).

The PARM= parameter controls the number of lines per page. It also suppresses if it is set to a value that is less than 6.

Sample Report

The following is a partial sample report:

```
S6BSMF26                                  G E N E R A L    S T A T I S T I C S                                  2011JAN06 08:40  PAGE   1
                MESSAGES     . . . DESTINATION . . .  . . . . . . LOGICAL . . . . . . . . . . . PHYSICAL . . . . . .
INTERVAL    DUR SEND  GETB   CR100 CR300   PH1  DBMS    GET GETF LOCK REPL RULE READ  JRNL REPL RULE TCBMS SRBEN   DATA OBJECT BROKER
END TIME    SEC RECV  FREB   CR200 CR400   PH2  OPER    PUT PUTF UNLK  INS CTAB WRITE EXCP  INS CTAB SRBMS SRBZI
------------ ---------  -------------PH0------   ----------------DEL----- ------------DEL--------I/O------   ------------------
2011JAN06   276   1M   7M      0   15K    1M     0     5M  963 705K   15  85K 7733     0 3607  297   12K  374K   DC52ADBA 2011JAN06
 07:54:31         1M   7M   1305    1M  5732    45   356K 2593 2248  72K   1M    0     0 7055  41K   17K  372K            07:49:54
                                                                    212K                 6524         11K
2011JAN06   300   2M   8M      0  8348    2M     0     6M 1521 782K   17 122K  540     0 8058  427   10K  306K   DC52ADBA 2011JAN06
 07:59:31         2M   8M   1695    2M  7167     0   310K 1121 3059 115K   2M    0     0  13K  71K   22K  304K            07:49:54
                                                                     85K                 2563         11K
2011JAN06   300 789K   3M      3  8216  781K     0     2M 1239 333K   8  81K 1424   734  12K 1141  7087  120K   DC52ADBA 2011JAN06
 08:04:32       789K   3M    642   784K  6564    66   224K  378 1077  98K 908K  734    29  16K  41K  7091  120K            07:49:54
                                                                     29K                  819         9240
2011JAN06   300 453K   2M      0  5881  448K     0     1M  865 199K   4  52K 1078     0 8911  883  6264   70K   DC52ADBA 2011JAN06
 08:09:32       453K   2M    379   450K  4702    58   153K  189  580  69K 532K    0     0  11K  23K  4220   70K            07:49:54
                                                                     14K                  402         6323
2011JAN06   RUN   4M   21M      6   39K    4M     0    16M 4748   2M   45 349K  11K 2436  35K 2967   38K  872K   DC52ADBA 2011JAN06
 08:11:20 TOTAL   4M   21M   4027    4M   25K   173     1M 4281 6973 368K   5M 2436    99  50K 177K   51K  868K            07:49:54
                                                                    340K                  10K          40K
```

Lines showing a numeric DUR SEC (interval duration in seconds) represent subtype 26 records.

RUN TOTAL lines are produced for TIBCO Object Service Broker subtype 10 records.

See Also

TIBCO Object Service Broker for z/OS Monitoring Performance for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSMF49 (User Consumption of Data Object Broker Services Report)

S6BSMF49 generates an End User Statistics report from all TIBCO Object Service Broker SMF subtype 49 and 47 records supplied via the //SMFIN statement.

Invocation

The S6BSMF49 member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Notes on Sample JCL

- The input file contains TIBCO Object Service Broker SMF records extracted using S6BSMFEX. For more information, refer to S6BSMFEX (SMF Extract Reports).
- Each record of the appropriate subtype results in a report line (or in some cases multiple lines).
- The report line sequence is the same as the //SMFIN data sequence.
- The number of lines per report page are controlled by EXEC statement PARM=## (for example, PARM=60). The words `LINES PER PAGE` appear as a comment only. Specifying PARM=0, or a number less than header lines + 1, results in report lines without headings. This can be useful for re-processing report information (with //REPORT to a data set, instead of SYSOUT).

Sample Report

Using the S6BSMF49 utility, you can generate an End User Statistics report from all the TIBCO Object Service Broker SMF subtype 49 and 47 records supplied via the statement //SMFIN. In this sample, the //SMFIN data is temporarily re-sorted by USERID and Logoff time.

```
S6BSMF49                                USER CONSUMPTION OF DATA OBJECT BROKER SERVICES                                2011JAN06 09:24  PAGE   1
                      RECORD      TYPE #   COMM  #QRY GETS    C.R. READS   QUERYCPU   ZIIPCPU        #SRVR  SERVER  DATA OBJECTBROKER
USERID   LOGON        TIME        ELAPSE   MODE  #CMT PUTS    RESP WRITES   FILECPU   SRBTIME  DBMS  CALLS  RESP       BATCH JOB STEP
-------------------------------------------------------------------------------------------------------------------------------------
SYSADMIN 2011JAN06  2011JAN06  D   01   XMEM   115  301   .261    84      .019      .019                              DC52ADBA 2011JAN06 07:49
          07:50:31   07:50:32      01   JOB     03   06          14      .008      .011                                   DCBATRE1  BATCH1
SYSADMIN 2011JAN06  2011JAN06  LOGOFF XMEM   115  301   .261    84      .019      .019                              DC52ADBA 2011JAN06 07:49
          07:50:31   07:50:32      01   JOB     03   06          14      .008      .011                                   DCBATRE1  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   01   XMEM  266K 995K  .197  1941   125.865  125.494                              DC52ADBA 2011JAN06 07:49
          07:50:31   07:54:31     240   JOB   2018 175K        5552     4.539    3.024                                   DCBATREF  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   02   XMEM   34K 127K  .268   540     5.091    5.042                              DC52ADBA 2011JAN06 07:49
          07:50:31   07:59:31     300   JOB   1686  48K        1598     1.490     .442                                   DCBATREF  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   03   XMEM   95K 352K  .224  1424    13.648   13.614                              DC52ADBA 2011JAN06 07:49
          07:50:31   08:04:32     300   JOB   4675 132K        5190     4.528    1.249                                   DCBATREF  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   04   XMEM   77K 282K  .250  1078    11.478   11.459                              DC52ADBA 2011JAN06 07:49
          07:50:31   08:09:32     300   JOB   3760 107K        4194     3.609    1.004                                   DCBATREF  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   05   XMEM   18K  66K  .292   273     2.965    2.963                              DC52ADBA 2011JAN06 07:49
          07:50:31   08:11:01      89   JOB    929  26K        1029      .891     .238                                   DCBATREF  BATCH1
SYSADMIN 2011JAN06  2011JAN06  LOGOFF XMEM   490K  1M   .220  5256   159.047  158.572                              DC52ADBA 2011JAN06 07:49
          07:50:31   08:11:01    1230   JOB    13K 488K         18K    15.057    5.957                                   DCBATREF  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   01   XMEM  435K  1M   .151  1888    78.605   78.096                              DC52ADBA 2011JAN06 07:49
          07:50:32   07:54:31     239   JOB   1194  57K        1817     2.244    4.663                                   DCBATRE2  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   02   XMEM  614K  2M   .128    00    91.739   91.166                              DC52ADBA 2011JAN06 07:49
          07:50:32   07:59:31     300   JOB   1954  95K        3203     2.232    6.514                                   DCBATRE2  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   03   XMEM   61K 201K  .127    00     9.134    9.092                              DC52ADBA 2011JAN06 07:49
          07:50:32   07:59:55      24   JOB    174 5028         169      .118     .639                                   DCBATRE2  BATCH1
SYSADMIN 2011JAN06  2011JAN06  LOGOFF XMEM    1M   3M   .137  1888   179.478  178.354                              DC52ADBA 2011JAN06 07:49
          07:50:32   07:59:55     563   JOB   3322 158K        5189     4.594   11.816                                   DCBATRE2  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   01   XMEM  442K  1M   .150  1786    83.238   82.699                              DC52ADBA 2011JAN06 07:49
          07:50:33   07:54:31     238   JOB   1254  59K        1880     2.221    4.730                                   DCBATRE3  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   02   XMEM  656K  2M   .131    00    99.499   98.887                              DC52ADBA 2011JAN06 07:49
          07:50:33   07:59:31     300   JOB   1768  81K        2717     1.893    6.940                                   DCBATRE3  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   03   XMEM   486 1713  .255    00      .068     .068                              DC52ADBA 2011JAN06 07:49
          07:50:33   07:59:34      03   JOB     33  780         26      .018     .006                                   DCBATRE3  BATCH1
SYSADMIN 2011JAN06  2011JAN06  LOGOFF XMEM    1M   3M   .139  1786   182.805  181.654                              DC52ADBA 2011JAN06 07:49
          07:50:33   07:59:34     541   JOB   3055 141K        4623     4.132   11.676                                   DCBATRE3  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   01   XMEM  465K  1M   .150  2034    85.834   85.275                              DC52ADBA 2011JAN06 07:49
          07:50:35   07:54:31     236   JOB   1264  65K        2073     2.488    4.988                                   DCBATRE4  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   02   XMEM  737K  2M   .125    00   109.191  108.528                              DC52ADBA 2011JAN06 07:49
          07:50:35   07:59:31     300   JOB   1758  86K        2897     2.018    7.784                                   DCBATRE4  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   03   XMEM  625K  2M   .212    00    97.191   97.035                              DC52ADBA 2011JAN06 07:49
          07:50:35   08:04:32     300   JOB   1682  85K        3348     2.292    6.632                                   DCBATRE4  BATCH1
SYSADMIN 2011JAN06  2011JAN06  D   04   XMEM  372K  1M   .214    00    58.255   58.193                              DC52ADBA 2011JAN06 07:49
          07:50:35   08:07:54     203   JOB    942  46K        1801     1.233    3.936                                   DCBATRE4  BATCH1
SYSADMIN 2011JAN06  2011JAN06  LOGOFF XMEM    2M   7M   .170  2034   350.471  349.031                              DC52ADBA 2011JAN06 07:49
          07:50:35   08:07:54    1040   JOB   5646 283K         10K     8.031   23.340                                   DCBATRE4  BATCH1
```

Notes

- Lines showing TYPE of LOGOFF represent subtype 49 logoff records.
- Lines showing TYPE of I ## are from subtype 47 interval records at interval ## since logon.

See Also *TIBCO Object Service Broker for z/OS Monitoring Performance* for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSMFAK (Add Key to SMF Records)

S6BSMFAK modifies TIBCO Object Service Broker Data Object Broker SMF records to relocate or copy their job identification to the standard offset.

Use S6BSMFAK to modify SMF subtype records (for example, SMF30) that do not have their job identification at the standard offset used by original-format job records.

S6BSMFAK modifies SMF records where necessary so that their job identification appears at the standard offset. Job identification within most SMF records is by Jobname and Reader-Start Date and Time. Non-job records receive a dummy job identification of Z9999999 to make them sort last. When job identification is inserted or relocated, this is indicated by setting bits in the first byte of **SMF##TME**. The maximum correct value for **SMF##TME** is X'0083D5FF'.

