TIBCO® Object Service Broker for z/OS

Managing Backup and Recovery

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 explains the backup and recovery features of TIBCO OSB for z/OS. It describes system key components and describes how you can back up your data and recover from errors. You can use this information, with assistance from your TIBCO Support representative, to develop the best customized solution for your unique backup and recovery requirements.

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 Introducing TIBCO Object Service Broker Backup and Recovery Components

This chapter describes the TIBCO Object Service Broker backup and recovery components for z/OS.

Topics

Overview

Operational Components

There are two major operational components of a TIBCO Object Service Broker system that are key elements of a backup and recovery strategy. They are:

- Data Object Broker
- Execution Environment

These components are discussed in the following sections.

Data Set Components

The Data Object Broker uses a number of key operational data sets to provide complete data integrity and maximum recoverability:

Data set	Refer to page…
Pagestore	7
Redolog	8
Contingency log	8
Cache	8
Journals	9
DBDLIB	9

Data Object Broker

Definition

The Data Object Broker is the server for TIBCO Object Service Broker data. It is responsible for all online operations against the Pagestore, which is the repository for all TIBCO Object Service Broker data. For more information about the Pagestore, refer to TIBCO Object Service Broker Data Sets on page 7.

Data Object Broker Functionality

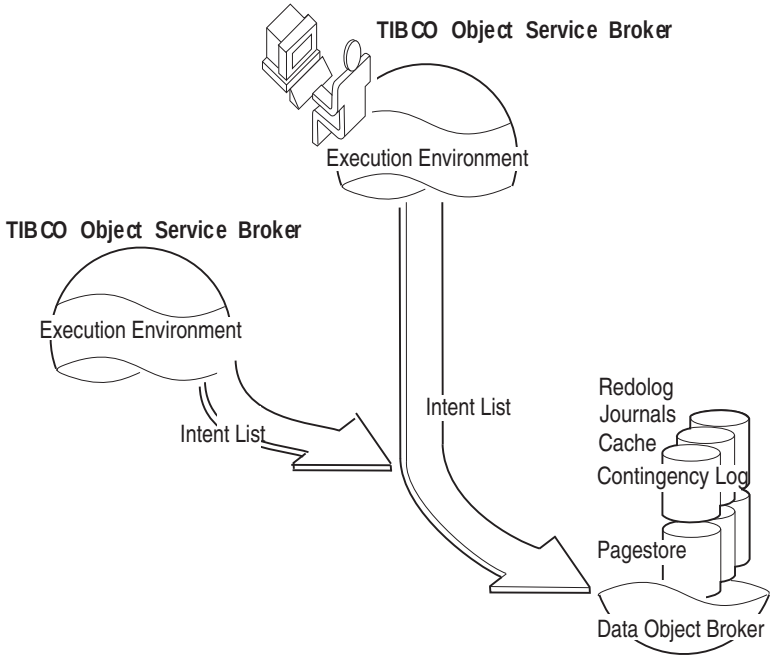
When the Data Object Broker receives a commit request from the Execution Environment, it performs the following:

1. Obtains and locks the required data pages from the Pagestore
2. Places the pages in memory
3. Records the update requests in the redolog
4. Updates the pages in memory and places them on a queue in memory for checkpoint processing
5. When a checkpoint occurs, writes the pages to a cache data set and propagates the changes to the journal data set and to the Pagestore

Committing Updates

All accesses and changes to the Pagestore go through the Data Object Broker. If a commit request involves another system (an external database or a peer TIBCO Object Service Broker system), the Data Object Broker logs the request in the contingency log until the update request is confirmed by the other system. For more detail, refer to Chapter 3, Understanding Fail Safe Processing, on page 15.

The Role of the Data Object Broker and Related Data Sets



Data Object Broker Communication

The method of communication between the Data Object Broker and Execution Environments depends upon what facilities are available. TIBCO Object Service Broker always attempts to use Cross Memory Services (XMS) for its communications but uses VTAM if XMS cannot be used, and TCP/IP if neither VTAM nor XMS are available.

Execution Environment

Definition

The Execution Environment runs, or executes, the TIBCO Object Service Broker applications for the end user. It manages such things as:

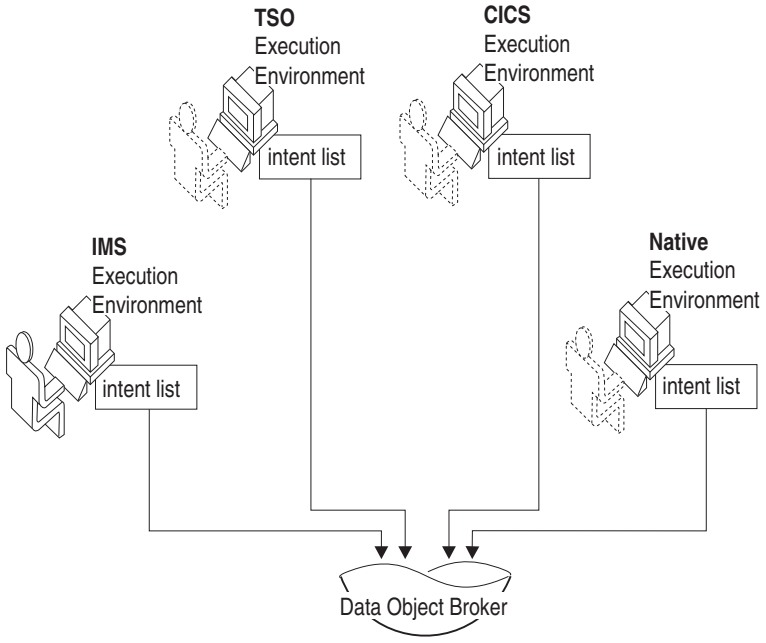
- Screen I/O
- Rules interpretation
- Logical data control

Execution Environment Functionality

The Execution Environment deals with the Data Object Broker on behalf of the user or application and passes on requests for data and commands to update the database.

The Execution Environment makes requests to the Data Object Broker to obtain information, processes the information, and presents it in the form of screen displays or reports. An Execution Environment can connect to only one Data Object Broker, whereas a Data Object Broker can support many Execution Environments. A Data Object Broker can also support connections to other peer Data Object Brokers for remote data access. In short, the Execution Environment provides the user and application interface to the physical data.

Relationship Between Execution Environments and a Data Object Broker



Types of Execution Environments

Depending on your system environment, you can have several types of Execution Environments. These include:

- TSO Execution Environment (single-user)
- Batch Execution Environment (single-user)
- CICS Execution Environment (multiple-user)
- Native Execution Environment (multiple-user)
- IMS TM Execution Environment (multiple-user)

Parameters

Execution Environments require a number of parameters. System-wide defaults for each type of Execution Environment are defined during installation but each Execution Environment can specify its own defaults at startup.

See Also *TIBCO Object Service Broker Parameters* for more information about Execution Environment parameters.

TIBCO Object Service Broker Data Sets

The following sections identify the various data sets used by TIBCO Object Service Broker. For performance reasons, each TIBCO Object Service Broker data set should be located on a different device. This is very important for high-use data sets such as the MetaStor, the redolog, the contingency log, and the cache. For faster recovery from DASD failures, the redolog data sets can be duplexed, so that identical updates are written to two data sets, the primary and duplex ones. For information about setting up duplex copies, refer to Duplexing the Redolog for Recoverability on page 58.

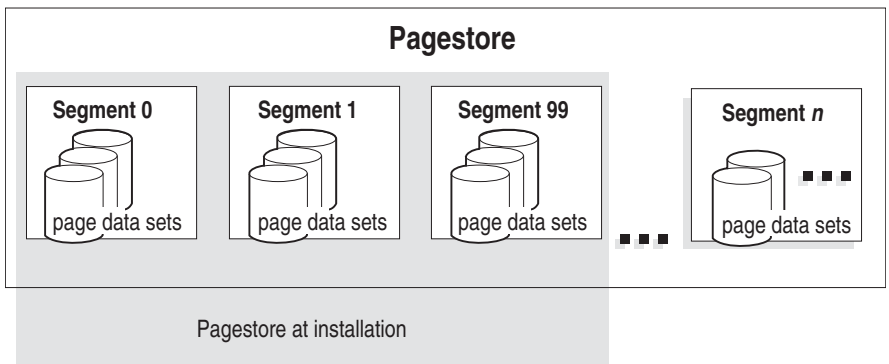
See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for more information on the definition, placement, and sizing of these data sets.

Pagestore

The Pagestore is a collection of VSAM ESDS data sets used for storing data in tables. In simple terms, this is the database. The TIBCO Object Service Broker Pagestore stores data using the TDS (Table Data Store) method. TDS tables are stored in a B+ tree structure.

Pagestore Segments

The Pagestore is divided into partitions known as segments. Each segment contains between 1 and 128 data sets that contain data, as shown in the following illustration.



MetaStor

The only required segment is the base segment (ID = 0), also known as the MetaStor. It contains table definitions, rules, and an index that identifies where the tables are defined. The MetaStor is the most heavily used segment. As distributed, TIBCO Object Service Broker has three segments and the MetaStor has three data sets.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for more information on the Pagestore.

TIBCO Object Service Broker Application Administion for more information about TDS segments and modifying the size of your Pagestore.

Redolog

The redolog data set retains update requests from the Execution Environment that were processed. TIBCO Object Service Broker uses the redolog in recovery situations to reprocess committed updates made to the Data Object Broker since the last checkpoint.

Contingency Log

The contingency log data set contains a record of all transactions that span resources, such as the local Data Object Broker and one or more external resources. An external resource is a database server or a peer TIBCO Object Service Broker Data Object Broker.

In the event of a power failure or external system failure, TIBCO Object Service Broker uses the contingency log to ensure consistency of updates across various peer TIBCO Object Service Broker systems and external database management systems.

For more information about the role of the contingency log in distributed data environments, refer to Chapter 3, Understanding Fail Safe Processing, on page 15.

Cache

TIBCO Object Service Broker uses two cache data sets as write-ahead logs for page images marked as modified at checkpoint time. When the system performs a checkpoint, the modified pages in memory are written to the cache, assuring their recoverability in the event of a failure. The data pages are written out asynchronously to the journals and to the Pagestore.

Journals

TIBCO Object Service Broker supports 2 to 255 journal data sets. The journals are used to retain updated page images. Only one journal is active at a time. When a journal is full, it spins off the page images to a backup via the spin process, which prepares the journal for re-use. For more detail, refer to Spinning the Journals on page 29.

If you have some segments that store easily reproducible data and you are sure that data recovery is not an issue, you can turn off journaling for those specific segments. For more information on this option, refer to Chapter 4, Understanding Journal Processing, on page 27.

DBDLIB

This data set is the repository for data set information for the Data Object Broker and for many utilities. The DBDLIB contains a description of all data sets known to TIBCO Object Service Broker, including:

- Name
- Usage
- Descriptions of the segments comprising the Pagestore
- Descriptions of the journals
- Data set name qualifiers for the operations data sets
- In the case of segments, the number of data sets

Chapter 2 Understanding Transaction Processing

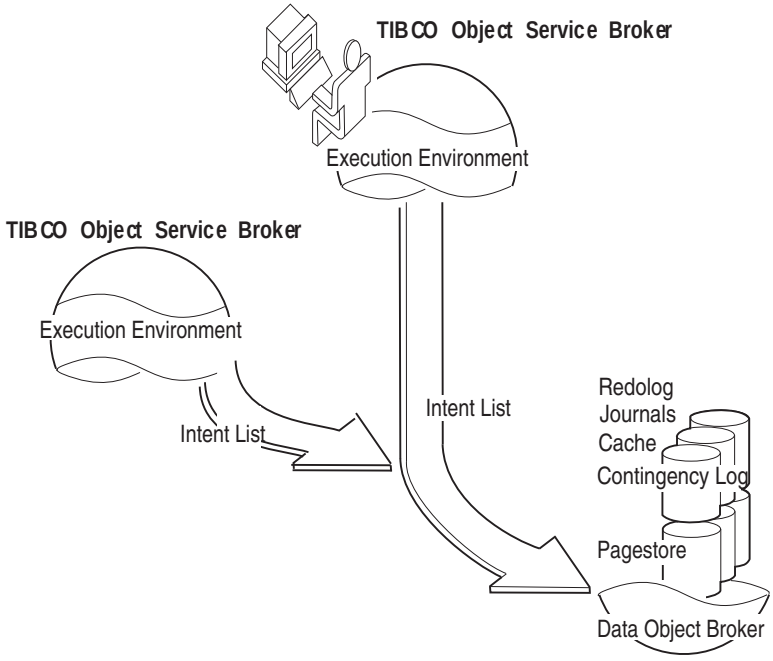
This chapter describes transaction processing for TIBCO Object Service Broker.

Topics

Overview

This chapter takes you through a typical TIBCO Object Service Broker transaction and explains what goes on behind the scenes from an operations perspective. The transaction used in the example is assumed to be running in update mode and updates a local TDS table.

Key Components Involved in Transaction Processing



Sample Transaction

Task A The Transaction Requests Data from a Table

The request for data can be a GET or a FORALL on a table. The following occurs:

1. The Execution Environment sends the request to the Data Object Broker. The Execution Environment requests that a logical shared-lock be placed on the occurrences being accessed.

   If the transaction runs with BROWSE=YES, no logical locks are taken.

2. The Data Object Broker retrieves the required data pages from memory or from the Pagestore and loads it into internal memory (assuming that the page is not already there).
3. The Data Object Broker passes the data to the Execution Environment.

Task B The Transaction Performs an Update

Possible accesses include the following:

- INSERT
- DELETE
- REPLACE

During the access, the following occurs:

1. The Execution Environment sends a lock upgrade request to the Data Object Broker to obtain an exclusive lock on the occurrence that it plans to update.
2. The Execution Environment formats an update request and saves the request in its intent list.

The data is not yet physically changed and is not changed until a commit is issued. The Execution Environment keeps track of the changes, however, to ensure integrity and consistency are maintained.

Task C An Intermediate Commit Command is Issued

The following occurs after the commit command is issued:

1. The Execution Environment passes the intent list to the Data Object Broker.
2. If an updated segment reaches its predefined capacity threshold, that is, it is nearing full capacity, warning messages are sent to the operator console.

3. The Data Object Broker updates data pages in memory. This ensures that these new database updates are reflected in future accesses to the same data by other transactions.
4. The Data Object Broker writes the database update request (intent list) to the redolog. This operation guarantees that all committed updates are written to the database.
5. The Execution Environment is notified to proceed with the application. It clears the intent list, but not the database locks that are held by the Data Object Broker, and continues execution of the application.
6. If a checkpoint is required, the Data Object Broker invokes its asynchronous checkpoint processor, which writes all the updated pages to the cache, journals, and Pagestore itself.
7. If the current journal fills up during a checkpoint, the Data Object Broker starts the spin process (refer to Spinning the Journals on page 29 for details) and switches to another journal.

Task D The Transaction Ends

The following occurs:

1. Since the ending of a transaction implies a Commit, the Execution Environment passes the intent list to the Data Object Broker. These are the commands issued after the most recent Commit.
2. The Data Object Broker processes the commands as before, updating pages, writing them to the redolog, and so on.
3. All locks held on behalf of the transaction are released.

Implications for Backup and Recovery

Here are some potential implications of transaction processing on your backup and recovery system.

Frequency of Checkpoints

You must ensure that you monitor the frequency of checkpoints incurred when new batch processes are added to your TIBCO Object Service Broker system. If the process causes a large number of pages to be updated, the resident page manager buffers fill quickly causing a significant increase in checkpoint requests. Both the redolog and journals must be large enough to accommodate the increased activity.

Chapter 3 Understanding Fail Safe Processing

This chapter describes fail safe processing for TIBCO Object Service Broker.

Topics

Fail Safe Processing

Definition

Fail Safe is the TIBCO Object Service Broker term for the commit strategy.

Fail Safe processing is controlled and coordinated by the Data Object Broker where the commit for the unit of work is issued. The coordinating Data Object Broker synchronizes all participating service providers to ensure that all updates, local and remote, either commit or rollback as one unit of work. This ensures data integrity.