After modifying these SMF records, you can sort them by job identification and time. Do not use the first byte of **SMF##TME** for sorting.

After sorting, use S6BSMFDK to restore the records modified by S6BSMFAK to their original layout. For more information about S6BSMFDK, refer to S6BSMFDK (Remove Key Added by S6BSMFAK).

Invocation

The S6BSMFAK member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run S6BSMFAK, SORT, and S6BSMFDK. This sample is provided as a reference only; modify the JCL for your needs.

Files //SMFIN, //SORTIN, //SORTOUT, and //SMFOUT can be in any variable length format: VBS, VB, or V.

This job runs in the following order:

- S6BSMFAK copies (modifies) records from //SMFIN to //SORTIN.
- SORT FIELDS sorts the data by job identification, and by date and time within job identification. Because S6BSMFAK uses bits from the first byte of the **SMF##TME** field to indicate job-ID key insertion, you should SORT on 3 bytes at **SMF##TME**+1 (last sort field).
- S6BSMFDK copies (restores) records from //SORTOUT to //SMFOUT.

Return Codes S6BSMFAK can issue the following return codes:

Return Code	Meaning
0	Normal execution.
16	Open Error (missing DD) on file //SMFIN or //SORTIN.

S6BSMFCH (SMF Check Report)

S6BSMFCH reports on the Date/Time/System ID ranges present in an SMF file, providing record counts. Using these reported counts, you can determine out-of-sequence, duplicate, and missing time ranges.

Use S6BSMFCH to check an SMF file. This file is normally in ascending date/time sequence. S6BSMFCH scans an SMF file and produces a report of record counts by date, time, and system ID.

S6BSMFCH produces a report line, with starting record and count, when any of the following occurs:

1. The system ID changes
2. The date changes
3. The time decrease between records is more than 1 minute
4. The time increase between records is more than GAP LIMIT (from PARM=number-of-minutes or default value 60 minutes)
5. Bracketing SMF 2 dump header or SMF 3 dump trailer is encountered

Invocation

The S6BSMFCH member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.

This sample is provided as a reference only; modify the JCL for your needs.

Notes on Sample JCL

- File //SMFIN is a sequential file in any variable length format: VBS, VB, or V.
- File //REPORT has record format FBA, with record length 80.
- The PARM=value shown is the default of 60 minutes. Changing this value controls the number of "gaps" detected (number of report lines).

Sample Report

The following illustrates a sample report produced using S6BSMFCH:

```
      SYS1.SMF.STAR.EXTRACT
      SYSID    DATE        START       END     1ST.RECORD    REC.COUNT
      ------------------------------------------------------------------
(     STAR   2009MAR19  00:36:32  12:33:03            2         9,891   )
(     STAR   2009MAR19  12:33:04  23:59:47        9,895        10,611   )
(     STAR   2009MAR20  00:00:01  00:32:35       20,506         1,164   )
(     STAR   2009MAR18  12:37:16  14:25:19       21,672        22,963   )
(     STAR   2009MAR18  14:25:19  16:09:24       44,637        26,163   )
(     STAR   2009MAR18  16:09:24  17:25:11       70,802        28,153   )
(     STAR   2009MAR18  17:25:11  17:39:21       98,957         8,483   )
(     STAR   2009MAR18  09:43:54  12:08:05      107,442        22,168   )
(     STAR   2009MAR18  00:30:52  09:43:54      129,612        26,925   )
(     STAR   2009MAR18  12:08:05  12:37:16      156,539         4,487   )
```

```
(    STAR    2009MAR18  17:39:21  17:57:54     161,028      13,121    )
(    STAR    2009MAR18  17:57:54  18:59:42     174,151      13,484    )
     ------------------------------------------------------------------
                                                              187,615
```

From this report, you can see that:

1. The data is out of date/time sequence.
2. There were frequent SMF switches on March 18th.
3. The end of the 18th is missing.

An open bracket to the left of SYSID indicates an SMF type 2 Dump Header record precedes the range. A closed bracket after REC.COUNT indicates an SMF type 3 Dump Trailer follows the range. The 1ST.RECORD is the relative record not counting the preceding SMF 2. Neither SMF type 2 nor type 3 records are included in REC.COUNT (except in the final total).

Return Codes

S6BSMFCH can issue the following return codes:

Return Code	Meaning
0	Normal execution.
4	Parameter expression invalid.
16	Open Error (missing DD) on file //SMFIN or //REPORT.
20	Both return codes 16 and 4.

S6BSMFDK (Remove Key Added by S6BSMFAK)

Use S6BSMFDK to restore records modified by S6BSMFAK to their original layout.

After sorting SMF records using S6BSMFAK, use S6BSMFDK to restore the modified records to their original layout. For more information about S6BSMFAK, refer to S6BSMFAK (Add Key to SMF Records).

Invocation

The S6BSMFAK member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run S6BSMFAK, SORT, and S6BSMFDK. This sample is provided as a reference only; modify the JCL for your needs.

Files //SMFIN, //SORTIN, //SORTOUT, and //SMFOUT can be in any variable length format: VBS, VB, or V.

This job runs in the following order:

1. S6BSMFAK copies (modifies) records from //SMFIN to //SORTIN.
2. SORT FIELDS sorts the data by job identification, and by date and time within job identification. Because HRMSMFAK uses bits from the first byte of the **SMF##TME** field to indicate job-ID key insertion, you should SORT on 3 bytes at **SMF##TME**+1 (last sort field).
3. S6BSMFDK copies (restores) records from //SORTOUT to //SMFOUT.

Return Codes

S6BSMFDK can issue the following return codes:

Return Code	Meaning
0	Normal execution.
16	Open Error (missing DD) on file //SORTOUT or //SMFOUT.

S6BSMFEX (SMF Extract Reports)

S6BSMFEX extracts all SMF records generated by TIBCO Object Service Broker or specific records for jobs whose names are specified in the PARM statement. Particular record types can be specifically excluded from the extract.

SMF records can be extracted from either tape or sequential disk file. Although you can run the TIBCO Object Service Broker SMF reporting programs using SMF archive tapes, it is often preferable to extract the required subset of SMF records into a sequential disk file. SMF records can then be sorted and processed more rapidly from this file than from tape.

Utilities for Extracting TIBCO Object Service Broker SMF Records

You can use one of two utilities to extract TIBCO Object Service Broker SMF records:

- S6BSMFEX, the TIBCO Object Service Broker utility
- IFASMFDP, the z/OS utility

S6BSMFEX is recommended over IFASMFDP in most instances because of its capabilities specific to TIBCO Object Service Broker and because it always treats its sequential input file as read-only. S6BSMFEX, however, cannot extract data from VSAM SMF collection data sets. For more information about using IFASMFDP, refer to *TIBCO Object Service Broker for z/OS Monitoring Performance*.

Using S6BSMFEX

S6BSMFEX can be used in either of two ways:

- SMF records can be selected by one or more job names specified in the PARM string.
- All TIBCO Object Service Broker SMF and SMF 30-4 records for the Data Object Broker program S6BCR000 are selected (no job name specified).

In either case, unwanted SMF record types can be excluded.

Invocation

The S6BSMFEX member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Notes on Sample JCL

- File //SMFIN is a sequential tape or disk input file in any variable length format: VBS, VB, or V. VSAM is not supported.

- File //SMFOUT is a sequential disk or tape output file in any variable-length format (VBS, VB, or V). Provide a DCB that can refer back to the input, as shown.
- File //REPORT is record format FBA with record length 80. It can be omitted but then there is no explanatory report.

Specifying Parameters

S6BSMFEX parameters are specified through the EXEC PARM= *expression* statement. In this sample JCL, records for jobs S6H0DOBA and S6H1DOB are to be selected but SMF type 6 and 230 records are to be excluded.

Considerations

- You can specify up to ten job names and exclude any number of SMF record types, to a maximum parameter string length of 100 characters.
- Generic job names can be specified, for example, S6H* would select S6H0DOBA, S6H0DOBB, and other job names beginning with S6H.
- Record types can be excluded by specifying the decimal record number preceded by a minus sign (-).
- Parameters are separated by commas, blanks, or both.

Sample Report

The following illustrates a sample report from S6BSMFEX:

```
S6BSMFEX              SMF JOB RECORD SELECTION              2009APR01 03:47:05 -> 2009APR01 07:29:30
PARM='S6ELDOBC,S6ELDOBA,-6,-230'
       454   INPUT RECORDS      SYS1.SMFMAR31.Y2009
       205  OUTPUT RECORDS      SYS1.TNV1.OSBT52.SMFDATA.EXTRACT
       102  OUTPUT RECORDS FOR JOBNAME  S6ELDOBA
       103  OUTPUT RECORDS FOR JOBNAME  S6ELDOBC
```

Return Codes

S6BSMFEX can issue the following return codes:

Return Code	Meaning
0	Normal execution.
4	Error in parameter expression, execution continued.
8	No eligible records found in input.
12	Both return code 8 and 4.
16	Open error (missing DD) on file //SMFIN or //SMFOUT.
20	Both return codes 16 and 4.

S6BSMFQT (Query Task CPU Usage Analysis)

S6BSMFQT generates a Data Object Broker Query Task Analysis report from all TIBCO Object Service Broker SMF subtype 26 records supplied via the statement //SMFIN.

Data Object Broker Requirement

The Data Object Broker being monitored must have SMF subtype 26 interval processing enabled. We recommend that you set this interval to a value of 15 minutes or less, for example, SMF26INTERVAL=15.