A service provider is one of the following:

- A local service for TDS tables, within the local Data Object Broker
- A peer Data Object Broker
- An external database server

Fail Safe Strategies

Determining a Fail Safe Strategy

If you have a multiple-systems environment, you must decide on an appropriate Fail Safe strategy. To do this, you must determine the Fail Safe level that meets your needs.

Fail Safe Levels

Use the following table to determine your required Fail Safe level:

Transaction Requirements	Fail Safe Level
Commits can be handled serially. External service providers commit first followed by local updates to TDS tables.	0

Transaction Requirements	Fail Safe Level
Transaction involves a local service for TDS tables and an external service provider. Local updates are contingent upon the completion of external updates.	1
Transaction involves multiple Data Object Brokers.	2

The following sections describe each Fail Safe level in detail.

Using the Resource Manager facility of the Administration menu, you can identify the maximum Fail Safe level supported by a service provider. During commit processing, the commit coordinator specifies, within defined limits, the commit strategy to use:

Type of Update [a]	**Type of Commit Given to Peer Server**
One resource only (local resource or remote resource).	Fail Safe level 0
Local and one remote resource.	Fail Safe level 1
Local and/or at least two remote resources.	Fail Safe level 2

a. For all three update types, the local resource is TDS on the local node and the remote resource is a peer.

Fail Safe Level 0 (Serial)

Fail Safe level 0 provides for the issuance of commit requests to each service provider in a commit group, one at a time. If there is only one updated resource, Fail Safe level 0 is adequate. When there are two or more service providers, there is potential for data to get out of sync if an error occurs.

Consider, for example, a commit group that has three service providers:

- Server 1
- Server 2
- Local TDS tables

This is how the Fail Safe process proceeds:

1. Server 1 is issued a commit. When it responds successfully, the commit continues. Otherwise, the commit abends.
2. Server 2 is issued a commit. When it responds successfully, the commit continues. Otherwise, if DBPROFILE=1, the commit group is terminated, abandoning all further updates; if DBPROFILE=0, the commit continues.
3. The local TDS table updates are committed.

Data synchronization errors between the service providers can occur if failures or communication loss happens after Server 1 responds successfully to the commit. If Server 1 does not respond, it is unknown whether Server 1 committed or not.

Changes to external resources (peer Data Object Brokers and external databases) are committed in the order in which the resources are first referenced, not necessarily in the order in which the resources are updated.

Fail Safe Level 1 (Contingent)

Fail Safe level 1 is restricted to environments where there are local updates and one service provider with remote (external) updates. An external service provider is capable of Fail Safe level-1 processing if a transaction database is defined and available. Refer to Transaction Database on page 20 for more information.

Commit processing under Fail Safe level 1 functions as follows:

1. The (local) Data Object Broker saves the intent list (that is, the TIBCO Object Service Broker component of the transaction) in the contingency log.
2. The (remote) external service provider generates a transaction control record and adds it to the pending database updates, then commits. The (local) Data Object Broker commits the intent list after receiving confirmation that the external update was successful.
3. If connection to the external service provider is lost before commit success or failure notification is received by the originating Data Object Broker, the associated transaction on the contingency log becomes an in-doubt transaction.
4. When communication is re-established, the external service provider can be queried for the presence or absence of the transaction control record that it added to the pending database updates. This verifies whether it was able to commit the unit of work.
5. If the control data is not found, the external system did not successfully commit and therefore any associated updates must be abandoned. If it is found, the updates are committed.

A combined IMS/DB2 external system is considered one resource and therefore can be supported under Fail Safe level 1.

Fail Safe Level 2 (Two-Phase Commit)

Fail Safe level 2 coordinates the commit of a database unit of work that is distributed across multiple service providers. To operate at Fail Safe level 2, the participating service providers—a combination of Data Object Brokers and external database servers—must be capable of working with the TIBCO Object Service Broker commit coordination protocol.

In phase 1, when the coordinating service provider signals all other participants to prepare to commit, each participating service provider logs the pending unit of work in its own contingency log.

Having received acknowledgments of a successful phase 1, the coordinator starts phase 2, giving the commit signal, and each service provider actually performs the commit.

Contingent Two-Phase Commit

In this instance, a unit of work is spread across multiple Fail Safe level-2-capable resources, and one Fail Safe level-1-capable resource.

Commit processing under this condition functions as follows:

1. Fail Safe level 2 begins and the commit coordinator signals a prepare to commit to all participants.
2. The Fail Safe level-1-capable resource is signalled to commit and all its commits are completed.
3. Fail Safe level-2 commits are completed.

You must identify the highest commit level supported by a resource in the resource manager definitions. For more information about the resource manager, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

Components Supporting Fail Safe Processing

There are two key components that support Fail Safe processing:

- Transaction database
- Contingency log

These components are described in the following sections.

Transaction Database

The transaction database, which resides on the external database system, is a part of Fail Safe processing for all external service providers. It consists of a table (or equivalent in the external database's terminology) containing a number of fields identified by TIBCO Object Service Broker. In the case of a peer TIBCO Object Service Broker Data Object Broker, the peer's local contingency log is used as the transaction database. The fields are:

Contents of field	Syntax	Length
Data Object Broker name.	C	8
Server identifier.	C	8
Transaction ID.	B [a]	4
Date/Time.	B	8
Server registration data.	C	25

a. Some databases require the definition of binary fields as character syntax.

Each field is assigned a name. The field name varies depending on the requirements of the external database. In most cases, this table (or equivalent) has a default name (such as S6BTRXDB). This name can be customized provided that the customized name is reflected in the TRXDB server parameter.

See Also The appropriate *TIBCO Service Gateway* manual for more information on the server parameters.

Contingency Log

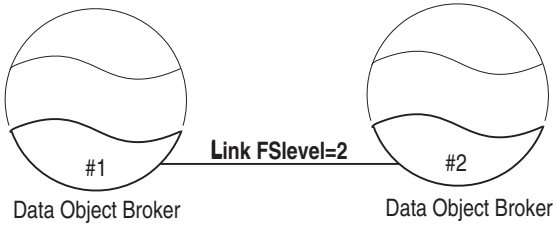
The second file is the contingency log, which is used by the Data Object Broker to hold transaction updates that are contingent upon service provider acknowledgments. This file is used when updates to multiple databases (whether local TDS or external service providers) occur in the same TIBCO Object Service Broker transaction.

Sample Distributed Processing Scenarios

The following scenarios illustrate Fail Safe processing in environments where there are multiple TIBCO Object Service Broker systems with or without external database servers.

Two Data Object Brokers

Consider a transaction in which a TDS table is updated in Data Object Broker #1 and another TDS table is updated in Data Object Broker #2. The Data Object Brokers are assumed to be connected by a link where the Fail Safe level is 2 as described in the following illustration:



In this case, Data Object Broker #2 is capable of Fail Safe level 2; however, because there is only one service provider and local TDS updates, Fail Safe level 1 provides adequate data integrity.

Commit Cycle for Two Data Object Brokers

The sequence of events for this commit cycle is as follows:

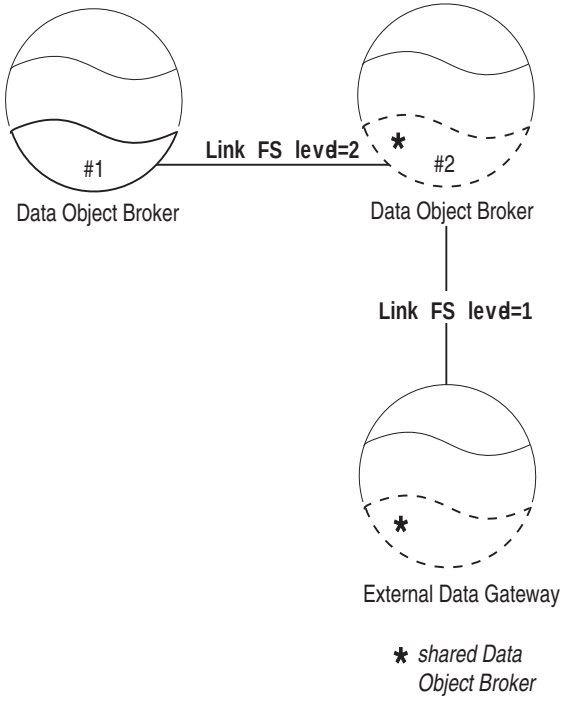
1. Data Object Broker #1 stores its pending updates in the contingency log.
2. Data Object Broker #1 sends a Fail Safe level-1 commit to Data Object Broker #2.
3. Data Object Broker #2 stores its pending updates in its contingency log.
4. Data Object Broker #2 commits its updates.
5. When Data Object Broker #1 receives the success message from Data Object Broker #2, it commits its updates (if a failure message is received, it abandons the updates).

6. Data Object Broker #1 sends a contingency log release directive to Data Object Broker #2 so that the remote contingency log entry can be released.

The same processing occurs if Data Object Broker #2 is an external database server, except for steps #3 and #4. In steps #3 and #4, the transaction database has an update added and the commit is processed by the external database server.

Two Data Object Brokers with an External Database Server

If Data Object Broker #2 has an external database server attached to it involved in the same transaction, it processes the request as for Fail Safe level 1. The following illustrates this process:



Commit Cycle for Two Data Object Brokers with an External Database Server

The sequence of events for this commit cycle is as follows:

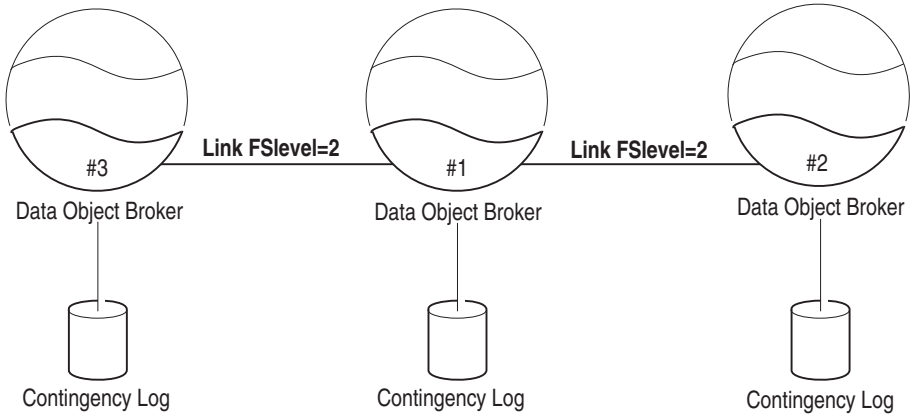
1. Data Object Broker #1 logs its intent list to the contingency log.
2. Data Object Broker #1 issues a Fail Safe level-1 Commit to Data Object Broker #2.

3. Data Object Broker #2 writes its intent list to the contingency log.
4. Data Object Broker #2 sends a Fail Safe level-1 commit to the external database.
5. Data Object Broker #2 commits its TIBCO Object Service Broker transaction, previously saved in the contingency log, and signals Data Object Broker #1 (the commit coordinator) that the commit was successful.
6. Data Object Broker #1 completes its commit processing.

Data Object Broker #1 is not aware that the commit protocol between Data Object Broker #2 and the External Data Server is at Fail Safe level 1.

Three Data Object Brokers

Another possible scenario is a TIBCO Object Service Broker transaction with TDS table updates on three Data Object Brokers as shown in the following:



Commit Cycle for Three Data Object Brokers

Since more than one updated service provider is involved in the transaction and each is capable of supporting Fail Safe level-2 processing, the sequence of events becomes:

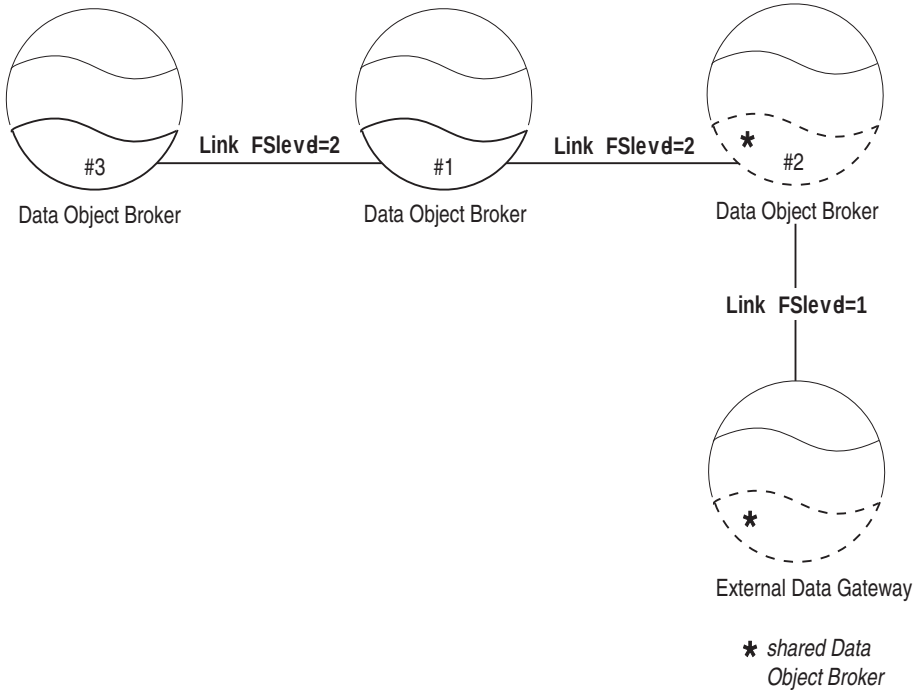
1. Data Object Broker #1 issues a prepare to commit to Data Object Brokers #2 and #3.
2. Data Object Broker #1 saves its intent list in its contingency log.
3. Data Object Brokers #2 and #3 each save their intent lists in their contingency logs and signal Data Object Broker #1 that the prepare to commit was successful.

4. Data Object Broker #1 issues a commit to Data Object Brokers #2 and #3 and does its own local commit.
5. When Data Object Brokers #2 and #3 confirm their commits and release their contingency logs, the user is notified.

Fail Safe level-2 does not require a contingency log release after the commit request.

Multiple Data Object Brokers with an External Database Server

This scenario involves three Data Object Brokers and an external database server:



Commit Cycle for Three Data Object Brokers with External Database Server

At transaction end, the sequence of events is as follows:

1. Data Object Broker #1 issues a prepare to commit request to Data Object Brokers #2 and #3 and writes its intent list to the contingency log.
2. Data Object Broker #3 responds that the prepare to commit was successful.
3. Data Object Broker #2 replies that it can handle only Fail Safe level-1 requests.

4. Data Object Broker #1 sends another message to Data Object Broker #2 telling it to perform a Fail Safe level-1 commit.
5. When done, Data Object Broker #2 sends a success message back to #1, which completes the commits in Data Object Brokers #1 and #3.

Commit Behavior Across Multiple Data Object Brokers

The following table shows the commit-related actions for the more common combinations of TDS and Fail Safe level settings across two service providers:

TDS	Transaction updated: Service Provider #1	Service Provider #2	Action
Y	Y, FSlevel = 0	N	1. Commit service provider #1. 2. Commit TDS.
Y	Y, FSlevel = 1	N	1. Write TDS updates to contingency log. 2. Commit service provider #1 including updates to the transaction database or equivalent. [a] 3. Commit TDS.
N	Y, FSlevel = 1	N	Commit service provider #1.
Y	Y, FSlevel = 1	Y, FSlevel = 1	Commit is rejected because a single transaction is not allowed to update more than one service provider with FSlevel = 1.
Y	Y, FSlevel = 1	Y, FSlevel = 0	Commit is rejected because you cannot mix FSlevel = 0 with FSlevel > 0 in the same transaction.
N	Y, FSlevel = 1	Y, FSlevel = 0	Commit is rejected.
Y	Y, FSlevel = 1	Y, FSlevel = 2	1. Write intent to contingency log. 2. Issue prepare to commit to service provider #2. 3. Wait for response. 4. Issue Fail Safe level 1 commit to provider #1. Wait for response. 6. Commit local intent and issue commit to service provider #2.
Y	Y, FSlevel = 2	Y, FSlevel = 2	1. Prepare to commit issued for all three. 2. All three are committed.

a. When the service provider is another Data Object Broker, the contingency log provides an equivalent function to the transaction database for Fail Safe level-1 commits. This database is identified by the TRXDB server parameter. Refer to Transaction Database on page 20 for information on TRXDB.