Invocation The S6BSMFQT member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Sample Report A sample report is shown here:

```
HRNSMFQT                                  Q U E R Y   T A S K   A N A L Y S I S                    2009JUL09 13:55 PAGE   1
JOBNAME        INTERVAL          INTERVAL       QUERY TASK      QUERY TASK      PERCENTAGE
                  END            DURATION      TRANSACTIONS      CPU TIME          BUSY
              DATE      TIME     MILLISECONDS                    MILLISECONDS
HRUNPRD   2009JUL06 00:54:31        7M                0               0               0
HRUNPRD   2009JUL07 00:54:33        7M                0               0               0
HRUNPRD   2009JUL08 00:54:36        7M                0               0               0
HRUNPRD   2009JUL06 02:54:31        7M                0               0               0
HRUNPRD   2009JUL07 02:54:33        7M                0               0               0
HRUNPRD   2009JUL08 02:54:36        7M                0               0               0
HRUNPRD   2009JUL06 04:54:31        7M                0               0               0
HRUNPRD   2009JUL07 04:54:33        7M                0               0               0
HRUNPRD   2009JUL08 04:54:36        7M                0               0               0
HRUNPRD   2009JUL06 06:54:31        7M             2514            2739               0
HRUNPRD   2009JUL07 06:54:34        7M             2516            2744               0
HRUNPRD   2009JUL08 06:54:36        7M             2514            2720               0
HRUNPRD   2009JUL06 08:54:32        7M                0               0               0
HRUNPRD   2009JUL07 08:54:34        7M                0               0               0
HRUNPRD   2009JUL08 08:54:36        7M                0               0               0
HRUNPRD   2009JUL06 10:54:32        7M                0               0               0
HRUNPRD   2009JUL07 10:54:34        7M                0               0               0
HRUNPRD   2009JUL08 10:54:36        7M                0               0               0
HRUNPRD   2009JUL06 12:54:32        7M                0               0               0
HRUNPRD   2009JUL07 12:54:34        7M                0               0               0
HRUNPRD   2009JUL08 12:54:36        7M                0               0               0
HRUNPRD   2009JUL06 14:54:32        7M                0               0               0
HRUNPRD   2009JUL07 14:54:35        7M                0               0               0
HRUNPRD   2009JUL05 16:54:30        7M                0               0               0
HRUNPRD   2009JUL06 16:54:32        7M              13K            9961               0
HRUNPRD   2009JUL07 16:54:35        7M                0               0               0
HRUNPRD   2009JUL05 18:54:30        7M                0               0               0
HRUNPRD   2009JUL06 18:54:33        7M                0               0               0
HRUNPRD   2009JUL07 18:54:35        7M                0               0               0
HRUNPRD   2009JUL05 20:54:30        7M                0               0               0
HRUNPRD   2009JUL06 20:54:33        7M                0               0               0
HRUNPRD   2009JUL07 20:54:35        7M                0               0               0
HRUNPRD   2009JUL05 22:54:31        7M                0               0               0
HRUNPRD   2009JUL06 22:54:33        7M                0               0               0
HRUNPRD   2009JUL07 22:54:35        7M                0               0               0
```

See Also *TIBCO Object Service Broker for z/OS Monitoring Performance* for more information about reporting on TIBCO Object Service Broker SMF records, including field information.

S6BSPCAP (Determine VSAM DS Capacity)

S6BSPCAP is a general purpose utility designed to assist you with processing VSAM data sets. It evaluates the capacity of a VSAM data set and returns whether the data set is using secondary extents. You can use S6BSPCAP to determine if a journal accumulation file exceeds its primary extents allocation.

Invocation

The S6BSPCAP member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The data set identified in the //MASTER DD statement can be any VSAM file.

Constraint

If DISP=OLD, the S6BSPCAP utility cannot be used against a VSAM data set that is already allocated to another user. If DISP=SHR and SHAREOPTIONS are enabled, the file allows multiple user accesses and no constraints exist

Return Codes

The return codes from S6BSPCAP indicate the following:

Return Code	Meaning
0	The VSAM data set is not empty and does not exceed its primary extents.
4	The VSAM data set is empty.
8	The VSAM data set exceeds its primary extents.

S6BSPDSN (Determine Number of GDGs)

S6BSPDSN is a general purpose utility that assists you with TIBCO Object Service Broker journal processing. Given the name of a generation data group (GDG), S6BSPDSN returns the number of generations that exist.

You can use S6BSPDSN to determine how many journal backups were written to disk and offload them to tape depending upon the return code.

Invocation

The member SPIN01 of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The sample SPIN01 JCL uses the GDG= parameter to determine the number of generations that exist for the file. For example, the following JCL will set the step return code to the number of generations of $HLQNONV$.SLQ.JOURNAL.SPINOUT:

```
//TESTSPN  EXEC PGM=S6BSPDSN,
//          PARM='GDG=$HLQNONV$.$SLQ$.JOURNAL.SPINOUT'
//STEPLIB  DD DSN=$HLQNONV$.$INSTVER$.AUTH,DISP=SHR
```

Alternate Parameters

You can also specify the following alternative parameters:

- `PARM='DSN=your_data_set_name'`

  If you use the DSN= parameter, S6BSPDSN returns the number of volumes allocated for the specified data set.

- `PARM='HLQ=high_level_qualifier'`

  This sets the return of S6BSPDSN to the number of data sets cataloged under the specified high level qualifier. This option dynamically invokes IDCAMS. It requires the following additional JCL statements:

```
//SYSIN    DD DISP=(new,delete,delete),unit=SYSDA,SPACE=(TRK,(5,5),
//          DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=8000)
//SYSPRINT DD DISP=(new,delete,delete),unit=SYSDA,SPACE=(TRK,(5,5))
```

For each of the above PARM functions, you can optionally add the DD statement:

```
//DSLIST   DD SYSOUT=A
```

or

```
//DSLIST   DD DISP=OLD,DSN=$HLQNONV$.$SLQ$.DSLIST
```

to obtain a list of file names processed by the request. In both cases, the output is FB with LRECL=80. The first two bytes represent the binary length of the data set name followed by the datas et name.

Return Codes

Ordinarily, the return code indicates the number of generations allocated in the GDG specified in the parameter or the number of volumes allocated to a data set if DSN is specified in the parameter.

Other return codes indicate the following error conditions:

Return Code	Meaning
1016	Parameter missing or less than 5 characters or keyword not GDG=, HLQ=, or DSN=
1012	Parameter header not GDG= or DSN=.
1008	HLQ= //SYSIN or //SYSPRINT omitted.
1000	Data sets not cataloged.

S6BSPJEX (Journal Data Extraction)

S6BSPJEX is part of the TIBCO Object Service Broker backup and recovery SPIN process. This utility extracts updated page images collected in journal data sets and copies them to an offline data set. Further processing steps in the SPIN process can merge the offline data set to build a continuous backup.

Invocation

The member SPIN01 of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

If you must spin a journal while your Data Object Broker is down, run S6BSPJEX with no parameters.

The S6BSPJEX utility accepts the following optional parameters:

AUDIT	Produces an audit trail that lists the checkpoint number and the number of pages for each checkpoint on the journal.
CHECK	Validates the data length and page count on the journal.
MDL=*model*	The model Execution Environment communications identifier (one to eight characters).
TDS=*commid*	The Data Object Broker communications identifier of your active TIBCO Object Service Broker system (one to eight characters).
USERID=*userid*	The user ID used to connect to the Data Object Broker. If USERID is omitted, the job name is used.

For the Data Object Broker to be notified that a spin is done, the MDL and TDS parameters must be specified.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Type	Reason	Meaning
0	Normal termination		

Return Code	Type	Reason	Meaning
4	Warning	04	Offload to dummy.
		08	Compression initialization failure.
		12	Offload complete. Data Object Broker not notified.
		16	ARCHLOG processing error.
8	ARCHLOG error	04	S6BTLALG load failure detected (ARCHLOG update routine).
12	Compression error	04	S6BSPCX0 error.
		08	Compression processing error.
16	Fatal error	04	Parameter syntax error.
		08	Offload data set open error.
		12	Journal compression space not available.
		16	Journal open error.
		20	Journal not formatted.
		24	Journal CI size too small.
		28	Control validation error.
20	VSAM error	04	VSAM error on journal read.
		08	VSAM error on offloead write.
		12	VSAM error rewriting the journal.

See Also *TIBCO Object Service Broker for z/OS Managing Backup and Recovery.*

S6BSPSEP (Separate Spun Journal by Segment)

S6BSPSEP is part of the TIBCO Object Service Broker backup and recovery process. Use S6BSPSEP as part of SPIN processing to separate the spun journal by segment.

Invocation

The S6BSPSEP member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The program reads the spun journal from the SPININ DD*name* and writes it by default to SPINOUT DD*name*. As it encountered data for new segments in tries to open a SPINO*nnn* DD*name* where *nnn* is the segment number. If the open succeeds, all data for the segment is written to that DD*name* otherwise it is written to SPINOUT. Any or all of the output files can be assigned to a DUMMY DD if you need only certain pages of a segment to be extracted or deleted.

The SYSPRINT contains the job statistics, including the number of pages encountered in each segment and where they are written to.

Return Codes

Upon successful termination, this utility returns zero return code. Problems with opening data sets, for example, result in the following abends:

Abend Code	Meaning
1	Unable to open SYSPRINT.
2	Unable to ope SPININ.
3	Unable to open SPINOUT*nnn*.
4	Spin record size too big.
5	Spin record size too small.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery.

S6BSPSPC (Determine Space Allocated to Sequential File)

S6BSPSPC is a general purpose utility that can be used to test the amount of space allocated to a sequential file.

Invocation

The S6BSPSPC member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility.This sample is provided as a reference only; modify the JCL for your needs.

The default parameter value is EXTENT. This returns the number of allocated extents. Setting the parameter value to TRACKS returns the number of allocated tracks.

Constraint

The input DISKFILE (DDNAME) must be a sequential file. Use the S6BSPCAP utility for VSAM files.

Return Codes

With the default parameter PARM=EXTENT, the return codes indicate the following:

Return Code	Meaning
0-16	The number of allocated extents.
44	Not PS/PO/DA data set.
48	Not disk data set.
52	Data set does not exist or is not cataloged.

Return Codes for PARM=TRACKS

When the alternate parameter, PARM=TRACKS, is used, the return codes indicate the following conditions:

Return Code	Meaning
0-4000	The number of allocated tracks.
4040	The number of allocated tracks exceeds 4000.
4044	Not PS/PO/DA data set.
4048	Not disk data set.
4052	Data set does not exist or is not cataloged.

S6BTLADM (Administration Menu)

S6BTLADM is an interactive monitoring and control facility for the TIBCO Object Service Broker environment. The utility displays statistics, generates diagnostic dumps, and browses Data Object Broker data.

Invocation This utility is run from a CLIST or as a TSO command. The CLIST ADMIN is distributed with TIBCO Object Service Broker and can be customized at installation time.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for more information about the CLIST and all options on the Administration menu.

S6BTLBPS (Back Up Page Data Sets)

S6BTLBPS is used to back up page data sets belonging to TDS segments. It takes all the used pages in the data sets, compresses them as much as possible, and then writes them to the output media (tape or disk).

Invocation

The S6BTLBPS member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

If a segment number is not specified as a parameter, 0 backs up the base segment (SEG=0). The segment must be offline.

S6BTLBPS can be called as part of the backup process.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Backup successful.
4	Compression disabled.
12	Open failure.

Audit Report

S6BTLBPS can produce an audit report that details the processing by page type with read and write counts, including page data set totals and percentage full. To enable the reporting feature, add a DD statement with ddname AUDIT as well as a runtime parameter of AUDIT.

To generate the report on a sequential data set, the record format required is (RECFM=FBA, LRECL=133).