Chapter 4 Understanding Journal Processing

This chapter describes journal processing for TIBCO Object Service Broker.

Topics

Introduction to Journal Processing

Definition

TIBCO Object Service Broker journals are data sets that provide a record of all modified page images. Journals are essential to the TIBCO Object Service Broker continuous backup and recovery procedures.

Overview

This chapter explains how TIBCO Object Service Broker:

- Spins (offloads) a journal
- Switches between journals

Journal backups can be merged together and used to refresh your current system backup. Batch utilities that update segments can also journal the updated page images. These images are the equivalent of those produced online by the Data Object Broker in the form of journal SPINs and can also be included in a backup procedure.

For information about developing a continuous backup procedure that meets your unique backup and recovery requirements, read this chapter and refer to Using the Continuous Backup Approach on page 48.

See Also *TIBCO Object Service Broker for z/OS Utilities* for information on the utilities that produce journals.

Spinning the Journals

When a journal data set becomes full, the Data Object Broker automatically switches to another one. The Data Object Broker still continues processing transactions during this switch. The journal that is full is then spun—emptied of its contents—and made available for use again. You can customize how a journal spin is executed.

Types of Journal Spins

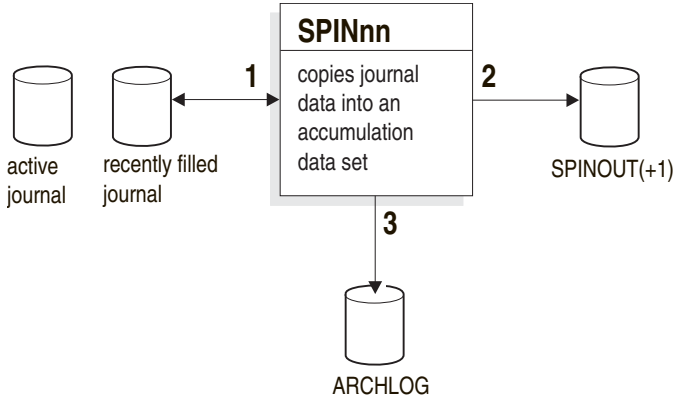
The Data Object Broker executes a journal spin in one of two ways:

- As a batch job
- As a started task

These approaches are outlined in Spinning Journals Using Batch Jobs on page 31 and Spinning Journals Using Started Tasks on page 32.

Journal Spin Process

The journal spin process is as follows:



The description of the steps shown in this diagram follows:

1. The contents of the full journal are read by the S6BSPJEX (Journal Data Extraction) utility.
2. The offloaded journal images are stored in accumulation files that are generation data sets (SPINOUT(+1)).
3. Spin and merge historical information is stored in ARCHLOG to help in the recovery process.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* and *TIBCO Object Service Broker for z/OS Utilities* for more information about journal processing.

TIBCO Object Service Broker for z/OS Utilities for information on the S6BSPJEX utility, the S6BTLFAL (Format ARCHLOG) utility, and the S6BTLPAL (Print ARCHLOG Information) utility.

Spinning Journals Using Batch Jobs

By default, journal spins are executed as batch jobs and are generally used by those who are not authorized to use started tasks.

Recommendation

If you decide to execute your journal spins as batch jobs, it is recommended that you use a cataloged procedure for the spin JCL. There are two benefits to using cataloged procedures:

- A minimum number of JCL statements is stored in memory.
- If you have multiple TIBCO Object Service Broker systems, you can keep all your journal spin procedures in one central procedure library.

Customization

Complete the following steps:

1. Specify the data set name from which spin jobs are submitted by entering the name as a value in the SPINDSNAME Data Object Broker parameter.

   It must be an existing partitioned data set containing member names set by the SPINMEMBER parameter.

2. Specify a separate member name for each of your journals in the SPINMEMBER Data Object Broker parameter.

3. Customize each spin member so that it points to its corresponding data set.

   JCL for journals 1 and 2 is located in members SPIN01, SPIN02, and others of the JCL data set.

4. Set the TDS and MDL S6BSPJEX parameters.

By default, the SPINOPTION Data Object Broker parameter is set to `JOB`. When the Data Object Broker starts up, it reads and stores in memory the spin JCL from a data set specified in the SPINDSNAME parameter and members specified in the SPINMEMBER parameter. When a spin is triggered, the saved spin JCL is submitted to the system by the internal reader.

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

TIBCO Object Service Broker for z/OS Utilities for more information about S6BSPJEX and its required parameters.

Spinning Journals Using Started Tasks

Why Use this Approach?

If started task programs are available to you, this approach is recommended for submitting journal spins because the spin can always execute immediately.

If you decide to use the started task approach and want to offload your journal directly to tape, first ensure that started tasks can issue tape mounts at your site.

Procedure

Complete the following steps:

1. Change the value of the SPINOPTION Data Object Broker parameter from JOB to STC.
2. Specify the name of your started task procedure in the SPINMEMBER Data Object Broker parameter.

   Member names set by the SPINMEMBER parameter must be in the system PROCLIB.
3. Customize the SPINxx and SPINSTC members in the JCL data set with OSEMOD to reflect your installation's naming convention. If you are using a started task (SPINOPTION=STC) rather than jobs (SPINOPTION=JOB) for your spins, you must copy the sample SPINSTC member into a data set that is part of your z/OS system PROCLIB concatenation. z/OS will search this concatenation when the Data Object Broker requests a started task journal spin.
4. Set the TDS and MDL S6BSPJEX parameters.

Journal Spin Start Command

This illustrates the journal spin start command issued by the Data Object Broker:

```
START SPINSTC,JRNLDSN=journal_data_set_name
```

In this example:

SPINSTC	The name of the journal spin started task procedure that is being invoked. You must use the sample SPINSTC member in the JCL data set distributed with TIBCO Object Service Broker, and modify it to specify a symbolic JRNLDSN parameter on the EXEC statement.
journal_data_set_name	The name of the journal data set that is being spun.

When the Data Object Broker issues the **START** command to invoke spin processing, the journal data set name specified in the command line is passed to the started task and the appropriate journal data set is offloaded.

See Also

TIBCO Object Service Broker Parameters for more information about these parameters.

TIBCO Object Service Broker for z/OS Installing and Operating for more information about operator commands.

TIBCO Object Service Broker for z/OS Utilities for more information about S6BSPJEX and its required parameters.

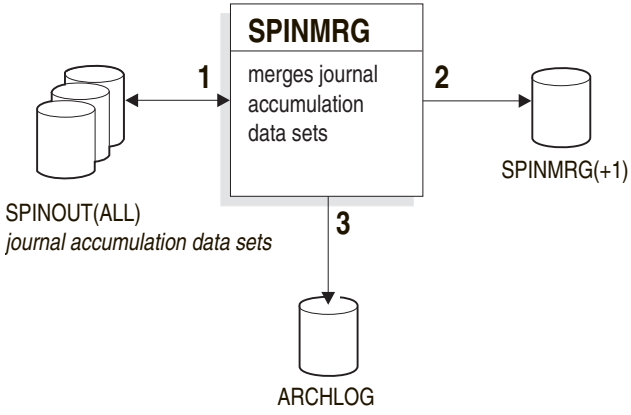
Merging Journal Data in Accumulation Data Sets

Merging Behavior

After the journal offload job step (S6BSPJEX), the SPIN procedure invokes the S6BSPDSN (Determine Number of GDGs) utility to determine how many generations exist in the journal accumulation generation data group. If the spin limit is exceeded, as specified by the $SPINLIM$ OSEMOD installation variable, the SPIN procedure submits the SPINMRG job, using IEBGENER, that merges all existing journal accumulation files into a single generation.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for details on the $SPINLIM$ OSEMOD installation variable.

Merging Data in the Accumulation Data Sets



SPINMRG Functionality

The SPINMRG job does the following:

1. Reads all journal accumulation data sets (SPINOUT(ALL)).
2. Creates a new journal accumulation data set generation (SPINMRG(+1)).
3. Records details of the journal accumulation activity in the log file (ARCHLOG).

Adjusting the Spin Limit

You can adjust the threshold at which point journal accumulation is triggered according to the following criteria:

- DASD availability. For example, to reduce the amount of disk space taken up by journal accumulation data sets, reduce the threshold number accordingly.
- The importance of the data and your requirements for immediate recovery. For example, if your data is extremely critical, you can merge your journal accumulation files together after every generation is written, and then merge your master accumulation with your latest backup several times a day.

You can choose from two merge options:

- Refer to Using the Continuous Backup Approach on page 48.
- Refer to Point in Time Recovery on page 102.

Determining Size and Number of Journals

Why Should Size and Number of Journals be Adjusted?

The number of journals and the size of journal data sets directly influence the time between spin and merge jobs.

You have control over the number and size of journal data sets. Journal data sets must be large enough so that they do not fill too often during the day; however, they must also be small enough so that the contents of a full journal can be saved in a reasonable length of time before the other journals also fill up.

You set the number of journal data sets with the ACBS parameter of the DB macro. You can define from 2 to 255 journal data sets to this parameter.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for more information about the DB macro.

TIBCO Object Service Broker Application Administion for information on how to make changes to journal data set sizes, including re-allocating and re-initializing journal data sets.

Switching Journals

Quiesced System Restrictions

If no journal is available to replace the full active journal, the Data Object Broker quiesces to protect the integrity of the database, and this message appears on the operator console:

```
S6BKX006A - ALL JOURNALS FULL - SYSTEM QUIESCED FOR COMMITS
```

While the system is quiesced, the Data Object Broker can process queries only from users already logged in to the system. It cannot process updates and does not allow new users to log in. The S6BTLADM (Administration Menu) utility can be started and used while the Data Object Broker is in quiesce mode. For more information about this condition, refer to Chapter 9, Errors in Journal Spinning, on page 117.

Releasing a Journal Manually

If the journal is ready for use and the spin job ended abnormally for another reason, you can reactivate the spin process.

An unsuccessfully offloaded journal cannot be brought back online. To reactivate a spun journal, enter the following command from the z/OS operator console:

```
Modify dob_jobname,Journalon=number
```

where:

- **`Modify is t`**he z/OS operator command
- *dob_jobname* is the name of the batch job or started task running the Data Object Broker
- **`Journalon`** is the TIBCO Object Service Broker operator command
- *number* identifies the number of the journal to bring online

If the journal against which you activate a spin is empty, the **Date/Time stamp** field of the ARCHLOG fills with blanks.

Turning Off Journal Processing for Selected Segments

Why Turn Off Journal Processing?

For selected segments that contain easily re-creatable data for which data recovery is not an issue, you can turn off journal processing for those segments.

While turning off journal processing has the advantage of saving valuable system resources, it is important to note that when journaling is turned off for a segment, that segment can no longer participate in a continuous backup strategy (even if journaling is re-enabled). If you want to re-establish continuous backup for the segment, you must first generate a new master copy of the segment.

Determining Journaling Setting

When a segment first comes online, the JOURNAL Data Object Broker parameter on a segment definition in member DBJCL of the JCL data set determines whether journal processing is initially on or off. In the following example, `JOURNAL=N` indicates that journal processing is disabled:

```
DB TYPE=PAGE,ACBS=3,NAME=$SEG0NAM$,ID=2,JOURNAL=N
```

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

Methods for Turning Journal Processing On and Off

The first of two ways to turn journal processing on and off for a specific segment is by using TIBCO Object Service Broker operator commands:

```
F dob_jobname,Dbjrnlon=segname or segnumber
F dob_jobname,Dbjrnloff=segname or segnumber
```

where:

`F` (or **`Modify`**)	The z/OS operator command.
dob_jobname	The name of the batch job or the system task name of the Data Object Broker.
`Dbjrnlon` or **`Dbjrnloff`**	The TIBCO Object Service Broker operator command.

segname	The name of a segment that is currently offline.
segnumber	The number of a segment that is currently offline.

Alternate Method

You can also use the S6BTLADM (Administration Menu) utility to turn journal processing on and off.

See Also *TIBCO Object Service Broker for z/OS Installing and Operating* for more information on the S6BTLADM utility and on operator commands.

Chapter 5 Backing Up Your System

This chapter describes different options for backing up your system.

Topics

Overview

TIBCO Object Service Broker Backup Utilities

TIBCO Object Service Broker provides you with a number of utilities so that you can build a customized backup system from the sample JCL provided at installation. The list of the TIBCO Object Service Broker backup utilities follows:

Backup Requirement	TIBCO Object Service Broker Utility
Back up page data sets	S6BTLBPS (Back Up Page Data Sets) or S6BTLUPS (Unload a Page Data Set to Backup).
Validate your backup	S6BBRPTR (Batch Pointer Check).

After a backup, it is recommended that you run the S6BBRPTR (Batch Pointer Check) utility.

- IDCAMS cannot be used to back up or copy TIBCO Object Service Broker page data sets because it changes the placement of records within the VSAM control interval. This invalidates internal TIBCO Object Service Broker index pointers.
- If the backup file created by the S6BTLBPS (Back Up Page Data Sets) utility is transferred to Windows or Solaris with FTP, the transfer must be in binary mode with the z/OS FTP LOCSITE subcommand with NORDW parameter specified.

See Also *TIBCO Object Service Broker for z/OS Utilities* for more information about individual utilities.

Planning Your System Backup

TIBCO Object Service Broker provides you with the flexibility to meet the backup and recovery requirements of your site whether you need regularly scheduled full system backups or a continuous backup approach that you can use to stay fully operational 24 hours a day, 7 days a week. Your TIBCO Support representative can help you build a backup and recovery plan that integrates one or both of these approaches to suit your requirements.

Full System Backup Approach

You must perform a full system backup when you first install TIBCO Object Service Broker. Depending upon your system availability requirements, you can also use this full system backup methodology as part of your regularly scheduled backup procedures. There are two ways you can perform a traditional full system backup:

Method	Refer to page…
Using TIBCO Object Service Broker Backup Utilities (and BACKUP JCL).	44
Using Full System Backup Products, such as DF/DSS or FDR, which creates a physical copy of the volume and saves it to tape.	54

Continuous Backup Approach

The continuous backup process enables you to maintain 24-hour operations and ensure full system recoverability at all times. Using the continuous backup approach, you never have to shut TIBCO Object Service Broker down for a routine backup. The continuous backup combines the most recent backups with the updates written out to the journal data sets to produce a complete and current backup copy.

Topic	Refer to page…
Using the Continuous Backup Approach.	48
Sample Continuous Backup Implementation.	52

Date and Time on Journal Pages

In the current release, all pagestore headers and journal records store the date and time in UTC time. That is, the time of the hardware clock without the local offset.

This makes the journaling process immune to spring and autumn time changes.

Using TIBCO Object Service Broker Backup Utilities (and BACKUP JCL)

Advantages to Using Backup Utilities

A sample BACKUP JCL is located in the JCL data set, distributed with TIBCO Object Service Broker. There are several key advantages to using this backup method over other backup tools:

- You can use the BACKUP JCL to back up the entire system or individual segments. You can back up any segment other than segment 0 by varying the segment offline, backing it up, and varying it back online.

  To back up segment 0, you must shut TIBCO Object Service Broker down.
- The backup process copies only those pages that actually contain data and also compresses the data as much as possible before writing it to tape.
- The backup can be integrated with journals created at a later time to produce an up-to-date backup.

Using Backup Utilities on Offline Segments

If you use the following offline utilities without batch journaling, you must refresh your latest backup with a complete backup of the processed segment to maintain the integrity of your backup data. These utilities update the segment offline and do not normally perform journaling:

- 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

See Also *TIBCO Object Service Broker for z/OS Utilities* for information about the individual utilities.

Journaling Offline

Utilities such as S6BBRCLR, S6BBRIAL, S6BBRPGC, S6BBRSET, S6BBRSIX, and S6BBRTBL have a facility for offline journaling. To invoke this facility, specify JOURNAL=Y as a parameter to the utility.

You must also add a JOURNAL DD statement to your JCL pointing to a VSAM data set. For example:

```
//CLEAR    EXEC PGM=S6BBRCLR,PARM=('BROWSE=N,SEGMENT=03',
//              'PSTABLE=your_table_to_clear,JOURNAL=Y')
//JOURNAL    DD DISP=OLD,DSN=your.offline.JOURNAL
```

Sample JCL is available in the BATJRNL member of the JCL data set that shows running the S6BBRCLR utility after allocating the journal.

Offline Journaling Constraints

Although offline journaling makes taking a full backup of the segment after running an offline update utility unnecessary, use it with caution:

- If the utility fails, combine the journal images with those produced by any utilities used in the recovery of the failure. Otherwise, partial updated images can appear in the continuous backup process, resulting in a corrupted backup that is unsuitable for recovery. Although running the S6BBRPTR (Batch Pointer Check) utility finds any physical pointer inconsistencies, the best strategy is to take all precautions to avoid them.
- The S6BBRPTR utility does not validate the logical consistency of the data; for example, an index can show the highest key on a data page different from the actual value on the data page.
- The journal data set should be allocated but not formatted. When the batch utility has built the journal, use IDCAMS to unload the records into a sequential data set for inclusion in the journal merge process.

Since the offline journaling facility has no provision for spinning journals, you must ensure that you have adequate space for the offline journal. If the journal fills up, a U0117 abend results and the batch job aborts. For more information on this abend code, refer to *TIBCO Object Service Broker Messages With Identifiers*.

See Also *TIBCO Object Service Broker for z/OS Utilities* for information on the utilities that produce journals and for specifics about batch journaling.

Backing Up a Segment

The following procedure shows how to create a backup of a selected segment using the supplied BACKUP JCL:

1. If you are backing up segment 0, or any segment defined as a system segment (according to the DBDLIB parameters), shut down the Data Object Broker by issuing a TIBCO Object Service Broker operator command.

   If you are backing up another segment, you can go directly to step 2.

   The command to shut down the Data Object Broker is:

   ```
   F dob_jobname,Shutdown
   ```

   You must ensure that no updates are pending for any segment being backed up. Make sure that any vary offline has completed successfully. If the Data Object Broker has been terminated, check that the shutdown completed normally. If not, restart the Data Object Broker and shut it down again. Perform a backup only after a normal shutdown.

2. Vary the segment offline.

   You can vary a segment offline from the Administration menu or you can issue a TIBCO Object Service Broker operator command. The command to vary a segment offline is:

   ```
   F dob_jobname,Dboffline=segmentname or segmentnumber
   ```

   where:

F	The z/OS MODIFY operator command.
dob_jobname	The name of the batch job or started task under which the Data Object Broker is running.
Dboffline	The TIBCO Object Service Broker operator command to vary a segment offline.
segmentname	The name of the segment you want to vary offline.
segmentnumber	The number of the segment you want to vary offline.

   Examples:

   ```
   F dob_jobname,Dboffline=3
   F dob_jobname,Dboffline=sales
   ```

3. Edit the BACKUP member in the JCL data set.

   Follow the embedded documentation to set the proper input and output parameters. Complete the following steps:

   a. Give values to variables that begin and end with a dollar sign ($).

      TIBCO Object Service Broker for z/OS Installing and Operating describes how to use the **OSEMOD** edit macro to replace these variables with the correct values.

   b. Specify the correct input segment number and the output data set name.

      The sample JCL targets the backup to disk on DASD.

   c. After you tailor this and other supplied JCL, make a backup copy of the JCL data set for your own disaster recovery purposes.

4. Save the modified file and submit it for execution.

   The BACKUP job calls the S6BTLBPS (Back Up Page Data Sets) utility to back up page data sets. This utility does the following:

   — Checks the DBDLIB to determine the number of data sets in the specified segment.

   — Sequentially outputs each used page to a file. Empty space within the page is truncated before the page is written, thus making the backup more efficient.

5. Vary the segment online or if you backed up segment 0, start up the Data Object Broker.

   Vary the segment online from the Administration menu or issue a TIBCO Object Service Broker operator command. The syntax to vary a segment online is:

   ```
   F dob_jobname,Dbonline=segmentname or segmentnumber
   ```

You can back up your data sets more quickly by running the S6BTLUPS (Unload a Page Data Set to Backup) utility as a separate job or started task for each data set in the segment.

Using the Continuous Backup Approach

Rather than take full or partial system backups at regular intervals, you can decide to follow the continuous backup approach and use your journals to update your latest backup. There are three tasks to the continuous backup procedure:

- *Merging Journal Accumulations, page 48*
- *Merging Journal Accumulations with Complete Backups, page 51*
- *Validating Your Backup, page 51*

The following sections explain these steps and suggest some implementation alternatives. The choices you make affect the procedures you follow regularly and the length of down time you experience if you ever must restore backed-up data.

Task A Merging Journal Accumulations

Depending upon the size of your journal data sets and the number of transactions you process, your journals can be spun many times during the day. When a journal spins, its contents are condensed and offloaded into a journal accumulation file.

Changing the Default Threshold

The systems administration and operations staff at your site decide how many journal accumulation files to keep on DASD before merging them together into one. The default threshold is 2. To change this default, edit the value of the $SPINLIM$ parameter in your spin procedure. The spin limit must not be greater than the limit specified in the installation parameter $JSRGDG$. For more information about $JSRGDG$, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

Sample JCL

The following example illustrates how you can use the standard SORT JCL to merge journal accumulations after every two journal spins. The JCL for this example is derived from member SPINMRG in the JCL data set.

The ARCHLOG is written out and retained to provide you with an audit trail of the accumulation process.

```
//$JOBNME$ JOB ($ACTCDE$),'MERGE OSB JOURNALS',
//          MSGCLASS=$MSGCLS$,CLASS=$PRCCLS$,TIME=60,
//          NOTIFY=$NOTIFY$,MSGLEVEL=(1,1),REGION=5M
//*
//* FUNCTION:  MERGE JOURNAL SPINOUT GDG'S INTO A SINGLE GDG
//*
//* CUSTOMIZATION: OSEMOD ISPF EDIT MACRO
//*
//* NOTES: JOURNAL SORTOUT DATA SET DEFAULTS SHOULD SUFFICE FOR A
//*         DEFAULT OF 10 GDG'S BEFORE MERGING.
//*
//*         MERGE SPUN JOURNALS INTO A SINGLE GDG
//*
//MERGE     EXEC PGM=SORT
//SORTIN    DD DSN=$HLQNONV$.$SLQ$.JOURNAL.SPINOUT,
//             DISP=(OLD,DELETE,KEEP)
//SORTOUT   DD DSN=$HLQNONV$.$SLQ$.JOURNAL.SPINMRG(+1),
//             DISP=(NEW,CATLG,DELETE),SPACE=(CYL,(100,20),RLSE),
//             UNIT=$INSTUNT$,
//             DCB=($HLQNONV$.$INSTVER$.MODEL.DSCB,
//             RECFM=VB,LRECL=4800,BLKSIZE=0)
//SORTWK01 DD UNIT=$INSTUNT$,SPACE=(CYL,(15,15))
//SORTWK02 DD UNIT=$INSTUNT$,SPACE=(CYL,(15,15))
//EXITLIB   DD DSN=$HLQNONV$.$INSTVER$.LOAD,
//             DISP=SHR
//TEMPLOG   DD UNIT=$INSTUNT$,SPACE=(TRK,(1,1))
//ARCHLOG   DD DSN=$HLQNONV$.$SLQ$.ARCHLOG.JRNL,
//             DISP=SHR
//SYSIN     DD DSN=$HLQNONV$.$INSTVER$.JCL(XSPINSRT),
//             DISP=SHR
//SYSOUT    DD SYSOUT=$SYSPRT$
//*
//*         SUBMIT CONTINUOUS BACKUP JOB
//*
//SPAWN     EXEC PGM=IEBGENER,
//          COND=(0,NE,MERGE)
//SYSPRINT DD SYSOUT=$SYSPRT$
//SYSUT1    DD DSN=$HLQNONV$.$INSTVER$.JCL(BKUPCON),
//             DISP=SHR
//SYSUT2    DD SYSOUT=(A,INTRDR)
//SYSIN     DD DUMMY
//
```

Sort Control Cards

This JCL uses standard sorting facilities and exit routines. The sorting procedure for merging journal accumulations reads in all existing journal accumulations (for example, as generation data sets within a GDG), eliminates old duplicate pages and outputs a master accumulation file to disk. If DASD is initially used for the GDG, the GDG base must be deleted and redefined if you switch to tape.

The following illustrates a typical sort control statement:

```
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)
<optional filtering INCLUDE statement>
RECORD TYPE=V
```

The following outline this sort control statement sorts the journal page images in a controlled order:

1. Ascending segment identification number (offset 5)
2. Ascending data set identification number (offset 7).

The data set number begins at zero and corresponds to the JCL data set as follows:

`<QUALIFIER>.<SEGMENTIDENT>.PAGE1`

3. Ascending page number.
4. Descending date of last update (offset 21)
5. Descending time of last update (offset 24)
6. Descending Checkpoint identification for last update (offset 30)
7. Descending transaction identification for last update (offset 27)

The MODS statement causes the sort to invoke special TIBCO Object Service Broker exits that select the most current page images for recovery.

To modify your sort control statement, you must know the format of the header information and use that to determine your SORT offsets. For more information, refer to Appendix A, Sort Control Manipulation, on page 121.

Point in Time Recovery Options

While the sample JCL uses sort exit S6BSPX35, an alternative exit, S6BSPU35, is available if you want to be able to recover segment pages using journal images to a specific point in time.

The point-in-time recovery option, enabled by the use of S6BSPU35, uses much more journal file space than recovery to the latest quiesce point that S6BSPX35 gives you. To allow point-in-time recovery, S6BSPU35 saves all journaled page images, unlike S6BSPX35, which saves only the latest version of each. For more information on point-in-time recovery, refer to Chapter 8, Recovery Procedures, on page 93.

If you opt for point-in-time recovery, consider using tape rather than DASD to economize on journal file space.

Task B Merging Journal Accumulations with Complete Backups

Using the continuous approach, you can merge your master accumulation file with your latest complete backup as frequently as your recovery needs require. In most cases, we recommend a daily refresh of your complete backup.

To refresh your latest backup, use the sort program to read in your journal accumulation file (or files) and your latest complete backup. The sort exit routines write information about the input and output records in an ARCHLOG and eliminate the older versions of duplicate pages. The output from this sort is a new system backup that includes all page updates recorded in the journal accumulation files.

Continuous backup and recovery works on fully completed (committed) transactions. Remember that transactions in the contingency log are not complete and are not part of the continuous backup and recovery process.

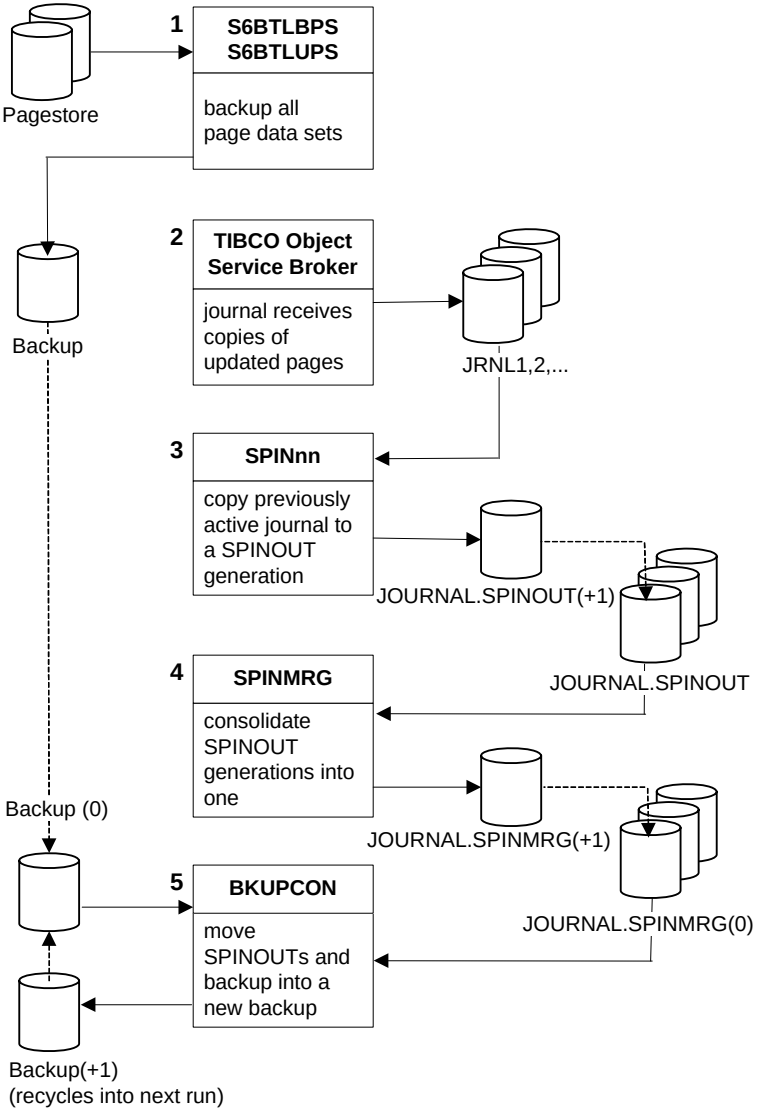
Task C Validating Your Backup

After you create your new backup, you must run the S6BBRPTR (Batch Pointer Check) utility against it to verify the accessibility of all pages in the backup. The S6BBRPTR utility is described in *TIBCO Object Service Broker for z/OS Utilities*.

Sample Continuous Backup Implementation

Overview of Continuous Backup Data Flow

The following diagram illustrates one possible continuous backup strategy. To develop a customized plan for your installation, consult your TIBCO Support representative.



Steps in the Continuous Backup Illustration

This diagram illustrates the following steps:

1. A Pagestore backup is created using TIBCO Object Service Broker backup utilities.
2. Images of all updated (Pagestore) pages are copied— journaled—to the active journal data set by the Data Object Broker. If you need to, set up a similar process for batch utility journaling.
3. When the active journal fills up, TIBCO Object Service Broker starts a spin job—named SPIN*nn*—to copy the contents of the journal data sets into the backup process.
4. Periodically, when the user-defined limit is reached on the number of JOURNAL.SPINOUT data sets, SPINMRG is automatically submitted to consolidate information into a new generation of the JOURNAL.SPINMRG data set. All generations of JOURNAL.SPINOUT are deleted.

   When a specific page appears more than once in the input SPINOUT data sets, a TIBCO Object Service Broker exit in this job's sort phase causes only the most recent version of the page to be written out. (If you are using point-in-time recovery, another exit, which retains all page images, is substituted.)
5. Having created a consolidated SPINMRG data set, consisting of the most recent images of updated pages, the previous SPINMRG job automatically submits BKUPCON. This updates the current Pagestore backup, creating a new generation of the BACKUP data set.

   The BKUPCON job takes the most recent JOURNAL.SPINMRG data set as input. This means that new SPINOUT generations or page images not spun from the current JOURNAL data sets—beyond those read by the latest BKUPCON— go into another backup generation at a later time. The output of BKUPCON is a new generation of the BACKUP generation data set. This data set contains a complete set of page images that can be used to recover a Pagestore using the TIBCO Object Service Broker restore utilities. You should always run the S6BBRPTR (Batch Pointer Check) utility against newly created backup data sets to ensure their integrity.

See Also *TIBCO Object Service Broker for z/OS Utilities* for information on the backup utilities and on batch journaling.

Using Full System Backup Products

While backup methods other than proposed with TIBCO Object Service Broker have the advantage of being relatively simple and fast, they are not space efficient. They save the entire Pagestore instead of just the used pages. After backing up your system using any of these methods, your only method of restoring the system is to overlay the existing system completely with the backup copy (refer to Chapter 8, Recovery Procedures, on page 93).

Usage

To create a backup of the system using one of these products, shut down TIBCO Object Service Broker and use the backup product as you normally would against other DASD.

IDCAMS cannot be used to back up or copy TIBCO Object Service Broker page data sets because it changes the placement of records within the VSAM control interval. This invalidates internal TIBCO Object Service Broker index pointers.

Backing Up and Restoring the RESOURCE File

Backing Up Online

Refer to *TIBCO Object Service Broker for z/OS Utilities* for information on backing up the resource file online using the S6BTLBRM (Resource Management Online Backup) utility.