Sample JCL

```
//BACKUP   EXEC PGM=S6BTLBPS,
//          PARM='SEG=01,AUDIT'
//STEPLIB  DD DSN=S6B.EL2.TNV1.INST52.AUTH,DISP=SHR
//DBDLIB   DD DSN=S6B.EL2.TVS1.OSBT52.DBDLIB,DISP=SHR
//JOURNAL  DD DSN=S6B.EL2.TNV1.OSBT52.BACKUP(+1),
//          DISP=(,CATLG,DELETE),SPACE=(CYL,(200,200),RLSE),
//          UNIT=3390,
//          DCB=(S6B.EL2.TNV1.OSBT52.MODEL.DSCB,
//          RECFM=VB,LRECL=4800,BLKSIZE=0)
//AUDIT    DD SYSOUT=*
/*
```

Sample Audit Report (First 80 Columns)

```
S6BBRPTR          POINTER VALIDATION FOR SEGMENT 1    DATE 2009 JUL 07 TIME 08:53     V520E049

TABLE/SIX NAME         I/1      D/B/R      H/G        S/6 F/X/0/?       TOTAL    ERRORS
SKIPPED SEGMENT #:0000          20112 RECORDS IGNORED
SKIPPED SEGMENT #:0063             59 RECORDS IGNORED
BACKUP  RECORDS READ       20227;          56 ACCEPTED;           0 UNUSED PAGES IGNORED

**** SYSTEM ***           1          1        0          0        0           2
@RULESLIBRARY             0          1        0          0        0           1
EVENTRULES                0          1        0          0        0           1
FIELDS                    0         10        1          0        0          11
MONTHLY_SALES             0          1        0          0        0           1
ORDERING                  0          2        1          0        0           3
PARMS                     0          8        1          0        0           9
SCREENFIELDS              0          1        0          0        0           1
SCREENS                   0          1        0          0        0           1
SCREENTABLES              0          1        0          0        0           1
SELECTION                 0          1        0          0        0           1
TABLES                    1          1        0          0        0           2
S6BBRPTR          POINTER VALIDATION FOR SEGMENT 1    DATE 2009 JUL 07 TIME 08:53     V520E049
TABLE/SIX NAME         I/1      D/B/R      H/G        S/6 F/X/0/?       TOTAL    ERRORS
********** ORPHAN PROCESSING INITIATED **********

S6BBRPTR          POINTER VALIDATION FOR SEGMENT 1    DATE 2009 JUL 07 TIME 08:53     V520E049
TABLE/SIX NAME         I/1      D/B/R      H/G        S/6 F/X/0/?       TOTAL    ERRORS

**** SYSTEM ***           0          0        0          0       22          22
GRAND TOTALS              2         29        3          0       22          56         0

          START TIME  08:53:44     END TIME  08:53:45
DATASET              READ      REFERENCED  OLDEST TIMESTAMP     NEWEST TIMESTAMP
PAGE1                  54              54 2009APR23 21:31:21  2009APR23 21:31:45
PAGE2                   1               1 2009APR23 21:31:45  2009APR23 21:31:45
PAGE3                   1               1 2009APR23 21:31:45  2009APR23 21:31:45
```

CONV Parameter

The CONV parameter converts page header dates from YYMMDD format to TIBCO Object Service Broker extended Julian date format (OYYDDD) during the page data set backup. In this format, O represents the century indicator (A - 2000, B - 2100, and so on), YY is the year within the century, and DDD is the day of the year. A message informs you if date conversion is activated.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

S6BTLBRM (Resource Management Online Backup)

S6BTLBRM (Resource Management Online Backup) creates a flat file back up of the resource data stored in the resource repository (Q1.Q2.RESOURCE). To create the backup, the utility connects to the specified Data Object Broker as an operator (refer to the TDS parameter) and retrieves the current resource data.

The backup file can be browsed or edited under TSO and it can also be converted to a KSDS VSAM data set using an IDCAMS REPRO.

Invocation

The S6BTLBRM member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Parameters

The parameters of the S6BTLBRM utility are in the form:

```
TDS=_____,MDL=_____,USER=_____
```

TDS	The commid of the target Data Object Broker from where the data is to be extracted.
MDL	The model communications identifier to be used for the connection.
USER	The TIBCO Object Service Broker user ID under which the utility is to connect. Defaults to the user's RACF ID.

If the SECUREADMIN Data Object Broker parameter is in use, the specified user ID must have administrator security clearance.

Output

AUDITLOG	Report of the back up. Record length: 132; Block length: 132
OUTPUT	Extracted resource management data. Record length: 128

Constraint

The Data Object Broker must be running when using this utility. To create a similar copy of the resource management data while the Data Object Broker is not running, use an IDCAMS REPRO.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Backup is successful.
12	No entries backed up.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

TIBCO Object Service Broker Parameters for information about the SECURADMIN Data Object Broker parameter.

TIBCO Object Service Broker for z/OS Installing and Operating for complete information about Data Object Broker resource repository.

The sample members S6BTLBRM and RESTRSCE in the TIBCO Object Service Broker JCL data set distributed with TIBCO Object Service Broker: S6BTLBRM uses the S6BTLBRM utility and RESTRSCE uses IDCAMS.

S6BTLCMD (Submit Operator Commands in Batch)

S6BTLCMD can be used to issue Data Object Broker operator commands in batch mode. This facility is useful in environments where TIBCO Object Service Broker administration personnel do not have access to a z/OS operator console.

S6BTLCMD steps can also be used in maintenance jobs to issue **DBOFFLINE** commands to bring a segment offline and **DBONLINE** commands after processing to bring the segment back online.

S6BTLCMD issues commands *only* to an active TIBCO Object Service Broker Data Object Broker. TIBCO Object Service Broker communications modules must be available (STEPLIB or LINKLIST). S6BTLCMD is capable of issuing any Data Object Broker operator command and is subject to the same security as other operator functions.

Invocation

The Data Object Broker operator commands must be entered one per line starting in column 1 and can be specified in one of the following ways:

- In a SYSIN file or PDS member

  The SYSIN file must be fixed block with record length 80.

- In-stream

The S6BTLCMD member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Parameters

The parameters of the S6BTLCMD utility are in the form:

```
PARM='delay_time, tds, mdl'
```

delay_time	A two-digit command delay time The delay time is in seconds (range 00 to 59) and occurs after each command is issued.
tds	The TIBCO Object Service Broker Data Object Broker communications identifier
mdl	The pattern for selecting the TIBCO Object Service Broker Execution Environment communications identifier

Return Codes

The results of the issued command can be verified by inspecting the operator console or system log. Return codes from S6BTLCMD are:

Return Code	Meaning
00	All commands were issued successfully.
08	One or more commands were rejected.
16	Cannot communicate with TIBCO Object Service Broker–no commands issued.
20	SYSIN file missing or invalid–no commands issued.
24	PARM= string missing or invalid–no commands issued

Error Conditions

If TIBCO Object Service Broker communication modules are absent, Abend S806 occurs.

See Also

TIBCO Object Service Broker for z/OS Installing and Operating for a complete list of Data Object Broker operator commands.

S6BTLDBR (Database Recovery Report)

S6BTLDBR collects TIBCO Object Service Broker recovery information in the primary and duplex redolog data sets, and in the cache data set. It then produces a report for TIBCO Support.

Invocation

The S6BTLDBR member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Characteristics of the report file are: RECFM=FBA, LRECL=133.

To print only the cache data set information, use the JCL without the PARM.

To print the redolog data set as well, use:

PARM='FULL'	To print the whole redolog
or	
PARM='PART'	To print the redolog for the second-to-last checkpoint and for the last checkpoint, plus the first record after the wrap point

Sample Report

The following illustrates the S6BTLDBR report:

```
Sample Report:  PARM=PART
                              DATABASE RECOVERY TOOL
                                VOLSER TRK  CHPT  DATE      #PGS  #TRX  FIRSTTRX LASTTRX  REDORBA  NEXTRBA  JRNLRBA  F
LAYCOC2.TNV1.OSBT52.CACHE1      USER08  525 00007 2009APR23 00030 00068 0000004E 00000091 000040A9 00008A72 0001B000 C0
LAYCOC2.TNV1.OSBT52.CACHE2      USER08  525 00006 2009APR22 00017 00007 00000047 0000004D 00003884 00003F1A 00012000 C0
                        REDOLOG - LAYCOC2.TVS1.OSBT52.REDOLOG
__RBA___ LEN_ TY __DATE__ ____TIME____ CURRCHPT LASTCHPT __TRAN#_ REDORBA_ JRNLRBA_ _____DESCRIPTION_____
00004000 80000000 00000000 00000000 20000000 00000000 00000000 00000000 00000000
00004020 0050 50 20090422 194733990080 00000007 00000006 00000000 00003F1A 0001B000 CHECKPOINT COMPLETE
00004070 0039 42 20090422 194734185380 00000007 00000006 00000000 00003F1A 0001B000 NORMAL SHUTDOWN
000040A9 0039 41 20090423 181852448350 00000007 00000007 00000000 000040A9 0001B000 FILETASK STARTED
000040E2 00D9 60 20090423 182140616427 00000007 00000007 0000004E 000040A9 0001B000 END/FULL INTENT
000041BB 0090 60 20090423 182140621421 00000007 00000007 0000004F 000040A9 0001B000 END/FULL INTENT
0000424B 00E5 60 20090423 182159234711 00000007 00000007 00000050 000040A9 0001B000 END/FULL INTENT
00004330 008E 60 20090423 182159464786 00000007 00000007 00000051 000040A9 0001B000 END/FULL INTENT
000043BE 0098 60 20090423 182825614303 00000007 00000007 00000052 000040A9 0001B000 END/FULL INTENT
00004456 0084 60 20090423 182825716810 00000007 00000007 00000053 000040A9 0001B000 END/FULL INTENT
000044DA 010C 60 20090423 182825864050 00000007 00000007 00000054 000040A9 0001B000 END/FULL INTENT
...
```

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Report completed successfully.

Return Code	Meaning
12	Unsuccessful completion. Check job log for associated information.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

S6BTLFAL (Format ARCHLOG)

S6BTLFAL is used to format the ARCHLOG. The ARCHLOG is a log file that retains information vital for recovery processing.

Invocation

The S6BTLFAL member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

ARCHLOG Information

Setup Requirements

The ARCHLOG is a fixed-block 132-byte data set that must:

- Reside on media that is capable of direct access processing
- Reside within a single extent
- Be of a block size large enough to retain the largest group (half-track blocking is recommended)
- Be formatted using S6BTLFAL before TIBCO Object Service Broker can use it. You can use the FORMAT1 member of the JCL data set for this original formatting.

Contents from Different Utilities

The ARCHLOG contains records of different record types. Which record types appear depends on the utilities you run. Not all types are created by all utilities:

Utility	Contents in ARCHLOG
The S6BSPJEX (Offload Journal Records) utility.	4 records per group: Identification Summary Input Output
The S6BSPX35 (Merge Output Sort Exit) utility.	A minimum of 3 records per group: Identification Input (count depends on how many SPINOUT data sets are included) Output

Utility	Contents in ARCHLOG
The S6BTLBPS (Back Up Page Data Sets) utility.	A minimum of 3 records per group: Identification Input (count depends on the number of page data sets in the segment) Output
The S6BTLFCA (Format Cache Data Sets) utility.	2 records per group: Identification Output
The S6BTLFJR (Format Journal Data Sets) utility.	2 records per group: Identification Output
The S6BTLFPS (Format TDS Page Data Sets) utility.	A minimum of 2 records per group: Identification Output (count depends on the number of page data sets in the segment)
The S6BTLRPS (Restore TDS Segment) utility.	A minimum of 3 records per group: Identification Input Output (count depends on the number of page data sets in the segment)
The S6BTLUPS (Unload a Page Data Set to Backup) utility.	3 records per group: Identification Input Output

This is the format of the five different record types.

Identification – identifies the job and time of the task:

- Start date (yyyymmmdd)
- Start time (hh:mm)

- End date (yyyymmmdd)
- End time (hh:mm)
- Duration (mm:ss)
- Job name
- Job identification number
- COMMID for the Data Object Broker
- Module name
- Return code
- Reason code
- Record type: 1

Summary – spin only, summaries input and output:

- Total pages read from input
- Total pages written
- Oldest checkpoint date (yyyymmmdd)
- Oldest checkpoint time (hh:mm)
- Oldest checkpoint number
- Newest checkpoint date (yyyymmmdd)
- Newest checkpoint time (hh:mm)
- Newest checkpoint number
- Record number: 2

Input – details of input data sets:

- Pages read from input
- Input data set name
- Volume serial of the input data set
- File format (VSAM, QSAM, or EXCP)
- Record number: 3

Output – details output data sets:

- Pages written to output
- Volume serial of the output data set
- Record number: 4

Comments – spin only, report errors, and so on:

- Text
- Record number: 5

Size Requirements

Before defining the ARCHLOG, you must determine how big to make it.

The following utilities account for the majority of the data in an ARCHLOG:

- S6BSPJEX (Offload Journal Records)
- S6BSPX35 (Merge Output Sort Exit)
- S6BTLBPS (Back Up Page Data Sets)
- S6BTLUPS (Unload a Page Data Set to Backup)

To estimate your space requirements, review the activity of your system as follows:

1. Review your system to determine the number of Spins occurring on an average day

   This also identifies the number of merges.

2. Calculate:

   — The number of segment backups you do a week with S6BTLUPS

   — The number of data sets you back up with S6BTLUPS

3. Determine the number of ARCHLOG records produced for each activity.

   Refer to the chart in Contents from Different Utilities on page 108.

4. Calculate the expected number of ARCHLOG records per week.

   It is recommended that you make your ARCHLOG big enough to retain a minimum of two full backup cycles.

5. Use the audit report produced by S6BTLFAL to verify that you created an ARCHLOG big enough for the total number of ARCHLOG records you estimated.

See Also *TIBCO Object Service Broker for z/OS Managing Backup and Recovery* for more information about the journal spin process.

Sample Report The following illustrates the audit report that comes out of this utility:

```
DATASET NAME     SYS1.STAR.ARCHLOG
VOLUME             STAR11
```

```
RECORD LENGTH          132
BLOCK SIZE           27984
BLOCKS/TRACK             2
RECORDS/TRACK          424
TRACK COUNT              2
TOTAL RECORDS          848
```

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Formatting is successful.
12	A processing error resulted in unsuccessful completion. Check the job log for associated information.

Related Utility

S6BTLPAL (Print ARCHLOG Information)

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for more information about the ARCHLOG.

S6BTLFCA (Format Cache Data Sets)

S6BTLFCA is used as part of the TIBCO Object Service Broker backup and recovery procedures. This utility initializes the cache data sets one at a time.

Invocation

The S6BTLFCA member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The cache data set you are initializing must reside within one extent.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Formatting is successful.
12	A processing error resulted in unsuccessful completion. Check the job log for associated information.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

S6BTLFCL (Format Contingency Log)

S6BTLFCL is used in database backup and recovery procedures in a TIBCO Object Service Broker network environment. S6BTLFCL formats the contingency log data set (REDOLOG.PENDING). The contingency log is a data set used by the Data Object Broker for:

- Recovery data when a transaction updates databases in one or more remote TIBCO Object Service Broker nodes
- TIBCO Object Service Broker network configuration information for remote nodes, peer servers, and external data gateway

Invocation

The S6BTLFCL member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Formatting is successful.
12	A processing error resulted in unsuccessful completion. Check the job log for associated information.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

S6BTLFJR (Format Journal Data Sets)

S6BTLFJR is used by the TIBCO Object Service Broker backup and recovery procedures. This utility formats the journal data sets prior to use. The journal DDNAME defines the data set to be formatted.

Invocation

The S6BTLFJR member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Return Codes

Upon successful termination, this utility gives a 0 return code. Codes on abnormal termination are:

Return Code	Meaning
102	Journal size is too small.
103	Journal contains data.
104	VSAM error detected on OPEN.
105	Journal characteristics are not all valid.
106	Error detected on VSAM WRITE.

In case of an abend, check the job log for associated diagnostic information.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

S6BTLFPS (Format TDS Page Data Sets)

S6BTLFPS formats all the pages in the page data sets of a TDS segment specified in the DD statements.

This utility checks for the minimum number of pages in a page data set.

Invocation

You must include a separate DD statement for every page you want to initialize.

The S6BTLFPS member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Formatting is successful.
4	No data sets were selected for formatting. Check the job log for associated information.
8	Pagestore could not be formatted because data set contains data. Check the job log for associated information.
12	A processing error resulted in unsuccessful completion. Check the job log for associated information.

See Also

TIBCO Object Service Broker for z/OS Installing and Operating for information on data set sizes.

TIBCO Object Service Broker Application Administion for information on increasing the size of existing data sets.

S6BTLFRL (Format Redolog)

S6BTLFRL is used by the TIBCO Object Service Broker backup and recovery procedures to format the redolog. It can also be used to format a duplex copy of the redolog if present. If the redolog is duplexed as defined in the DBGEN statement, a single execution of S6BTLFRL formats both the primary and duplex data sets if they are empty.

S6BTLFRL by default formats only empty data sets. If you specify the FORCE parameter, S6BTLFRL formats the redolog data set even if valid data appears to be present in the log. In most situations, you should delete and redefine the data set being formatted before invoking this utility. The redolog is defined using a standard system utility such as IDCAMS.

S6BTLFRL can also be used to produce a copy of the redolog. This could be necessary, for example, if a redolog or its duplex is damaged. Before you restart TIBCO Object Service Broker, create a copy of the good data set: whichever of the redolog or its duplex that is not damaged. Refer to Invocation for Copying on page 118.

Invocation

The S6BTLFRL member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Sample Output

The following illustrates sample output from the S6BTLFRL utility:

```
+S6BKX002I DSN=TEST.STAR.REDOLOG
+S6BTL075E REDOLOG DATASET NOT EMPTY
+S6BTL056I FORMATTING DUPLEX REDOLOG
+S6BKX002I DSN=TEST.STAR.REDOLOG.DUPLEX
+S6BDB035I REDOLOG CI SIZE IS 4096 BYTES
+S6BTL073I 000018000 PAGES SUCCESSFULLY FORMATTED
```

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Formatting is successful.

Return Code	Meaning
8	Redolog could not be formatted because the data set contains data. Check the job log for associated information. This return code would not appear when you use the FORCE parameter.
12	A processing error resulted in unsuccessful completion. Check the job log for associated information.

Results Summary

The following table summarizes the effect of invoking this utility on both primary and duplex redolog data sets:

Data Present in Primary Redolog Data Set	Data Present in Duplex Redolog Data Set	Result
Y	Y	Return Code 8
Y		Primary: Return Code 8 Duplex: Format
	Y	Primary: Format Duplex: Return Code 8
		Return Code 0 Both data Sets formatted

Invocation for Copying

When copying a redolog, S6BTLFRL checks that the following are true:

- The output redolog was formatted by S6BTLFRL
- The input redolog had been formatted and contains data
- The two data sets have the same size
- The two redolog data sets have the same control interval (CI) size
- Their CI size is either 4096 or 8192

The following are sample JCL statements for running S6BTLFRL to copy a redolog.

```
//COPY     EXEC PGM=S6BTLFRL,PARM='COPY'
//STEPLIB  DD DSN=$HLQNONV$.$INSTVER$.LOAD,
//            DISP=SHR
//DBDLIB   DD DSN=$HLQVSAM$.$SLQ$.DBDLIB,
//            DISP=SHR
//REDOIN   DD DSN=$HLQVSAM$.$SLQ$.REDOLOG,
//            DISP=SHR
```

```
//REDOOUT  DD DSN=$HLQVSAM$.$SLQ$.REDOLOG.DUPLEX,
//            DISP=SHR
//SYSPRINT DD SYSOUT=$SYSPRT$
//SYSIN    DD DUMMY
```

See Also

TIBCO Object Service Broker for z/OS Installing and Operating for more information on defining the redolog data set.

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

S6BTLPAL (Print ARCHLOG Information)

S6BTLPAL reads the ARCHLOG and prints a report starting with the most recent entry. Because the ARCHLOG is a direct access data set that wraps, it is difficult to view using tools like the TSO editor. The information in the ARCHLOG report is vital to the analysis of recovery requirements.

Invocation

The S6BTLPAL member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs. The output from this utility is written to the AUDIT DD. For saving the report to file, the characteristics of the report data set are: RECFM=FBA and LRECL=132.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Report processing has been successful.
12	Processing error has resulted in unsuccessful completion. Check the job log for associated information.

Related Utility

S6BTLFAL (Format ARCHLOG)

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery.