Backing Up the RESOURCE File Offline

To back up the RESOURCE file definitions offline (peer connections, schedules, and so on), you can run from the JCL as part of your Data Object Broker JCL after shutdown.

```
//* SAMPLE JCL TO BACKUP RESOURCE FILE
//S1       EXEC PGM=IDCAMS,REGION=2048K
//RESOURCE DD DISP=SHR,DSN=$HLQVSAM$.$SLQ$.RESOURCE
//BACKUP   DD DISP=(,CATLG),DSN=$HLQVSAM$.$SLQ$.BKUP.RESOURCE,
//    DCB=(RECFM=VB,LRECL=132,BLKSIZE=0,DSORG=PS),
//    SPACE=(TRK,(5,5),RLSE),UNIT=SYSDA
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
    REPRO INFILE(RESOURCE) OUTFILE(BACKUP)
/*
```

Restoring the RESOURCE File

To restore the RESOURCE file offline, use the DELETE and DEFINE JCL, which is part of the shipped JCL in the S6A6POST member in the OSB.INSTALL data set:

```
//* SAMPLE JCL TO DELETE AND REALLOCATE RESOURCE FILE
//STEP8  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=$SYSPRT$
//SYSIN  DD   *
DELETE $HLQVSAM$.$SLQ$.RESOURCE
SET MAXCC=0
DEFINE CLUSTER                                          -
    (NAME($HLQVSAM$.$SLQ$.RESOURCE)                     -
    VOLUMES($VOLUM05$)                                  -
    INDEXED                                             -
    RECSZ(128,128)                                      -
    KEYS(19,0)                                          -
    FREESPACE(20,10)                                    -
    CYLINDERS(2,2) CISZ(2048))                          -
    DATA(NAME($HLQVSAM$.$SLQ$.RESOURCE.DATA))
    INDEX(NAME($HLQVSAM$.$SLQ$.RESOURCE.INDEX))
/*
```

After running the previous JCL, run the following JCL which us part of the shipped JCL in the RESTRSCE member in the JCL data set.

```
//* SAMPLE JCL TO RELOAD THE RESOURCE FILE FROM BACKUP
//S1       EXEC PGM=IDCAMS,REGION=2048K
//RESOURCE DD DISP=SHR,DSN=$HLQVSAM$.$SLQ$.RESOURCE
//BACKUP   DD DISP=SHR,DSN=$HLQVSAM$.$SLQ$.BKUP.RESOURCE
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
   REPRO INFILE(BACKUP) OUTFILE(RESOURCE) REPLACE
/*
```

Chapter 6 Managing Data Sets

This chapter describes how to manage your data sets.

Topics

Duplexing the Redolog for Recoverability

Purpose

A duplex copy of the redolog should be maintained as a safeguard against potential hardware failures experienced while TIBCO Object Service Broker is running.

Creating a Duplex Copy of the Redolog

To create a duplex copy of the redolog follow these steps:

1. Perform a clean shutdown of the Data Object Broker.
2. Specify that the redolog is to be duplexed in the DBGEN statement of your JCL:

   ```
   DB TYPE=REDOLOG,DUPLEX=$REDODUP$
   ```

   The JCL is located in member DBJCL of the JCL data set.
3. Change the parameter value for the $REDODUP$ OSEMOD symbolic from its default of N to Y.
4. Change the parameter value for the $REDOVDF$ OSEMOD symbolic from its default of SPLXREDO to DPLXREDO.

   These two symbolics are listed in member OSEMOD of the CLIST data set. For more information on customizing OSEMOD symbolics, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.
5. Submit member DBJCL.
6. If you want the Data Object Broker to stop processing as the result of an I/O error on either the primary or duplex data set, specify the following Data Object Broker startup parameter:

   ```
   DUPLEXLOGFAIL=END
   ```

   The default value is `CONTINUE`. This instructs the Data Object Broker to continue processing updates as long as there is at least one redolog. For detail information about this and other Data Object Broker parameters, refer to *TIBCO Object Service Broker Parameters*.
7. Allocate the duplex redolog data set by customizing member DPLXREDO in the CNTL data set.

The following gives sample IDCAMS controls cards to allocate a duplex redolog data set:

They contain OSEMOD symbolics. By default, a single redolog is created, as allocated in member SPLXREDO.

```
DELETE $HLQVSAM$.$SLQ$.REDOLOG
 DELETE $HLQVSAM$.$SLQ$.REDOLOG.DUPLEX
 SET MAXCC=0
 DEFINE CLUSTER                                                -
    (NAME($HLQVSAM$.$SLQ$.REDOLOG)                              -
     UNIQUE BUFSPC(8192)                                        -
     VOLUMES($VOLUM05$)                                         -
     NONINDEXED                                                 -
     RECSZ(200,4050)                                            -
     CYLINDERS(100,0) CISZ(4096))                               -
    DATA(NAME('$HLQVSAM$.$SLQ$.REDOLOG.DATA'))
 DEFINE CLUSTER                                                -
    (NAME($HLQVSAM$.$SLQ$.REDOLOG.DUPLEX)                       -
     UNIQUE BUFSPC(8192)                                        -
     VOLUMES($VOLUM06$)                                         -
     NONINDEXED                                                 -
     RECSZ(200,4050)                                            -
     CYLINDERS(100,0) CISZ(4096))                               -
    DATA(NAME('$HLQVSAM$.$SLQ$.REDOLOG.DATA.DUPLEX'))
```

8. Format the duplex redolog using the S6BTLFRL (Format Redolog) utility.

   The S6BTLFRL utility uses the DBDLIB to determine whether the data set is duplexed. If the data set is being duplexed for the first time, the S6BTLFRL utility issues the message S6BLT075E indicating that the primary redolog data set is not empty and that it bypasses formatting it. The duplex redolog data set is dynamically allocated and formatted. The duplex data set for the redolog has the suffix .DUPLEX appended to the name.

   For more information about the use of the S6BTLFRL utility, refer to *TIBCO Object Service Broker for z/OS Utilities*.

9. Delete, define and format both cache data sets.
10. Restart the Data Object Broker.

    The job log should indicate that the redolog data set is duplexed.

Considerations

- The duplex redolog must have the same size allocations as the primary redolog.

- The redolog and its duplex copy should reside on separate disk devices for maximum recoverability.
- When the redolog is duplexed, if either the primary or duplex data set is removed, TIBCO Object Service Broker cannot initialize until either the DBGEN is changed or the data set is redefined and reformatted.
- During recovery from a redolog data set, TIBCO Object Service Broker automatically recovers from the primary redolog unless an error is detected, in which case recovery switches to the duplex data set. For more information on TIBCO Object Service Broker recovery, refer to Chapter 8, Recovery Procedures, on page 93.

Moving Data Sets to Different DASD Devices

Purpose

With this procedure, you can move your Data Object Broker data sets from one type of DASD device to another, while maintaining data integrity and preventing loss of data.

Considerations

Before migrating a data set:

- Determine whether there are in-doubt transactions pending confirmation from external service providers. To do this, use the S6BTLADM (Administration Menu) utility. Clear all pending transactions prior to migrating to new DASD.

  Refer to *TIBCO Object Service Broker for z/OS Installing and Operating* for information about the Administration menu.

- For all the Data Object Broker system files including the Pagestore Segment 0 and the Audit Log segment, you must bring the Data Object Broker down with a normal, successful shutdown. If the shutdown terminates abnormally, restart and shut down the Data Object Broker until you get a normal, successful shutdown.

- For the user-data Pagestore segments (those segments other than Segment 0 and the Audit Log segment), you can simply vary offline the segments to be moved.

See Also

For definitions of the Data Object Broker files, refer to:

- For the resource file, *TIBCO Object Service Broker for z/OS Installing and Operating*.
- For the other files, TIBCO Object Service Broker Data Sets on page 7

Details by File

Data Set	Format/ Restore	Allocate with	Special Considerations
Cache	S6BTLFCA	IEFBR14	The Data Object Broker must be successfully shut down. The redolog must be reformatted.
Redolog	S6BTLFRL	IDCAMS	The Data Object Broker must be successfully shut down. The cache data sets must be reformatted.
Journal	S6BTLFJR	IDCAMS	The Data Object Broker must be successfully shut down.
Contingency log	S6BTLFCL	IEFBR14	The Data Object Broker must be successfully shut down.
Resource	IDCAMS	IDCAMS	Does not require formatting if recovery of data set from backup is performed.
Pagestore	S6BTLFPS	IDCAMS	There must be no pointer check errors. Use only provided backup and restore utilities. The Data Object Broker must be successfully shut down (Segment 0 and ACCESSLOG segment) or segments must be varied offline (user-data segments).

System Files

If the Data Object Broker terminated successfully with all pending transactions cleared, you can do the following, according to the detail that follows:

- You can delete the CACHEx, REDOLOG, JRNLx, and contingency log data sets, then reallocate and reformat them on a new device.
- You can back up the RESOURCE data set and the data sets for the Pagestore segment 0 and ACCESSLOG segment and restore them to the new device.

If the Data Object Broker did not shutdown successfully, do not attempt to migrate any data sets to new DASD. If you do, data integrity problems can occur.

Cache Data Sets

Type	Direct Access Non-VSAM (DSORG=DA)
Number/Name	2, $HLQNONV$.SLQ.CACHE{1 | 2}
Format with	The S6BTLFCA (Format Cache Data Sets) utility
Migration	When the Data Object Broker is shutdown normally, you can delete and reallocate these data sets with IEFBR14. Re-initialize the data sets using the S6BTLFCA utility. You must also delete, reallocate, and reformat the redolog.

Redolog

Type	VSAM, ESDS
Number/Name	1, $HLQVSAM$.SLQ.REDOLOG
Format with	The S6BTLFRL (Format Redolog) utility
Migration	When the Data Object Broker is shutdown normally, you can delete and reallocate a new redolog data set. Use the S6BTLFRL utility to reformat the new data set. You must also delete, reallocate, and reformat the cache data sets.

Journal Data Sets

Type	VSAM, ESDS
Number/Name	2 to 255, $HLQVSAM$.SLQ.JRNL{1 | 2 | ... | 255}
Format with	The S6BTLFJR (Format Journal Data Sets) utility
Migration	When the Data Object Broker is shutdown normally, you can run SPIN*xx* jobs to copy any data held within the journal data sets. At this point, you can delete the old journal data sets and allocate new data sets using IDCAMS. Alternatively, you can perform new backups of all segment data sets and simply delete and redefine the new journals. Format the new journal data sets with the S6BTLFJR utility.

Contingency Log Data Set

Type	Direct Access non-VSAM (DSORG=DA)
Number/Name	1, $HLQNONV$.SLQ.REDOLOG.PENDING
Format with	The S6BTLFCL (Format Contingency Log) utility
Migration	Use the S6BTLADM (Administration Menu) utility to determine if in-doubt transactions are pending confirmation from external service providers. Clear all pending transactions and shut down the Data Object Broker (ensuring a successful shutdown) prior to migrating to the new device. Refer to *TIBCO Object Service Broker for z/OS Installing and Operating* for information about the Administration menu.

Resource

Type	VSAM, KSDS
Number/Name	1, $HLQVSAM$.SLQ.RESOURCE
Backup with	The S6BTLBRM (Resource Management Online Backup) utility (if the Data Object Broker is running); IDCAMS (if the Data Object Broker is down)
Restore with	IDCAMS (Data Object Broker must be down)
Format with	IDCAMS (see JCL member S6A6POST in the OSB.INSTALL data set)

Migration	You can use the S6BTLBRM utility to backup the contents of the online resource file to a flat file. If the Data Object Broker is not running, you can use the IDCAMS REPRO command to copy the contents to a flat file. In either case, the flat file should be RECFM=FB with a LRECL=128. You can use the IDCAMS DELETE and DEFINE cluster commands in the JCL sample (S6A6POST in the OSB.INSTALL data set) as guides to deleting and reallocating this file to another volume. Run the equivalent of JCL in the RESTRSCE member in the JCL data set to restore your resource data set from your previous backup, which was taken using either the S6BTLBRM utility or IDCAMS.

Pagestore System Segments

Type	VSAM, ESDS
Number/Name	2: Segment 0 and the Audit Log segment $HLQVSAM$.$SEGNAM$.PAGE#
Backup with	The S6BTLUPS (Unload Page Data Set to Backup) utility, which you can run in parallel under many jobs to speed up backups. The S6BTLBPS (Back Up Page Data Sets) utility
Restore with	The S6BTLRPS (Restore TDS Segment) utility
Format with	The S6BTLFPS (Format Page Data Sets) utility
Migration	When the Data Object Broker is shutdown normally, you can perform the procedure in Pagestore Migration on page 66.

Pagestore User-Data Segments

Type	VSAM, ESDS
Number/Name	User selectable, $HLQVSAM$.$SEGNAM$.PAGE#

Backup with	The S6BTLUPS (Unload Page Data Set to Backup) utility, which you can run in parallel under many jobs to speed up backups. The S6BTLBPS (Back Up Page Data Sets) utility
Restore with	The S6BTLRPS (Restore TDS Segment) utility
Format with	The S6BTLFPS (Format Page Data Sets) utility
Migration	After varying offline the segments you want to move or shutting down the Data Object Broker normally, you can perform the procedure that follows.

Pagestore Migration

1. Take a backup of the segment whose data files you want to migrate, using one of the S6BTLBPS (Back Up Page Data Sets) utility or the S6BTLUPS (Unload a Page Data Set to Backup) utility.
2. If using the S6BTLUPS utility, consolidate the backup images using the equivalent of a SORT/MERGE step.

   You must do this because each backup invocation can create a separate backup data set. You must consolidate these data sets into a standard, single, backup data set sorted by segment, page number, and so on. See step #4. in Steps in the Continuous Backup Illustration on page 53 for details.
3. Run the S6BBRPTR (Batch Pointer Check) utility against the backup.
4. If problems are reported by the S6BBRPTR utility, correct the errors before proceeding. Take another backup and run the S6BBRPTR utility again. Validation of the integrity of your data files, can require running the S6BBRPTR utility more than once.
5. Delete the old data files and reallocate the new ones.
6. Format the new page data files using the S6BTLFPS (Format TDS Page Data Set) utility.
7. Restore the data sets using the previous clean backup from step #1. or step #4. using the S6BTLRPS (Restore TDS Segment) utility.

Utilities such as IDCAMS are incompatible with the structure of the Pagestore data sets. You must not use them to move or reformat the page data sets. These data sets cannot be processed at the VSAM logical record level.

Chapter 7 Maintaining Segments Using Third-Party Products

This chapter describes how to maintain TIBCO Object Service Broker segments using third-party products.

Topics

Overview

This chapter explains how to maintain TIBCO Object Service Broker segments using third-party products while the segments are still online and accessible to the TIBCO Object Service Broker system. This facility interfaces with:

- The Backup While Open (BWO) feature of IBM DFSMSdss
- The Softek TDMF, Mainframe Edition product with the chargeable feature Offline Volume Access (OVA) enabled

This chapter assumes that you have either of these products and basic experience using either of them to perform point-in-time copies of data.

TIBCO Object Service Broker Commands Available

TIBCO Object Service Broker has a set of commands that facilitate synchronization using TIBCO Object Service Broker with either the BWO feature of DFSMSdss or Softek TDMF:

1. The FREEZE command flushes all updated pages from storage to the page data sets. If necessary, checkpoints are taken and flushed. When all the I/O activity for this operation is complete, an operator message appears. While the TIBCO Object Service Broker system is frozen, transactions proceed as normal, but no write I/O is performed to the database and journals. If the BWO feature is used, the page and journal data sets that are under the control of DFSMSdfp are set into BWO status. This enables DFSMSdss to issue dumps and copies that can be taken of them.
2. The UNFREEZE command allows I/O activity to database and journals to resume. If BWO support is used, prior to resuming I/O, the page and journal data sets that are under the control of DFSMSdfp are reset to normal access thus disabling concurrent access.
3. The STATUS command displays the current state of the system. There are four possible states:

   — Active

   — Frozen

   — In the process of freezing

   — In the process of unfreezing

   If the system is in the process of freezing and there are long running tasks active, an information message appears for each such task.

4. If BWO support is used, the BWOSTATUS command displays the current settings of the BWO flags for each page and journal data set and the return and reason codes from the last action performed on the data set.