Sample Report

The following illustrates the S6BTLPAL report:

```
S6BTLPAL                                                    ARCHLOG REPORT                                2009JUL07 11:02:41

 MERGE    START 2009JUN28 21:41 END 2009JUN28 21:41 LAPSE 00:00:15 JOB BKCNVTH2 JOB24008 USR OSBBSTC  UTILITY HRNSPX35 CODE 00-00  1
   IN           74402 S6B.VTH.JOURNAL.SPINMRG.G0015V00             OSBD27 FORMAT QSAM                                          3
   IN                 S6B.VTH.BKUPCON.S002.G0017V00                OSBD26 FORMAT QSAM                                          3
  OUT           70857 S6B.VTH.BKUPCON.S002.G0018V00                OSBD29 FORMAT QSAM                                          4

 MERGE    START 2009JUN28 21:41 END 2009JUN28 21:41 LAPSE 00:00:14 JOB BKCNVTH0 JOB24007 USR OSBBSTC  UTILITY HRNSPX35 CODE 00-00  1
   IN          143040 S6B.VTH.JOURNAL.SPINMRG.G0015V00             OSBD27 FORMAT QSAM                                          3
   IN                 S6B.VTH.BKUPCON.S000.G0017V00                OSBD08 FORMAT QSAM                                          3
  OUT          139556 S6B.VTH.BKUPCON.S000.G0018V00                OSBD26 FORMAT QSAM                                          4

 MERGE    START 2009JUN28 21:41 END 2009JUN28 21:41 LAPSE 00:00:08 JOB BKCNVTH9 JOB24010 USR OSBBSTC  UTILITY HRNSPX35 CODE 00-00  1
   IN           30570 S6B.VTH.JOURNAL.SPINMRG.G0015V00             OSBD27 FORMAT QSAM                                          3
   IN                 S6B.VTH.BKUPCON.S009.G0017V00                OSBD05 FORMAT QSAM                                          3
  OUT           27756 S6B.VTH.BKUPCON.S009.G0018V00                OSBD28 FORMAT QSAM                                          4

MERGE    START 2009JUN28 21:41 END 2009JUN28 21:41 LAPSE 00:00:00 JOB BKCNVTH3 JOB24009 USR OSBBSTC  UTILITY HRNSPX35 CODE 00-00  1
   IN            4198 S6B.VTH.JOURNAL.SPINMRG.G0015V00             OSBD27 FORMAT QSAM                                          3
   IN                 S6B.VTH.BKUPCON.S003.G0017V00                OSBD01 FORMAT QSAM                                          3
  OUT             646 S6B.VTH.BKUPCON.S003.G0018V00                OSBD23 FORMAT QSAM                                          4

 MERGE    START 2009JUN28 21:41 END 2009JUN28 21:41 LAPSE 00:00:01 JOB S6H1BSPM JOB24006 USR OSBBSTC  UTILITY HRNSPX35 CODE 00-00  1
   IN            8804 S6B.VTH.JOURNAL.SPINOUT.G0002V00             OSBD16 FORMAT QSAM                                          3
   IN                 S6B.VTH.JOURNAL.SPINOUT.G0001V00             OSBD21 FORMAT QSAM                                          3
  OUT            3552 S6B.VTH.JOURNAL.SPINMRG.G0015V00             OSBD27 FORMAT QSAM                                          4

 SPIN     START 2009JUN28 21:41 END 2009JUN28 21:41 LAPSE    00:01 JOB S6H1BSP2 JOB24005 DOB S6H1DOBB UTILITY HRNSPJEX CODE 00-00  1
   IN            4553 OUT           4553 OLD 2009JUN15 04:26 CP            3745 NEW 2009JUN29 04:37 CP            3781          2
   IN            4553 S6B.VTH.JRNL2                               OSBD20 FORMAT VSAM                                          3
  OUT            4553 S6B.VTH.JOURNAL.SPINOUT.G0002V00            OSBD16 FORMAT QSAM                                          4
```

S6BTLPCA (Print Cache Information)

S6BTLPCA formats and prints information from the cache data set specified in the CACHE DD card. This information is used by TIBCO Support.

Invocation

The S6BTLPCA member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The output from this utility is written to the REPORT DD. For saving the report to file, the characteristics of the report data set are: RECFM=FBA and LRECL=121.

Sample Report

The following illustrates the first 80 columns of a partial S6BTLPCA report:

```
S6BTLPCA                                PRINT CACHE UTILITY                                PAGE: 001
              CHPTID: 0000006           DATE: 2009APR22              TIMESTAMP: C413A51A 0A32E983
              RRBA: 00003884            JRBA: 00012000               PAGES: 0000017
              TRX COUNT: 0000007        FIRST TRX: 00000047          LAST TRX: 0000004D

  TRACK: 001  0000 0200 0000 FFFF FFFF FFFF FFFF F000 A091 1219 4733 0000 0000 0006 0F80 0040     4089
              0000 0100 0000 FFFF FFFF FFFF FFFF F000 A091 1219 4733 0000 0000 0006 0F80 0040     4089
              0000 0000 0000 FFFF FFFF FFFF FFFF F000 A091 1219 4733 0000 0000 0006 0F80 0040     4089
              0000 0100 1268 FFFF FFFF FFFF FFFF C400 A091 1219 4733 0000 4D00 0006 0001 0098      184
              0000 0100 0E5A FFFF FFFF FFFF FFFF C700 A091 1219 4733 0000 4D00 0006 0001 0014       52
              0000 0000 0157 FFFF FFFF FFFF FFFF C400 A091 1219 4733 0000 4C00 0006 0001 002E       78
```

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Report processing is successful.
12	A processing error resulted in unsuccessful completion. Check the job log for associated information.

S6BTLPCL (Print Contingency Log)

S6BTLPCL 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.

Invocation

The S6BTLPCL member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The report data set characteristics are: RECFM=FBA and LRECL=121.

Sample Report

The following illustrates the S6BTLPCL report:

```
S6BTLPCL                          CONTINGENCY LOG PRINT                                  DATE 2009JUL07   TIME 11:33:07
  TRACK  006   LOCATION (CCHHR) 0246000501      TASK S6ELDOBC     LAST PHYSICAL WRITE:  DATE 2009MAR31   TIME 08:53:03
  USER   LAYCOCE      LOCK ID 00000014  TYPE PHASE 1     STATUS  PROGRESS   INITIATED:  DATE 2009MAR31   TIME 08:52:58
  ORIGIN: NODE NODEB                    USER LAYCOCE    LOCK ID 00000248              DATE 2009MAR31   TIME 08:52:58
                TYPE  SERVER ID  NODE             ACCESS ID  REGISTRATION   PROCESS  STATUS
 HEADER     50002214 0001C200 E2F6C5D3 C4D6C2C3 C3F7C7A7 B9664281 C3F7C7A2 D739B283   ......B.S6ELDOBCC7GX...AC7GSP..C
            F0F9F0F3 F3F10000 00000014 00000000 00000000 D5D6C4C5 C2404040 40404040   090331.............NODEB
            40404040 00000248 D3C1E8C3 D6C3C540 C3F7C7A2 D76CEB84 D3C1E8C3 D6C3C540      ....LAYCOCE C7GSP%.DLAYCOCE
            0008A800 1726C820 00000000 00000000 00000000 1726C810 00000091 00000000   ..Y...H.............H....J....
            00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ................................
            00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ................................
            00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ................................
            00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ................................
            00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ................................
            00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ................................
            00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ................................
            00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000   ................................
            00000000 00000000 00000000 00000000 00000000                              ................................
```

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Report processing is successful.
12	A processing error resulted in unsuccessful completion. Check the job log for associated information.

S6BTLRPS (Restore TDS Segment)

S6BTLRPS is used to do either of the following:

- Restore page data sets that are in a TDS segment, as part of the TIBCO Object Service Broker recovery process.
- Apply a partial backup on a segment, as part of operational maintenance. This is done using an optional APPLY parameter.

Prerequisites

Restoring a Data Set

Before you can restore page data sets, complete the following steps:

1. Ensure that you have complete backup copies that were created with S6BTLBPS.

   The target segment must be offline.
2. Extract the page data sets from the backup copy.
3. Initialize the target page data sets using S6BTLFPS.

Do not redefine and format the segment before you run S6BTLRPS to apply partial backups, that is, when using the APPLY parameter.

Invocation

Sample for Restoring a Segment

The S6BTLRPS member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Use:

PARM='SEG = ##'	To indicate the segment number.

Sample for Applying a Partial Backup

The following is a sample JCL to apply a partial backup to segment 1:

```
//RESTORE    EXEC PGM=S6BTLRPS,PARM='APPLY,SEG=01'
//STEPLIB   DD DSN=$HLQNONV$.$INSTVER$.LOAD,
//             DISP=SHR
//DBDLIB    DD DSN=$HLQVSAM$.$SLQ$.DBDLIB,
//             DISP=SHR
//JOURNAL   DD DSN=$HLQNONV$.$SLQ$.SPINOUT(0),
//             DISP=SHR
```

Use:

PARM = APPLY	To indicate that the partial backup can be accepted.
PARM ='SEG = ##'	To indicate the segment number.

Use the APPLY parameter with caution. If not used properly, you could corrupt your MetaStor.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Restore or apply is successful.
4	Restore or apply is successful but check the job log for warnings.
8	A processing error resulted in unsuccessful completion. Check the job log for associated information.
12	A processing error resulted in unsuccessful completion. Check the job log for associated information.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

S6BTLSRP (Select Recovery Pages)

S6BTLSRP is a recovery tool that restores the system to a specified point in time. This utility can be used if the Pagestore is improperly modified by a programming error.

Before you use this or any other database recovery tool, you should take all necessary precautions including backing up your current system.

Invocation

S6BTLSRP requires a datestamp and a timestamp in this form:

```
PARM='DATE=YYYYMMMDD,TIME=HHMMSS'
```

The time must be expressed in Universal Coordinated Time (UTC/GMT) to avoid discrepancies between time zones and between standard and daylight time.

The utility uses this date and time as the point of recovery and selects the pages with timestamps less than or equal to the date and time specified. Pages are selected on the basis of being closest to, but not after, the time and date specified in the PARM statement.

The S6BTLSRP member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

Considerations

- The PAGEFILE data set is the continuous backup (in segment/page sequence) used as input to this utility.
- The report data set has these characteristics: RECFM=FBA and LRECL=81.

Sample Report

The following illustrates the S6BTLSRP report:

```
S6BTLSRP                  POINT-IN-TIME PAGE SELECTION      2009JUL07  11:50:18
         PARMS: DATE = 2009MAR17   TIME = 010203
   //PAGEFILE  LAYCOC2.TNV1.OSBT52.BACKUP.G0014V00
   //EXTRACT   LAYCOC2.TNV1.OSBT52.EXTRACT
   //PAGEFILE DATE-TIME RANGE:  2006AUG16 20:29:57 -> 2009APR23 18:46:08
                  3  PAGES READ
                  0  PAGES WRITTEN
                  0  PAGES READ < TIME-DATE RANGE - NOT SELECTED
                  1  PAGES READ > TIME-DATE RANGE - NOT SELECTED
                  0  FREE OR LOGICALLY DELETED PAGES - NOT SELECTED
    RETURN CODE = 16   ABTERM: PAGEFILE LACKS BITMAP(S) FOR DATE,TIME GIVEN
```

Format of EXTRACT

The output data set EXTRACT is used to restore the Pagestore. You can pass this data set to S6BTLRPS. EXTRACT must be in the following format:

- RECFM VBS, VB, or V
- LRECL 4100
- BLKSIZE 0

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Restore is successful.
4	Investigate warning messages in the job log.
8	Investigate warning messages in the job log.
12	Investigate warning messages in the job log.
16	At least one bitmap is missing.
20	Backup is out of sequence.
24	A file processing error is encountered.
28	Parameter specification is in error.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

S6BTLUPS (Unload a Page Data Set to Backup)

S6BTLUPS is used to back up TDS segments that are currently offline. It takes a specified page data set and writes it to an output sequential data set.

You can set up batch jobs to execute S6BTLUPS for each data set in a segment and process the jobs simultaneously. This results in a faster backup of a large segment than other backup methods.

If you choose to run simultaneous jobs to back up your system, you must ensure that each job executes successfully before relying on the output as a backup.

Invocation

The S6BTLUPS member of the JCL data set distributed with TIBCO Object Service Broker contains sample JCL required to run this utility. This sample is provided as a reference only; modify the JCL for your needs.

The output of this utility must be incorporated into your regular backup.

Return Codes

Upon termination, this utility returns one of the following codes:

Return Code	Meaning
0	Backup is successful.
8	Backup is not successful. Check the job log for diagnostic information.

See Also

TIBCO Object Service Broker for z/OS Managing Backup and Recovery for complete information about TIBCO Object Service Broker backup and recovery.

SIXBUILD_CARDS (Prepare Cards for Batch Secondary Index Build)

SIXBUILD_CARDS helps you define the control cards required by the utility S6BBRSIX (Batch Secondary Index Build for TDS tables).

SIXBUILD_CARDS displays a series of screens for you to identify the table and secondary index fields, and then writes the necessary controls in a pre-allocated data set. For complete information about the Batch Secondary Index Build utility refer to S6BBRSIX (Batch Secondary Index Build for TDS Tables).

Invocation

Follow the instructions in the following sections to prepare cards.

- Prepare Data Set for Control Cards, page 128
- Invoke SIXBUILD_CARDS, page 128
- Specify Secondary Indexes, page 130
- Review Output Data Set, page 131

Task A Prepare Data Set for Control Cards

Before you can use SIXBUILD_CARDS, you must allocate a data set to hold the control cards. This data set must have a record length of 80 bytes and be fixed block.

Task B Invoke SIXBUILD_CARDS

To invoke SIXBUILD_CARDS, complete the following steps:

1. Execute SIXBUILD_CARDS from the workbench as follows:

```
EX execute rule ==> SIXBUILD_CARDS
```

The following is the initial screen is illustrated with sample input:

```
                Control Card Definition for SIXBUILD

Control Card File     : USR01.BATCH.CNTLCARD(SIX)

Table                 : EMP_MSTR_TDS

Page Fill Levels      : (Default = 75 % if blank)
                        Group Index     :
                        Secondary Index :

# of Input Records    :                    (Default = 100000          if blank)

Total # of Fields     :                    (Default = 50              if blank)

Dynamic Unit Name     :                    (Default = SYSDA           if blank)

Dynamic Block Size    :                    (Default = 4096            if blank)


PFKEYS: 1=HELP 3=EXIT 4=CONTINUE 12=EXIT
```

2. Supply values for the following fields:

Control Card File	The pre-allocated data set (and member name if the data set is partitioned). You can change the data set name by typing over it.
Table	The lower half of the screen specifies override values for the batch secondary index build process. Default values are available and appear on the screen. To override any of these defaults, type a new value in the space provided. The fields are explained in detail in Appendix A, Defining Batch Load Control Cards Manually, on page 133.
Page Fill Levels	For more information about adjusting Page Fill Levels for TDS tables, refer to Page Fill Tailoring on page 137.

3. To continue defining the control cards, press PF4.

Task C Specify Secondary Indexes

The following is the screen to specify secondary indexes:

```
                        Control Card Definition SIXBUILD

These are the fields and their key types for the selected HURON table.

Table: EMP_MSTR_TDS

    Primary Secondary   Field Name
       Y       _        PK
               _        F1
               _        F2
               _        F3
               _        F4
               _        F5
               _        F6




PFKEYS: 1=HELP 3=EXIT 4=CREATE CTL CARDS 12=CANCEL
Select the field for sixbuild by typing a 'Y' in front of it
```

This screen displays a list of TIBCO Object Service Broker TDS fields with the currently assigned primary keys. Type a Y in the Secondary column beside the fields for which you want to define a secondary index. You can define a secondary index on any field except the first primary key field.

After specifying the secondary index fields, press PF4. The control cards are written out to your data set and the control cards definition is complete.

Task D Review Output Data Set

The following is an example of an output data set containing the control cards:

```
S        75  75 0000000000100000 050 SYSDA
H 001 R                                                    EMP_MSTR_TDS
H 002 P PRM1                                  S C 0015
H 003 P PRM2                                  S C 0004
H 004 F PK                                    I C 0009     P
H 005 F F1                                    S C 0015     S
H 006 F F2                                    S C 0015
H 007 F F3                                    S C 0025
H 008 F F4                                    S C 0025
H 009 F F5                                    Q P 0006 02
H 010 F F6                                    S C 0008
H 011 F F7                                    C B 0002
```

The P and S designations next to the PK and F1 fields. These letters mean that PK is a primary key and F1 is a secondary index. If a field is both a primary key and a secondary index, the designation is Q.

SIXBUILD_CARDS Record Layout

These tables relate the on-screen control fields to their equivalent control-record fields:

Input File Definition	Screen	Record ID/Type	Columns
Table Name	1	H/R	64 - 79
Character Set:	1	H/R	9 - 12
Page Fill Levels:	1		
Group Index		S	3 - 4
Secondary Index		S	9 - 10
# of Input Records	1	S	15 - 29
Total # of Fields	1	S	31 - 33
Dynamic Unit Name	1	S	35 - 42
Dynamic Block Size	1	S	44 - 48
# Volumes for work file	n/a	S	50 - 51

Field Definitions	Screen	Record ID/Type	Columns
Secondary Key Fields	2		
TIBCO Object Service Broker Field Definitions:	n/a		
Name		H/F or P	9 - 44
Semantic Type		H/F or P	46
Syntax		H/F or P	48
Length		H/F or P	49 - 52
Decimal		H/F or P	54 - 55
Key Type		H/F	57

Considerations

Note the following:

- H type records contain a sequence number in columns 3 - 5, which must be consecutive and must commence with 001. The types within these records must be in the order: R (if any), followed by P (if any), followed by F.
- You use the Dynamic Block Size for any temporary work files required to process secondary indexes and for table instances. The small default block size can be detrimental to good performance. For large tables, that is, tables with large numbers of occurrences, consider optimizing the Dynamic Block Size to build secondary indexes.

Sample Report

The following illustrates the S6BTLPCL report:

Appendix A Defining Batch Load Control Cards Manually

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

Topics

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*. built the control card data set, you can use the information in this chapter to locate and change any particular field if necessary.

Types of Control Cards

The S6BBRTBL (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 137.

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 137.

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 137.

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 137.

Input Records To Process (columns 15 – 29)

The approximate number of input records to be processed during the run. The value must be right aligned and should not contain commas. If omitted, the default value is 100000. This value is used when calculating the size of the various dynamically allocated work files.

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)

The symbolic name of a DASD unit that contains all the dynamically allocated files. The default value is SYSDA. To process work files as quickly as possible, use VIO type symbolic names.

Dynamic Block Size (columns 44 – 48)

The block size for the dynamic work files. If omitted, the block size defaults to 4096 to accommodate a block within one swappable page of memory.

This block size is used to create secondary indexes and is extremely important for optimum performance. It should be set to the optimum size to achieve maximum performance benefit. Refer to Appendix B, S6BBRTBL/S6BBRSIX Tuning, on page 143 for more information.

#VOLUMES (columns 50 – 51)

When processing very large files and building secondary indexes, a single work file volume could be insufficient. Refer to Appendix B, S6BBRTBL/S6BBRSIX Tuning, on page 143 for more information.

Page Fill Tailoring

Tailoring Guidelines

With the S6BBRTBL (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 can 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), in the cross-referenced target field, **Output Definition Entry Type F entry**, you must enter the null equivalent value. If left blank, the default null equivalent value takes effect. For more information, refer to Appendix C, Null Handling, on page 151.

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). If your input data is not in this format or does not have syntax C, you can use an exit as described in S6BBRTBL (Batch Load) on page 41 to convert the data before loading it into the table.

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 BATCHUNLD_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.

[For fields other than raw data (RD) or Unicode (UN)] Type (columns 46)

This column contains the semantic type of the parameter or field.

[For fields other than RD or UN] Syntax (column 48)

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

[For fields of syntax RD or UN] Syntax (columns 45 – 48)

These columns contain the syntax.

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 keys 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 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 151 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 B S6BBRTBL/S6BBRSIX Tuning

This appendix describes how to tune large tables with secondary indexes or to build secondary indexes on existing large tables, tuning your configuration can significantly decrease the elapsed time of jobs.

Topics

Overview

Reason for Tuning

When loading large tables with secondary indexes or building secondary indexes on existing large tables, tuning your configuration can significantly decrease the elapsed time of the job.

Tuning Overview

Jobs should be run in a favored z/OS batch performance group, as they tend to be I/O bound. The activities undertaken by S6BBRTBL and S6BBRSIX and the tuning opportunities or impact at each activity are as follows:

1. Read input file/table, save index data on work file, output primary data pages:

File/Segment	Tuning
Input file	Optimum block size and DCB=(BUFNO= *xx*) where *xx* > 5 (S6BBRTBL only).
Work file	Optimum block size (refer to "Secondary Indexes and Block Size Allocation" on page 146).
Target Segment	Multiple (as many as possible) data sets with separate paths and volumes.

2. Read work file, output primary data index pages (S6BBRTBL utility only):

File/Segment	Tuning
Work file	Optimum block size (refer to "Secondary Indexes and Block Size Allocation" on page 146).
Target Segment	Multiple (as many as possible) data sets with separate paths and volumes.

3. Sort work file for each secondary and output secondary index page.

File/Segment	Tuning
Work file	Optimum block size (refer to “Secondary Indexes and Block Size Allocation” on page 146).
Target Segment	Multiple (as many as possible) data sets with separate paths and volumes.

If the input file and target segment already exist prior to the need for the proposed load or index build there is little opportunity for tuning these components, but if it is possible it is recommended to improve performance.

Work File Allocation and Tuning

Primary and Secondary Allocation

The work file is dynamically allocated during processing and is used to temporarily save index information prior to building table index pages.

- When only a single volume is available for the work file (the default), the primary allocation is always sufficient to complete the job based on the maximum number of records/rows (columns 15-29 of control record type S).
- The secondary allocation with a single volume is always 50 percent of the primary allocation.
- When multiple volumes are specified (in columns 50-51 of control record type S) the secondary allocation for the work file is 75 percent of the primary.

The default number of volumes available for allocation of the work file is one that is normally sufficient but with very large tables the space required could exceed the total space of a single volume. To reduce the primary allocation, in this case it is necessary to manipulate the number of records or rows.

Secondary Indexes and Block Size Allocation

When secondary indexes are to be built, the block size of the work file is critical: the elapsed time (with similar system activity) of a job using an optimum block size can be up to half that of the same job using the default block size of 4096 bytes.

Sample Job Preparation

The following is a step by step example of preparing for a large load that includes secondaries. The procedure for S6BBRSIX is almost identical, except that the input data comes from a table rather than a file, and therefore the amount of work file space required is slightly less than would be required for a load.