Required Facilities and Process for Backup While Open

What is Backup While Open (BWO)?

Backup while open serialization, which is a supported feature of DFSMSdss, allows DFSMSdss to perform backup of data sets that are open for update for long periods of time. It can perform a logical data set dump of these data sets even if another application has them serialized. Backup while open is a better method than using SHARE or TOLERATE(ENQFAILURE) for dumping TIBCO Object Service Broker VSAM data sets that are in use and open for update. When you dump data sets that are designated by TIBCO Object Service Broker as eligible for backup while open processing, data integrity is maintained through serialization interactions between TIBCO Object Service Broker, VSAM record management, DFSMSdfp, and DFSMSdss.

Prerequisites

To use the BWO feature of DFSMSdss, your system must have the following in place:

1. IBM DFSMSdss installed
2. The ability to allocate TIBCO Object Service Broker page data sets under the control of DFSMS on DASD that supports the CONCURRENT copy feature

Process of Using the BWO Feature of DFSMSdss

The general process of using the BWO feature of DFSMSdss to maintain TIBCO Object Service Broker segments while maintaining access to the TIBCO Object Service Broker data is as follows:

1. To prevent access by a TIBCO Object Service Broker offline utility, ensure the segments to be maintained are ONLINE to the TIBCO Object Service Broker system.
2. Issue the FREEZE TIBCO Object Service Broker command.
3. When the TIBCO Object Service Broker system is frozen, run the DFSMSdss jobs to copy or dump the desired data sets. The DFSMSdss jobs should specify the Concurrent and NotifyCC options.

4. Use the UNFREEZE TIBCO Object Service Broker command to get TIBCO Object Service Broker to continue processing.

You can use the UNFREEZE command when the ADR734I messages have been issued for all the data sets being dumped. It is not required to wait until the backup jobs run to completion.

5. Allow the DFSMSdss jobs to run to completion.

Examples of Maintenance Steps Using the BWO Feature of DFSMSdss

Step 1: Set up DFSMSdss jobs to perform concurrent dumps of the desired TIBCO Object Service Broker data sets

In the example and, the page data sets backing segment 2 are dumped:

```
//USRDFR#1 JOB (1),'DFDSS DUMP',REGION=4M,MSGCLASS=Y
//STEP1    EXEC  PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//TAPE1    DD DSN=USR10.DSS.SEG2.PAGE1,DISP=(,CATLG),
//            UNIT=S6BPOOL,SPACE=(CYL,(200,50))
//TAPE2    DD DSN=USR10.DSS.SEG2.PAGE2,DISP=(,CATLG),
//            UNIT=S6BPOOL,SPACE=(CYL,(200,50))
//TAPE3    DD DSN=USR10.DSS.SEG2.PAGE3,DISP=(,CATLG),
//            UNIT=S6BPOOL,SPACE=(CYL,(200,50))
//DASD     DD UNIT=S6BPOOL,SPACE=(CYL,(200,50))
//SYSIN    DD *
 PARALLEL
 DUMP DATASET(INCLUDE(USR10.R50.CR2.SEG2.PAGE1))  -
           OUTDDNAME(TAPE1) CONCURRENT NOTIFYCC
 DUMP DATASET(INCLUDE(USR10.R50.CR2.SEG2.PAGE2))  -
           OUTDDNAME(TAPE2) CONCURRENT NOTIFYCC
 DUMP DATASET(INCLUDE(USR10.R50.CR2.SEG2.PAGE3))  -
           OUTDDNAME(TAPE3) CONCURRENT NOTIFYCC
//
```

Step 2: Freeze the TIBCO Object Service Broker system

Issue the TIBCO Object Service Broker FREEZE command. A sample of the system log output at the time of the freeze is shown at follows,:

```
10.50.12 JOB08525  S6BKP005L-USRA MODIFY USR50CR2,FREEZE
10.50.12 JOB08525  S6BKP006I-USRA MODIFY COMMAND ACCEPTED
10.50.12 JOB08525  S6BKX051L-USRA START CHECKPOINT -      186
10.50.12 JOB08525  S6BKX069I-USRA SYSTEM IS FROZEN: USR50CR2
```

Step 3: Run the DFSMSdss job

Run the DFSMSdss job described previously. Here is sample output from this job:

```
J E S 2  J O B  L O G  --  S Y S T E M  R 6 E 0
--  N O D E  H R N M V S
10.54.47 JOB08559 ---- THURSDAY,  12 DEC 2002 ----
10.54.47 JOB08559  IRR010I  USERID USR11    IS ASSIGNED TO THIS JOB.
10.54.47 JOB08559  ICH70001I USR11 LAST ACCESS AT 10:54:03 ON THURSDAY,DEC 12,2002
10.54.47 JOB08559  $HASP373 USRDFR#1 STARTED - INIT 5    - CLASS A - SYS R6E0
10.54.47 JOB08559  IEF403I USRDFR#1 - STARTED - TIME=10.54.47
10.54.54 JOB08559  ADR767I (003)-T0MI (02), 2002.346 10:54:54 CONCURRENT COPY  421
   421             INITIALIZATION SUCCESSFUL FOR DATA SET USR10.R50.CR2.SEG2.PAGE2 IN
   421             CATALOG ICFCAT.R6E0P1. SERIALIZATION IS RELEASED.
10.54.54 JOB08559  ADR801I (003)-DTDSC(01), DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA 422
   422             SETS WERE SELECTED: 0 FAILED SERIALIZATION AND 0 FAILED FOR OTHER
   422             REASONS.
10.54.54 JOB08559  ADR734I (003)-DTDSC(01), 2002.346 10:54:54 CONCURRENT COPY 423
   423             INITIALIZATION SUCCESSFUL FOR 1 OF 1 SELECTED DATA SETS. SERIALIZATION
   423             FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT. THE INTERMEDIATE RETURN
   423             CODE IS 0004.
10.55.07 JOB08559  ADR767I (002)-T0MI (02), 2002.346 10:55:07 CONCURRENT COPY  427
10.55.07 JOB08559  ADR767I (004)-T0MI (02), 2002.346 10:55:07 CONCURRENT COPY  428
   427             INITIALIZATION SUCCESSFUL FOR DATA SET USR10.R50.CR2.SEG2.PAGE1 IN
   427             CATALOG ICFCAT.R6E0P1. SERIALIZATION IS RELEASED.
10.55.07 JOB08559  ADR801I (002)-DTDSC(01), DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA 429
   429             SETS WERE SELECTED: 0 FAILED SERIALIZATION AND 0 FAILED FOR OTHER
   429             REASONS.
       428 INITIALIZATION SUCCESSFUL FOR DATA SET USR10.R50.CR2.SEG2.PAGE3 IN
   428             CATALOG ICFCAT.R6E0P1. SERIALIZATION IS RELEASED.
10.55.07 JOB08559  ADR801I (004)-DTDSC(01), DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA 430
10.55.07 JOB08559  ADR734I (002)-DTDSC(01), 2002.346 10:55:07 CONCURRENT COPY  431
  430             SETS WERE SELECTED: 0 FAILED SERIALIZATION AND 0 FAILED FOR OTHER
  430             REASONS.
  431             INITIALIZATION SUCCESSFUL FOR 1 OF 1 SELECTED DATA SETS. SERIALIZATION
  431             FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT. THE INTERMEDIATE RETURN
  431             CODE IS 0004.
 10.55.07 JOB08559  ADR734I (004)-DTDSC(01), 2002.346 10:55:07 CONCURRENT COPY  432
432             INITIALIZATION SUCCESSFUL FOR 1 OF 1 SELECTED DATA SETS. SERIALIZATION
432             FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT. THE INTERMEDIATE RETURN
432             CODE IS 0004.
10.55.29 JOB08559  ADR013I (003)-CLTSK(01), 2002.346 10:55:29 TASK COMPLETED WITH RETURN 529
   529             CODE 0004
10.55.30 JOB08559  ADR013I (004)-CLTSK(01), 2002.346 10:55:30 TASK COMPLETED WITH RETURN 531
   531             CODE 0004
 10.55.30 JOB08559  ADR013I (002)-CLTSK(01), 2002.346 10:55:30 TASK COMPLETED WITH RETURN  532
   532             CODE 0004
10.55.30 JOB08559  -      --TIMINGS (MINS.)--            ----PAGING COUNTS---
10.55.30 JOB08559  -JOBNAME  STEPNAME PROCSTEP    RC    EXCP    CONN    TCB    SRB  CLOCK   SERV
PG  PAGE  SWAP  VIO SWAPS
10.55.30 JOB08559  -USRDFR#1 STEP1 04 2813 14871 .02 .02     .7  66771   0     0     0     0     0
10.55.30 JOB08559  IEF404I USRDFR#1 - ENDED - TIME=10.55.30
10.55.30 JOB08559  -USRDFR#1 ENDED.  NAME-DFDSS DUMP           TOTAL TCB CPU TIME=   .02  TOTAL
ELAPSED TIME=    .7
10.55.30 JOB08559  $HASP395 USRDFR#1 ENDED

PAGE 0001     5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES     2002.346 10:54
PARALLEL
ADR101I (R/I)-RI01 (01), TASKID 001 HAS BEEN ASSIGNED TO COMMAND 'PARALLEL'
 DUMP DATASET(INCLUDE(USR10.R50.CR2.SEG2.PAGE1))  -
OUTDDNAME(TAPE1) CONCURRENT NOTIFYCC
ADR101I (R/I)-RI01 (01), TASKID 002 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
 DUMP DATASET(INCLUDE(USR10.R50.CR2.SEG2.PAGE2))  -
          OUTDDNAME(TAPE2) CONCURRENT NOTIFYCC
ADR101I (R/I)-RI01 (01), TASKID 003 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
 DUMP DATASET(INCLUDE(USR10.R50.CR2.SEG2.PAGE3))  -
          OUTDDNAME(TAPE3) CONCURRENT NOTIFYCC
```

```
ADR101I (R/I)-RI01 (01), TASKID 004 HAS BEEN ASSIGNED TO COMMAND 'DUMP '
ADR109I (R/I)-RI01 (01), 2002.346 10:54:48 INITIAL SCAN OF USER CONTROL STATEMENTS COMPLETED.
ADR014I (SCH)-DSSU (02), 2002.346 10:54:48 ALL PREVIOUSLY SCHEDULED TASKS COMPLETED. PARALLEL
MODE NOW IN EFFECT
ADR016I (002)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR016I (003)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR016I (004)-PRIME(01), RACF LOGGING OPTION IN EFFECT FOR THIS TASK
ADR006I (003)-STEND(01), 2002.346 10:54:48 EXECUTION BEGINS
ADR721I (003)-DTDSC(01), DATA SET USR10.R50.CR2.SEG2.PAGE2 IN CATALOG ICFCAT.R6E0P1 ON VOLUME
ASL019 IS BEING PROCESSED AS A
BACKUP-WHILE-OPEN DATA SET
ADR767I (003)-T0MI (02), 2002.346 10:54:54 CONCURRENT COPY INITIALIZATION SUCCESSFUL FOR
DATA SET USR10.R50.CR2.SEG2.PAGE2 IN
                         CATALOG ICFCAT.R6E0P1. SERIALIZATION IS RELEASED.
ADR730W (003)-DTDSC(01), CLUSTER USR10.R50.CR2.SEG2.PAGE2 IS OPEN
ADR801I (003)-DTDSC(01), DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA SETS WERE SELECTED: 0 FAILED
SERIALIZATION AND 0 FAILED FOROTHER REASONS.
ADR734I (003)-DTDSC(01), 2002.346 10:54:54 CONCURRENT COPY INITIALIZATION SUCCESSFUL FOR 1 OF 1
SELECTED DATA SETS. SERIALIZATION  FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT. THE
INTERMEDIATE RETURN CODE IS 0004.
ADR454I (003)-DTDSC(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                          CLUSTER NAME   USR10.R50.CR2.SEG2.PAGE2
                          CATALOG NAME   ICFCAT.R6E0P1
                          COMPONENT NAME USR10.R50.CR2.SEG2.PAGE2.DATA
ADR006I (003)-STEND(02), 2002.346 10:55:29 EXECUTION ENDS
ADR013I (003)-CLTSK(01), 2002.346 10:55:29 TASK COMPLETED WITH RETURN CODE 0004 ADR006I
(004)-STEND(01), 2002.346 10:54:48 EXECUTION BEGINS
ADR721I (004)-DTDSC(01), DATA SET USR10.R50.CR2.SEG2.PAGE3 IN CATALOG ICFCAT.R6E0P1 ON
VOLUME ASL048 IS BEING PROCESSED AS A
                         BACKUP-WHILE-OPEN DATA SET
ADR767I (004)-T0MI (02), 2002.346 10:55:07 CONCURRENT COPY INITIALIZATION SUCCESSFUL FOR DATA SET
USR10.R50.CR2.SEG2.PAGE3 IN
                         CATALOG ICFCAT.R6E0P1. SERIALIZATION IS RELEASED.
ADR730W (004)-DTDSC(01), CLUSTER USR10.R50.CR2.SEG2.PAGE3 IS OPEN
ADR801I (004)-DTDSC(01), DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA SETS WERE SELECTED: 0 FAILED
SERIALIZATION AND 0 FAILED FOR OTHER REASONS.
PAGE 0002     5695-DF175  DFSMSDSS V2R10.0 DATA SET SERVICES     2002.346 10:54
ADR734I (004)-DTDSC(01), 2002.346 10:55:07 CONCURRENT COPY INITIALIZATION SUCCESSFUL FOR 1 OF 1
SELECTED DATA SETS. SERIALIZATION FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT. THE INTERMEDIATE
RETURN CODE IS 0004.
ADR454I (004)-DTDSC(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
CLUSTER NAME   USR10.R50.CR2.SEG2.PAGE3
CATALOG NAME   ICFCAT.R6E0P1
COMPONENT NAME USR10.R50.CR2.SEG2.PAGE3.DATA
ADR006I (004)-STEND(02), 2002.346 10:55:30 EXECUTION ENDS
ADR006I (002)-STEND(01), 2002.346 10:54:48 EXECUTION BEGINS
ADR013I (004)-CLTSK(01), 2002.346 10:55:30 TASK COMPLETED WITH RETURN CODE 0004
ADR721I (002)-DTDSC(01), DATA SET USR10.R50.CR2.SEG2.PAGE1 IN CATALOG ICFCAT.R6E0P1 ON VOLUME
ASL049 IS BEING PROCESSED AS A
BACKUP-WHILE-OPEN DATA SET
ADR767I (002)-T0MI (02), 2002.346 10:55:07 CONCURRENT COPY INITIALIZATION SUCCESSFUL FOR DATA SET
USR10.R50.CR2.SEG2.PAGE1 IN
CATALOG ICFCAT.R6E0P1. SERIALIZATION IS RELEASED.
ADR730W (002)-DTDSC(01), CLUSTER USR10.R50.CR2.SEG2.PAGE1 IS OPEN
ADR801I (002)-DTDSC(01), DATA SET FILTERING IS COMPLETE. 1 OF 1 DATA SETS WERE SELECTED: 0 FAILED
SERIALIZATION AND 0 FAILED FOR   OTHER REASONS.
ADR734I (002)-DTDSC(01), 2002.346 10:55:07 CONCURRENT COPY INITIALIZATION SUCCESSFUL FOR 1 OF 1
SELECTED DATA SETS. SERIALIZATION FOR THIS DATA IS RELEASED IF DFSMSDSS HELD IT. THE INTERMEDIATE
RETURN CODE IS 0004.
ADR454I (002)-DTDSC(01), THE FOLLOWING DATA SETS WERE SUCCESSFULLY PROCESSED
                          CLUSTER NAME   USR10.R50.CR2.SEG2.PAGE1
                          CATALOG NAME   ICFCAT.R6E0P1
COMPONENT NAME USR10.R50.CR2.SEG2.PAGE1.DATA
ADR006I (002)-STEND(02), 2002.346 10:55:30 EXECUTION ENDS
ADR013I (002)-CLTSK(01), 2002.346 10:55:30 TASK COMPLETED WITH RETURN CODE 0004
ADR012I (SCH)-DSSU (01), 2002.346 10:55:30 DFSMSDSS PROCESSING COMPLETE. HIGHEST RETURN CODE IS
0004 FROM:
        TASK    002
        TASK    003
        TASK    004
```