Procedure

1. Build the utility control file using BATCHLOAD_CARDS.

   A variable block data set containing approximately 150 million records is loaded to the table ATABLE, located on segment 1. This table:
   - Contains all data set information except the last field, FLD11
   - Is not subject to heavy modification online so pages require no free space (fill% = 99)
   - Requires 8 secondary indexes

```
S 99 99 99 99 000000150000000 050 WORKV     4096
I 001 R                                      V
I 002 F PRIMARY_KEY                            B 004           PRIMARY_KEY
I 003 F FLD01                                  C 006           FLD01
I 004 F FLD02                                  C 006           FLD02
I 005 F FLD03                                  C 029           FLD03
I 006 F FLD04                                  C 002           FLD04
I 007 F FLD05                                  C 007           FLD05
I 008 F FLD06                                  P 005           FLD06
I 009 F FLD07                                  C 005           FLD07
I 010 F FLD08                                  C 065           FLD08
I 011 F FLD09                                  C 010           FLD09
I 012 F FLD10                                  C 005           FLD10
I 013 F FLD11                                  C1500
H 001 R ENGL                                                   ATABLE
H 002 F PRIMARY_KEY                          I B 004    P
H 003 F FLD01                                I C 006    S
H 004 F FLD02                                I C 006    S
H 005 F FLD03                                S C 029    S
H 006 F FLD04                                I C 002    S
H 007 F FLD05                                I C 007    S
H 008 F FLD06                                C P 005    S
H 009 F FLD07                                I C 005    S
H 010 F FLD08                                S C 065
H 011 F FLD09                                I C 010    S
H 012 F FLD10                                I C 005
```

2. Build JCL for the utility job step.

Sample JCL follows:

```
//LOAD     EXEC PGM=S6BBRTBL,REGION=4M,
//          PARM='BROWSE=N,SEGMENT=01'
//STEBLIB  DD DSN=S6B.S50.AUTH,
//          DISP=SHR
//DBDLIB   DD DSN=S6B.050.DBDLIB
//          DISP=SHR
//CNTRL    DD DSN=S6B.050.S6BBRTBL.CARDS,
//          DISP=SHR
//INPUT    DD DSN=S6B.050.S6BBRTBL.DATA,
//          DISP=SHR
//AUDIT    DD  SYSOUT=*
```

3. Change PARM entry to BROWSE=Y as follows:

   ```
   PARM='BROWSE=Y,SEGMENT=01'
   ```

4. Run the job and obtain the audit output.

   Sample output follows:

```
CARDS READ  INPUT  24   HURON  24   SPEC   1   VALUE   0   TOTAL   49
 SPACE UTILIZATION (BYTES)  DATA 4010 INDEX 4010 SEC.IX 4010 GROUP.IX 4010
 MAX FLDS  50  MAX RCDS 150,000,000  DYNALLOC UNIT WORKV  ,01 DYNALLOC BLKSZ 4096
 FILE:   IXWORK LRECL=00083  BLKSIZE=04067          IXWORK
                                                 #BLOCKS=(0003061234,0001530610)
 S6BBL173I BROWSE MODE - REQUEST VALIDATION COMPLETED
```

5. Calculate optimum block size.

   With the default work data set block size at 4096 (actually 4067) space is required for over three million blocks of primary space on one disk volume, “UNIT WORKV, 01”.

   By obtaining the optimum block size or just under a half track for a fixed block data set with a record length of 83, the space required is minimized and sorting is optimized.

   Using either menu option 3.2 of ISPF or program **`IEFBR14`**, allocate a dummy data set with the characteristics of the work data set: FB UNIT=WORKV LRECL=83, specifying a block size of zero. The system calculates the optimum block size for that data set on that UNIT (WORKV).

   Replace 4096 in the S control record with the optimum block size. For example:

   ```
   S 99 99 99 99 000000150000000 050 WORKV    27971
   ```

6. Run the load again with BROWSE=Y.

   New sample load output follows:

```
CARDS READ  INPUT  24   HURON  24   SPEC   1   VALUE   0   TOTAL   49
 SPACE UTILIZATION (BYTES)  DATA 4010 INDEX 4010 SEC.IX 4010 GROUP.IX 4010
 MAX FLDS   50  MAX RCDS 150,000,000 DYNALLOC UNIT WORKV  ,01 DYNALLOC BLKSZ 27971
 FILE:   IXWORK LRECL=00083  BLKSIZE=27971          IXWORK
                                                  #BLOCKS=(0000445113,0000222557)
 S6BBL173I BROWSE MODE - REQUEST VALIDATION COMPLETED
```

7. Modify S control record.

   The load requires 445113 blocks (222557 tracks) of space for the work data set. One volume of this example's UNIT "WORKV" contains 50000 tracks, so the load requires four and a half volumes of disk space for the work file.

   The load always requests the total space required as the primary allocation. Therefore, it is necessary to modify the S control record so that the calculation results in a primary (and secondaries = 75% of primary) allocation suited to the space available.

   There are 7 volumes in the WORKV UNIT group; one volume is completely empty and is used for the primary allocation (approximately 100000 blocks). The remaining six volumes contain sufficient contiguous space to satisfy the secondary allocations (approximately 7 x 50000 blocks).

   Modify the control file type S record as follows:

```
S 99 99 99 99 000000033000000 050 WORKV    27971 07
```

Be careful not to specify more volumes than actually exist in the chosen UNIT. If you do, the job aborts.

8. Run the load again with BROWSE=Y.

   Sample output follows:

```
CARDS READ  INPUT  24   HURON  24   SPEC   1   VALUE   0   TOTAL   49
 SPACE UTILIZATION (BYTES)  DATA 4010 INDEX 4010 SEC.IX 4010 GROUP.IX 4010
 MAX FLDS   50  MAX RCDS 150,000,000 DYNALLOC UNIT WORKV  ,07 DYNALLOC BLKSZ 27971
 FILE:   IXWORK LRECL=00083  BLKSIZE=27971          IXWORK
                                                  #BLOCKS=(0000097932,0000048967)
 S6BBL173I BROWSE MODE - REQUEST VALIDATION COMPLETED
```

9. Change PARM entry to BROWSE=N as follows:

```
PARM='BROWSE=N,SEGMENT=01'
```

10. Check work file space availability.

    Before submitting the load, check that the work file space is available when required. Since the primary allocation is not sufficient to finish the load, the job could abort due to lack of space after wasting a considerable amount of resources as well as elapsed time.

Appendix C Null Handling

This appendix describes how to handle null values.

Topics

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 F, C, V, UN, and RD

Fields containing binary zeroes are considered to be nulls for syntax types F, V, UN, and RD. 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 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.

OLDNULL Option (Batch Unloads S6BBRULB and S6BBRULH Only)

If this EXEC parameter is specified, the unload utilities unload null values as zeroes. The parameter setting OLDNULL=Y 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 oldnull option:

```
EXEC PGM=S6BBRULB,PARM='SEGMENT=segment#,OLDNULL=Y'
EXEC PGM=S6BBRULH,PARM='U=userid,P=password,T=DOBname,M=modelname,OLDNULL=Y'
EXEC PGM=S6BBRULH,PARM='TTTTTTTTMMMMMMMMOLDNULL'
```

where:

TTTTTTTT	Is an 8 character DOBname (add blanks to the right of a shorter DOBname if necessary)
MMMMMMMM	Is an 8 character modelname (add blanks to the right of a shorter modelname if necessary)

Floating Point Nulls

You cannot represent a floating point null in fixed format. When a floating point null is unloaded using S6BBRULB or S6BBRULH, it is unloaded as true zero. Therefore, when floating point nulls are reloaded using S6BBRTBL, they are loaded as a zero.

Appendix D Batch Journaling

This appendix describes how to use batch journaling.

Topics

Journaling Updated Page Images

Batch utilities that update segments can journal the updated page images. These images are equivalent to the ones produced online by the Data Object Broker in the form of journal SPINs.

The following utilities can journal their updated page images:

- S6BBRCLR (Batch Table Clear)
- S6BBRIAL (Move ACCESSLOG)
- S6BBRPGC (Pagestore Correction)
- S6BBRSET (Batch Segment Re-initialization)
- S6BBRSIX (Batch Secondary Index Build for TDS Tables)
- S6BBRTBL (Batch Load)

Required Process

To implement batch Journaling, you must perform the following. This process is described in more detail in Specific Implementation Steps on page 159.

- For each utility, define a batch journal data set.
- Add the parameter PARM='...,JOURNAL=Y' to the utility's EXEC statement.
- Add a //JOURNAL DD statement to the batch utility job step.
- Add subsequent job steps to the utility JCL so that the data set can simulate a SPIN.

You can then introduce the journaled page images into your normal spin/backup merge process.

JCL data set Member Provided

Use the member BATJRNL in the JCL data set provided with TIBCO Object Service Broker as a guide when setting up this process. This member includes:

- A batch journal allocation
- A sample using S6BBRCLR to illustrate batch journaling
- An IDCAMS step to perform a SPIN of a journal
- Steps to check SPINOUT generations and to submit a merge spin job, if a merge job is required.

Specific Implementation Steps

Step 1: Customize Symbolic Parameters

Use the ISPF standard OSEMOD customization macro to customize the symbolic parameters for the JCL member BATJRNL.

Step 2: Modify JCL

Modify your utility JCL to conform to the BATJRNL member in the JCL data set distributed with TIBCO Object Service Broker. This involves several steps:

1. Using step 1 of the BATJRNL as a sample, add JCL to allocate the batch journal.

   Do *not* attempt to initialize batch journals (unlike with Data Object Broker journals).

2. Add PARM='....,JOURNAL=Y' to the utility EXEC parameter.

3. Add JCL to reference the batch journal data set in the utility step, for example:

   ```
   //JOURNAL   DD   DISP=OLD,DSN=$HLQVSAM$.$SLQ$.BATCH.JRNL1
   ```

4. Add subsequent job steps SPIN1, TESTSPN, and SPAWN to do the following:

   — SPIN1 to copy the batch journal data to a new SPINOUT GDG.

   — TESTSPN to check the number of generations of SPINOUT GDGs.

   — SPAWN to submit a SPINMRG job to consolidate the SPINOUT GDGs into a new SPINMRG GDG data set. This job step executes only if the number of SPINOUT GDGs found by step TESTSPN exceeds your generation limit.

Post Processing Requirements

After completing the previous steps, you must also do the following:

- Copy the journaled images into your spin/merge process even if the utility does not terminate successfully.
- Delete and redefine the batch journal data set, each time you run the utility.
- As with all SPINs, ensure that all journaled images are first copied into your spin/merge process successfully. Failure to do so causes your spin merged backup to be incorrect.
- Run S6BBRPTR (Batch Pointer Check) against your spin/merged backup on a regular basis.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W