\

Step 4: Unfreeze the TIBCO Object Service Broker system

After the ADR734I messages are issued for all the data sets being dumped, you can issue the TIBCO Object Service Broker UNFREEZE command to resume TIBCO Object Service Broker processing.

```
10.55.37 JOB08525  S6BKP005L-USRA MODIFY USR50CR2,UNFREEZE
10.55.37 JOB08525  S6BKP006I-USRA MODIFY COMMAND ACCEPTED
10.55.37 JOB08525  S6BKX070I-USRA SYSTEM IS UNFROZEN: USR50CR2
10.55.37 JOB08525  S6BKX052L-USRA END CHECKPOINT -      186
```

Step 5: Sample output from Status and BWOStatus commands

Status command

When the system is active:

```
11.40.37 JOB08525  S6BKP005L-USRA MODIFY USR50CR2,STATUS
11.40.37 JOB08525  S6BKP204I-USRA SYSTEM IS ACTIVE:     USR50CR2
11.40.37 JOB08525  S6BKP208I-USRA BACKUP WHILE OPEN SUPPORT ENABLED
```

When the system is frozen:

```
11.41.54 JOB08525  S6BKP005L-USRA MODIFY USR50CR2,STATUS
11.41.54 JOB08525  S6BKP204I-USRA SYSTEM IS FROZEN:     USR50CR2
11.41.54 JOB08525  S6BKP208I-USRA BACKUP WHILE OPEN SUPPORT ENABLED
```

BWOStatus command

When the system is active (the status of 000 indicates that the data set is ineligible for BWO):

```
11.45.49 JOB08525  S6BKP005L-USRA MODIFY USR50CR2,BWOSTATUS
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG0.PAGE1
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG0.PAGE2
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG0.PAGE3
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG1.PAGE1
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG1.PAGE2
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG1.PAGE3
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG2.PAGE1
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG2.PAGE2
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG2.PAGE3
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG3.PAGE1
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG3.PAGE2
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.SEG3.PAGE3
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.S99.PAGE1
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.S99.PAGE2
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.S99.PAGE3
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.JRNL1
11.45.49 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 000  RESET   (00-00) USR10.R50.CR2.JRNL2
```

When the system is frozen (the status of 100 indicates that the data set is eligible for BWO):

```
11.44.17 JOB08525  S6BKP005L-USRA MODIFY USR50CR2,BWOSTATUS
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG0.PAGE1
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG0.PAGE2
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG0.PAGE3
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG1.PAGE1
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG1.PAGE2
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG1.PAGE3
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG2.PAGE1
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG2.PAGE2
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG2.PAGE3
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG3.PAGE1
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG3.PAGE2
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.SEG3.PAGE3
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.S99.PAGE1
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.S99.PAGE2
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.S99.PAGE3
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.JRNL1
11.44.17 JOB08525  S6BKF051I-USRA CURRENT BWO STATUS 100  BWO     (00-00) USR10.R50.CR2.JRNL2
```

Required Facilities and Process for Softek TDMF

What is Softek TDMF?

Softek TDMF Storage Management Software is a vendor-independent, non-disruptive software solution that can help perform three critical tasks needed in today's IT centers:

1. Swap migrations – move data from one storage volume to another, while data centers maintain full read/write access and response time.
2. Point-In-Time migrations – replicate data from one storage volume to another for a multitude of purposes.
3. Provide easy access to the point-in-time copies of your data.

What is Softek TDMF Offline Volume Access Facility (OVA)?

The Softek TDMF Offline Volume Access (OVA) allows for easy processing of a point-in-time (PIT) volume copy. With OVA, data can be read from the offline target volumes while normal processing continues to update the source volumes. Using OVA, for example, customers can copy or backup to tape full DASD volumes and/or the databases on them, creating a PIT-consistent archived copy, without halting updates to those databases for the length of time required for backup jobs or image copies.

Prerequisites

To use Softek TDMF, Mainframe Edition to maintain TIBCO Object Service Broker segments, your system must have the following installed:

1. Softek TDMF, Mainframe Edition, available from Softek (http://www.softek.com), with the Offline Volume Access (OVA) component.

TDMFIPGM OVA Interface Program

The Softek TDMF Interface program (TDMFIPGM) is used to register each OVA to the Softek TDMF sessions creating (or that did create) the PIT copies. When registration is complete, a user-specified program is invoked via TDMFIPGM. While this program is active, I/O requests from OVA that are directed towards an OVA source volume are redirected to the PIT target volume.

Control statements for TDMFIPGM specify volume serial numbers that are to be included in, or excluded from, I/O redirection. It is also permissible to specify data set names rather than volume serial numbers for the inclusion or exclusion functions. However, the entire volume where such a data set is resident is always included or excluded from the I/O redirection.

TDMFIPGM also intercepts reserve and DEQ requests from the invoked program and modifies these to act on the PIT target device.

Process of Using TDMF OVA to Maintain TIBCO Object Service Broker Segments

OVA can be used to maintain TIBCO Object Service Broker segments while maintaining access to the TIBCO Object Service Broker data. The general process is as follows:

1. Ensure that the segments to be maintained are ONLINE to the TIBCO Object Service Broker system. This prevents access by a TIBCO Object Service Broker offline utility.
2. Start a Softek TDMF point-in-time copy of all the DASD volumes that contain page data sets for the segments to be maintained.
3. When Softek TDMF is ready to synchronize, issue the FREEZE TIBCO Object Service Broker command.
4. When the TIBCO Object Service Broker system is frozen, complete the Softek TDMF point-in-time copy of the DASD volumes.
5. When the Softek TDMF point-in-time copy is complete, issue the UNFREEZE TIBCO Object Service Broker command to get TIBCO Object Service Broker to continue processing.
6. Run the desired TIBCO Object Service Broker utilities under the control of OVA.

Examples of Maintenance Steps Using Softek TDMF

The examples shown in this section were created using a version of TDMF that is no longer supported. However, all the facilities used are still supported by the latest version of TDMF. In addition, a new format of control card has been introduced, although those shown here still function correctly. TDMF is a product of Softek Storage Solutions Corporation which is now owned by IBM. For updated information regarding TDMF, contact Softek (www.softek.com).

Step 1: Obtain List of Volumes to be Copied Using Softek TDMF

In the example, the page data sets for segments 0 and 1 reside on DASD volumes TMP247 and TMP24A. The following gives the output from an ISPF 3.4 data set list:

```
TMP24A TSO4.HUR50.CR2.SEG0.PAGE1    *VSAM* TSO4.HUR50.CR2.SEG0.PAGE1.DATA
TMP247 TSO4.HUR50.CR2.SEG0.PAGE2    *VSAM* TSO4.HUR50.CR2.SEG0.PAGE2.DATA
TMP247 TSO4.HUR50.CR2.SEG0.PAGE3    *VSAM* TSO4.HUR50.CR2.SEG0.PAGE3.DATA
TMP24A TSO4.HUR50.CR2.SEG1.PAGE1    *VSAM* TSO4.HUR50.CR2.SEG1.PAGE1.DATA
TMP24A TSO4.HUR50.CR2.SEG1.PAGE2    *VSAM* TSO4.HUR50.CR2.SEG1.PAGE2.DATA
TMP24A TSO4.HUR50.CR2.SEG1.PAGE3    *VSAM* TSO4.HUR50.CR2.SEG1.PAGE3.DATA
```

Step 2: Set up a Migration Job to Perform a Point-in-time Copy of the Volumes

The following example sets up a Softek TDMF migration job. In the example that follows, volumes TMP247 and TMP24A are copied to volumes TDM318 and TDM319. The copies are assigned a group name of USERG, and only allocated tracks are copied:

```
//USR10TDM JOB (0),'USER',MSGCLASS=H,CLASS=A,NOTIFY=&SYSUID
//STEP1   EXEC PGM=TDMFMAIN,PARM=MASTER
//STEPLIB  DD  DSN=SYS3.TDM210.TDMLLIB,DISP=SHR
//SECCOM   DD  DSN=SYS3.TDM210.TDMLLIB,DISP=SHR
//SYSCOM   DD  DSN=SYS3.SYSCOM4,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSSNAP  DD  SYSOUT=*
//SYSIN    DD  *
TDMF OPTIONS CDS  MASTER Y Y Y Y
TDMF SYSTEMS CDS  MASTER SYS3.SYSCOM4
----+----1----+----2----+----3----+----4----+----5----+----6----+-------+----
TDMF MIGRATE TMP247 TDM318        P P N       USERG          Y Y Y Y
TDMF MIGRATE TMP24A TDM319        P P N       USERG          Y Y Y Y
*    <----->                      Next record must be a MIGRATE record
*            <----->              Volume serial number of source volume
*                  <---->         Volume serial number of target volume
*Type of migration ------------>  P
*POINT-IN-TIME
*When synchronization occurs  ----> P
*Prompt required via
*TDMFMON to continue
*Optional volume Purge ------------>  N
*No purge
*Optional group name  ----------------------> <------>
*These volumes are considered to be a part
*of the group named USERG
*Optional terminate group on error ---------------------->   Y
*Migration copied for whole group if an error occurs
*Optional bypass PPRC checking     ----------------------->   Y
*Allow SWAP from active PPRC device to non-mirrored device
*Optional fast copy requested      --------------------------> Y
*Only tracks described in VTOC as allocated are copied
*Optional Associated Address Space Facility requested ----------> Y
*I/O redirection from source to target after the migration
TDMF END
//
```

Step 3: Confirm that all Segments to be Processed are Online to TIBCO Object Service Broker

You can do this using either the TIBCO Object Service Broker Administration Menu SEGMENT/DASD Statistics Option or by issuing the TIBCO Object Service Broker Data Object Broker command DBSEGMENTSTATUS. In the examples that follows, segments 0, 1, 2, 3, and 99 are online, and segment 4 is offline.

Sample output from DBSEGMENTSTATUS command:

```
S6BKP005L-USRA MODIFY USR50CR2,DBSEGMENTSTATUS
S6BKP050I-USRA A ID  NAME     D/S  MODE     JRN      PAGES     USED      READ     WRITES
S6BKP051I-USRA O 000 CR2.SEG0   3  R/W-S    YES      54000     24432        0          0
S6BKP051I-USRA O 001 CR2.SEG1   3  R/W       NO      54000      3114        0          0
S6BKP051I-USRA O 002 CR2.SEG2   3  R/W      YES      54000        55        0          0
S6BKP051I-USRA O 003 CR2.SEG3   3  R/W      YES      54000        55        0          0
S6BKP051I-USRA F 004 CR2.SEG4   3  R/W       NO          0         0        0          0
S6BKP051I-USRA O 099 CR2.S99    3  R/W-S     NO      27000       426        0          0
```

Sample output from Administrator Menu SEGMENT/DASD statistics option:

```
S6B1RNADMB1 USR50CR2              SEGMENT STATISTICS              2005 MAY18 10:16:56
SEG NAME     MOD PAGES   FREE    % JRN SYS DEL D/S  WARN     READ    WRITE     HOLD
  0 CR2.SEG0 R/W   54K    29K   54 YES YES LOG   3 70/10        0        0        0
  1 CR2.SEG1 R/W   54K    50K   94 NO  NO  LOG   3 70/05        0        0        0
  2 CR2.SEG2 R/W   54K    53K   99 YES NO  LOG   3 80/05        0        0        0
  3 CR2.SEG3 R/W   54K    53K   99 YES NO  LOG   3 80/05        0        0        0
  4 CR2.SEG4 OFF                  NO  NO  LOG   3 80/05
 99 CR2.S99  R/W   27K    26K   98 NO  YES LOG   3 80/05        0        0        0
```

Step 4: Run the Softek TDMF Point-in-time Migration Job

Submit the Softek TDMF migration job detailed above.

System Log Output at the Start of Softek TDMF Migration Job

```
$HASP373 USR10TDM STARTED - INIT    2 - CLASS A - SYS CDS
IEF403I USR10TDM - STARTED - TIME=13.57.56
 *01 TDM2412A Confirmation requested, reply CANCEL, or group USERG    .
  R 01,USERG
 TDM2420I Migration initialization process starting for group USERG
```

Output from Softek TDMF Monitor while Migration Job is Executing

```
Transparent Data Migration Facility.   Master V2.1.0 Session Active.
ComDataSet : SYS3.SYSCOM4
Source    Migration   Percent Complete --------->
VolSer      Phase     0...10...20...30...40...50...60...70...80...90..100
TMP247    Copy        -->
TMP24A    Copy        >
```

Output from Job Log and Softek TDMF Monitor When Migration is Ready to Synchronize

```
*02 TDM2414A Ready to synchronize, reply CANCEL, or group USERG   .
Transparent Data Migration Facility.   Master V2.1.0 Session Active.
 ComDataSet : SYS3.SYSCOM4
 Source    Migration   Percent Complete --------->
 VolSer      Phase     0...10...20...30...40...50...60...70...80...90..100
 TMP247  Refresh 6     ------------------------------------------------->
 TMP24A  Refresh 2     ------------------------------------------------->
```

Now the migration is ready to synchronize, FREEZE the TIBCO Object Service Broker system.

```
F USR50CR2,FREEZE
S6BKP005L-USRA MODIFY USR50CR2,FREEZE
S6BKP006I-USRA MODIFY COMMAND ACCEPTED
S6BKX051L-USRA START CHECKPOINT - 000002
S6BKX069I-USRA SYSTEM FROZEN USR50CR2
```

This processing initializes a checkpoint before the system is FROZEN. The checkpoint completes when the system is UNFROZEN: see later.

Output From Job Log and Softek TDMF Monitor when TIBCO Object Service Broker is FROZEN

Now reply to outstanding WTOR to synchronize Softek TDMF copy.

```
R 02,USERG
IEE600I REPLY TO 02 IS;USERG

TDM2803I Volume activity is being resumed for group USERG    .
Transparent Data Migration Facility.    Master V2.1.0 Session Active.
ComDataSet : SYS3.SYSCOM4
Source     Migration    Percent Complete -------‡
VolSer       Phase      0…10…20…30…40…50…60…70…80…90..100
TMP247   Waiting OVA
TMP24A   Waiting OVA
```

Softek TDMF migration is complete and waits for OVA sessions. These must be started within 15 minutes.

Output From Job Log when Softek TDMF Migration is Complete

Enter the UNFREEZE TIBCO Object Service Broker command to allow TIBCO Object Service Broker to continue processing.

```
F USR50CR2,UNFREEZE
S6BKP005L-USRA MODIFY USR50CR2,UNFREEZE
S6BKP006I-USRA MODIFY COMMAND ACCEPTED
S6BKX070I-USRA SYSTEM UNFROZEN USR50CR2
S6BKX052L-USRA END CHECKPOINT - 000002
```

Now that TIBCO Object Service Broker is UNFROZEN, the checkpoint started by FREEZE completes.

Step 5: Run TIBCO Object Service Broker Utilities Under the Control of OVA

Job to Back up Segment 0 Using S6BTLBPS under OVA

```
//USRCB2S0 JOB (1),'BACKUP S0',MSGCLASS=H,TIME=(3,00),REGION=4M
//JOBLIB   DD DSN=TSO4.HUR50.LOAD,DISP=SHR
//         DD DSN=SYS3.TDM210.TDMLLIB,DISP=SHR
//*--------------------------------------------
//*
//* +-------------------------------------------------------------+
//* |                                                             |
//* |  Sample JCL for Associated Address Space Facility Job.      |
//* |                                                             |
//* +-------------------------------------------------------------+
//IPGM    EXEC PGM=TDMFIPGM
//IPGMOUT   DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//IPGMIN    DD *
      PROGRAM S6BTLBPS
      PARM SEG=0
      INCLUDE (TMP247,TMP24A)
      EXCLUDE DSN=TSO4.HUR50.SEG0
//* +-------------------------------------------------------------+
//* |  Above is the JCL for an OVA job.                           |
//* |  Below this line is the JCL required by the PROGRAM above   |
//* +-------------------------------------------------------------+
//DBDLIB   DD DSN=TSO4.HUR50.CR2.DBDLIB,DISP=SHR
//JOURNAL  DD DISP=SHR,DSN=TSO4.HUR50.SEG0
//
```

Output from Softek TDMF Monitor and Backup Job

```
JOB00158 ---- TUESDAY,   15 MAY 2001 ----
JOB00158  IRR010I  USERID TSO4     IS ASSIGNED TO THIS JOB.
JOB00158  ICH70001I TSO4     LAST ACCESS AT 14:35:07 ON TUESDAY, MAY 15, 2001
JOB00158  $HASP373 USRCB2S0 STARTED - INIT    3 - CLASS A - SYS CDS
JOB00158  IEF403I USRCB2S0 - STARTED - TIME=14.41.12
JOB00158  +S6BKX002I DSN=TSO4.HUR50.CR2.SEG0.PAGE1
JOB00158  IEC161I 056-084,USRCB2S0,IPGM,SYS00001,,,TSO4.HUR50.CR2.SEG0.PAGE1,
JOB00158  IEC161I TSO4.HUR50.CR2.SEG0.PAGE1.DATA,USERCAT.TSO
JOB00158  IEC161I 062-086,USRCB2S0,IPGM,SYS00001,,,TSO4.HUR50.CR2.SEG0.PAGE1,
JOB00158  IEC161I TSO4.HUR50.CR2.SEG0.PAGE1.DATA,USERCAT.TSO
JOB00158  +S6BKX002I DSN=TSO4.HUR50.CR2.SEG0.PAGE2
JOB00158  IEC161I 056-084,USRCB2S0,IPGM,SYS00002,,,TSO4.HUR50.CR2.SEG0.PAGE2,
JOB00158  IEC161I TSO4.HUR50.CR2.SEG0.PAGE2.DATA,USERCAT.TSO
JOB00158  IEC161I 062-086,USRCB2S0,IPGM,SYS00002,,,TSO4.HUR50.CR2.SEG0.PAGE2,
JOB00158  IEC161I TSO4.HUR50.CR2.SEG0.PAGE2.DATA,USERCAT.TSO
JOB00158  +S6BKX002I DSN=TSO4.HUR50.CR2.SEG0.PAGE3
JOB00158  IEC161I 056-084,USRCB2S0,IPGM,SYS00003,,,TSO4.HUR50.CR2.SEG0.PAGE3,
JOB00158  IEC161I TSO4.HUR50.CR2.SEG0.PAGE3.DATA,USERCAT.TSO
JOB00158  IEC161I 062-086,USRCB2S0,IPGM,SYS00003,,,TSO4.HUR50.CR2.SEG0.PAGE3,
JOB00158  IEC161I TSO4.HUR50.CR2.SEG0.PAGE3.DATA,USERCAT.TSO
JOB00158  +S6BTL022I 000020669 PAGES ARCHIVED TO JOURNAL
JOB00158  -                                                  --TIMINGS (MINS.)--
JOB00158  -JOBNAME  STEPNAME PROCSTEP    RC   EXCP   CONN    TCB    SRB  CLOCK
JOB00158  -USRCB2S0 IPGM                 00   3490      0    .33    .06    2.4
JOB00158  IEF404I USRCB2S0 - ENDED - TIME=14.43.36
JOB00158  -USRCB2S0 ENDED.  NAME-BACKUP S0           TOTAL TCB CPU TIME=   .33
JOB00158  $HASP395 USRCB2S0 ENDED
```

```
TRANSPARENT DATA MIGRATION FACILITY V2.1.0
                             COPYRIGHT AMDAHL CORPORATION 1996-2001
             MESSAGE         INPUT STATEMENT / MESSAGE TEXT
    TIME     NUMBER
             TDM8000I              PROGRAM S6BTLBPS
             TDM8000I              PARM SEG=0
             TDM8000I              INCLUDE (TMP247,TMP24A)
             TDM8000I              EXCLUDE DSN=TSO4.HUR50.SEG0
                            TRANSPARENT DATA MIGRATION FACILITY V2.1.0
                             COPYRIGHT AMDAHL CORPORATION 1996-2001
             MESSAGE         INPUT STATEMENT / MESSAGE TEXT
    TIME     NUMBER
14:41:34.590 TDM8100I         Registration successful for TDMF job USR10TDM.
14:41:34.590 TDM8004I         Registered on 001 TDMF session(s).
14:41:34.591 TDM8005I         Source   Actual   TDMF job   Point-in-Time Complete
                               vol      vol
                              TMP247   TDM318   USR10TDM   05/15/2001  14:28:01.
                              TMP24A   TDM319   USR10TDM   05/15/2001  14:28:01.
14:43:07.781 TDM8027I         Program S6BTLBPS completed. Return code: 0000.
Transparent Data Migration Facility.   Master V2.1.0 Session Active.
ComDataSet : SYS3.SYSCOM4
Source    Migration   Percent Complete --------->
VolSer      Phase     0...10...20...30...40...50...60...70...80...90..100
TMP247    OVA Active
TMP24A    OVA Active
```

The VSAM verification messages IEC161I can be ignored. These are produced as the VSAM data sets are open during the Softek TDMF point-in-time copy.

Job to Pointer Check Segment 0 Using S6BBRPTR Under OVA

```
//USRCB2P0 JOB (1),'PTRCHECK S0',MSGCLASS=Y,TIME=(3,00),REGION=4M
//JOBLIB   DD DSN=TSO4.HUR50.LOAD,DISP=SHR
//         DD DSN=SYS3.TDM210.TDMLLIB,DISP=SHR
/*-----------------------------------------
//*
//* +-------------------------------------------------------------+
//* |                                                             |
//* |  Sample JCL for Associated Address Space Facility Job.      |
//* |                                                             |
//* +-------------------------------------------------------------+
//IPGM    EXEC PGM=TDMFIPGM
//IPGMOUT   DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
//IPGMIN    DD *
PROGRAM S6BBRPTR
PARM SEGMENT=0,HDR
INCLUDE (TMP247,TMP24A)
//* +-------------------------------------------------------------+
//* |  Above is the JCL for an OVA job.                           |
//* |  Below this line is the JCL required by the PROGRAM above   |
//* +-------------------------------------------------------------+
//DBDLIB   DD DSN=TSO4.HUR50.CR2.DBDLIB,DISP=SHR
//INDEX    DD DSN=&&INDEX,UNIT=sysda,
//         DISP=(MOD,DELETE,DELETE),
//         SPACE=(CYL,(5,5)),
//         DCB=(RECFM=VB,LRECL=4800,BLKSIZE=19069)
//PAGES    DD UNIT=SYSDA,SPACE=(CYL,(5,5)),
//         DCB=(RECFM=FB,LRECL=4096,BLKSIZE=4096)
//AUDIT  DD SYSOUT=H
        //ORPHAN DD SYSOUT=H,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//REFLOG DD SYSOUT=H,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//* "PGHDR" DD STATEMENT ONLY REQUIRED IF "HDR" PARM USED
//PGHDR  DD SYSOUT=H,DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//* "ERRLOG" CAN BE USED AS INPUT TO S6BBRPGC (RECLAIM ORPHANS)
//ERRLOG DD UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),
//          DCB=(RECFM=FB,LRECL=132,BLKSIZE=0)
```

Job to Back up Segment 0 Page Data Set 2 Using S6BTLUPS Under OVA

```
//USRCB2S0 JOB (1),'BACKUP S0',MSGCLASS=H,TIME=(3,00),REGION=4M        JOB0016
//JOBLIB   DD DSN=TSO4.HUR50.LOAD,DISP=SHR    < USE FOR CR2            0002000
//         DD DSN=SYS3.TDM210.TDMLLIB,DISP=SHR                         0002000
//*--------------------------------------------                        0003000
//*
//* +-------------------------------------------------------------+
//* |                                                             |
//* |  Sample JCL for Associated Address Space Facility Job.      |
//* |                                                             |
//* +-------------------------------------------------------------+
//IPGM    EXEC PGM=TDMFIPGM
//IPGMOUT   DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
                              //IPGMIN    DD *
   PROGRAM S6BTLUPS
INCLUDE (TMP247,TMP24A)
                      EXCLUDE DSN=TSO4.HUR50.SEG0.P2
//* +-------------------------------------------------------------+
//* |  ABOVE IS THE JCL FOR AN OVA JOB.                           |
//* |  BELOW THIS LINE IS THE JCL REQUIRED BY THE PROGRAM ABOVE   |
//* +-------------------------------------------------------------+
//INDD     DD DISP=SHR,DSN=TSO4.HUR50.CR2.SEG0.PAGE2
//OUTDD    DD DISP=SHR,DSN=TSO4.HUR50.SEG0.P2
```

Output from Softek TDMF Migration Job

```
13:57:55 JOB00105 ---- TUESDAY,   15 MAY 2001 ----
13.57.55 JOB00105  IRR010I  USERID TSO4     IS ASSIGNED TO THIS JOB.
13.57.56 JOB00105  ICH70001I TSO4     LAST ACCESS AT 13:40:41 ON TUESDAY, MAY 15, 2001
13.57.56 JOB00105  $HASP373 USR10TDM STARTED - INIT    2 - CLASS A - SYS CDS
13.57.56 JOB00105  IEF403I USR10TDM - STARTED - TIME=13.57.56
13.58.06 JOB00105 *01 TDM2412A Confirmation requested, reply CANCEL, or group USERG   .
13.59.13 JOB00105  R 01,USERG
13.59.17 JOB00105  TDM2420I Migration initialization process starting for group USERG   .
14.22.29 JOB00105 *02 TDM2414A Ready to synchronize, reply CANCEL, or group USERG   .
14.27.59 JOB00105  R 02,USERG
14.28.01 JOB00105  TDM2803I Volume activity is being resumed for group USERG   .
15.19.53 JOB00105  TDM2426I Point-In-Time Migration completed successfully group USERG   .
15.19.59 JOB00105  IEF404I USR10TDM - ENDED - TIME=15.19.59
15.19.59 JOB00105  -USR10TDM ENDED.  NAME-USER             TOTAL TCB CPU TIME=   .36  TOTAL ELAPSED TIME=  82.0
      TRANSPARENT DATA MIGRATION FACILITY V2.3.0
         Fujitsu Softek
      Technical Support SITE 12456
COPYRIGHT FUJITSU SOFTWARE TECHNOLOGY CORPORATION 1996-2003                        PAGE 001

   SYS VOL MESSAGE                                TDMF INPUT
   DATE        TIME     ---- NUMBER ----                                  MESSAGE TEXT
05/15/2001 13:58:03.252  01    TDM1000I  TDMF OPTIONS CDS  MASTER Y Y Y Y
05/15/2001 13:58:03.252  01    TDM1000I  TDMF SYSTEMS CDS  MASTER SYS3.SYSCOM4
05/15/2001 13:58:03.252  01 01 TDM1000I  TDMF MIGRATE TMP247 TDM318        P P N       USERG          Y Y Y Y
05/15/2001 13:58:03.252  01 02 TDM1000I  TDMF MIGRATE TMP24A TDM319        P P N       USERG          Y Y Y Y
05/15/2001 13:58:03.252        TDM1000I  TDMF END
      TRANSPARENT DATA MIGRATION FACILITY V2.3.0
         Fujitsu Softek
      Technical Support SITE 12456
   COPYRIGHT FUJITSU SOFTWARE TECHNOLOGY CORPORATION 1996-2003                     PAGE 002
SYS VOL MESSAGE                              SYSTEM  CDS
   DATE        TIME     ---- NUMBER ----                                  MESSAGE TEXT
05/15/2001 13:58:01.599  01    TDM1727I Installed software for this system is: TDMF PTF #Z93191 Level Software.
05/15/2001 13:58:03.253  01    TDM1449I The Time-of-Day option was specified as LOCAL.
05/15/2001 13:58:03.253  01    TDM1453I The Pacing System Option was specified as OFF.
05/15/2001 13:58:03.253  01    TDM1455I Pacing has been changed to ON by the OPTIONS statement in TDMF job stream.
05/15/2001 13:58:03.253  01    TDM1465I The Operator Messages Option was specified as OFF.
05/15/2001 13:58:03.253  01    TDM1466I The Operator Messages Option has been changed to ON.
05/15/2001 13:58:03.253  01    TDM1457I Confirmation on Initialization system option was specified as OFF.
05/15/2001 13:58:03.253  01    TDM1458I The Confirmation on Initialization system option has been set to ON.
05/15/2001 13:58:03.253  01    TDM1461I Terminate All Volumes in Group on Error Option was specified as OFF.
05/15/2001 13:58:03.253  01    TDM1753I Target Volume Empty option set to "N" or not specified.
05/15/2001 13:58:03.334  01    TDM1379I The application program interface (API) for IBM is not available.
05/15/2001 13:58:03.364  01    TDM1380I The application program interface (API) for STK is not available.
05/15/2001 15:19:56.806  01    TDM1642E HISTORY DataSet definition invalid.
      TRANSPARENT DATA MIGRATION FACILITY V2.3.0
         Fujitsu Softek
      Technical Support SITE 12456
   COPYRIGHT FUJITSU SOFTWARE TECHNOLOGY CORPORATION 1996-2003                     PAGE 003
SYS VOL MESSAGE                              SYSTEM  CDS
   DATE        TIME     ---- NUMBER ----                                  MESSAGE TEXT

05/15/2001 13:58:03.284  01 01 TDM1481I A volume TermGrp option was changed to ON by a Migrate card option.
05/15/2001 13:58:03.284  01 01 TDM1578W The "Allow SWAP to non-PPRC" option is not relevant for this Point-in-Time
05/15/2001 13:58:03.284  01 01 TDM1381I The volume Fast Copy option was requested by a Migrate input statement.
05/15/2001 13:58:03.284  01 01 TDM1468I The Associated Address Space Facility (OVA) was specified for a volume.
05/15/2001 13:58:03.861  01 01 TDM1177I The source volume TMP247 is mounted on device 0247 on this system.
05/15/2001 13:58:03.861  01 01 TDM1180I At TDMF initialization, caching is not active on the source volume subsystem.
05/15/2001 13:58:03.861  01 01 TDM1183I At TDMF initialization, the source volume has caching (CFW) deactivated.
05/15/2001 13:58:03.861  01 01 TDM1185I At TDMF initialization, the source volume has DASD fast write deactivated.
05/15/2001 13:58:03.861  01 01 TDM1186I The target volume TDM318 is mounted on device 0318 on this system.
05/15/2001 13:58:03.861  01 01 TDM1189I At TDMF initialization, the target volume has caching (CFW) activated.
05/15/2001 13:58:03.861  01 01 TDM1191I At TDMF initialization, the target volume has DASD fast write allowed.
05/15/2001 13:58:06.418  01 01 TDM2412A Confirmation requested, reply CANCEL, or group USERG   .
05/15/2001 13:59:17.745  01 01 TDM2418I Group migration confirmation received from console TSO4    .
05/15/2001 13:59:17.753  01 01 TDM2406I All storage frames to migrate this volume have been successfully page fixed.
05/15/2001 13:59:17.753  01 01 TDM2403I This volume successfully selected for initialization and is a group volume.
05/15/2001 13:59:17.759  01 01 TDM2281I The master system is starting the initialization process for a volume.
05/15/2001 13:59:17.759  01 01 TDM2420I Migration initialization process starting for group USERG   .
05/15/2001 13:59:17.970  01 01 TDM2283I The master system is starting the migration process for a volume.
05/15/2001 13:59:18.476  01 01 TDM3528I Starting the copy of the source volume.
05/15/2001 14:13:40.183  01 01 TDM3529I Completed the copy of the source volume.
05/15/2001 14:13:55.333  01 01 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:13:55.333  01 01 TDM3531I Completed a refresh of the target volume.
05/15/2001 14:14:10.550  01 01 TDM2285I The master system is starting the quiesce process for a volume.
05/15/2001 14:14:10.551  01 01 TDM2291I The master system is starting the resume process for a volume.
05/15/2001 14:14:10.553  01 01 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:14:10.553  01 01 TDM3531I Completed a refresh of the target volume.
05/15/2001 14:21:14.195  01 01 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:21:14.358  01 01 TDM3531I Completed a refresh of the target volume.
05/15/2001 14:21:59.410  01 01 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:21:59.499  01 01 TDM3531I Completed a refresh of the target volume.
```

```
05/15/2001 14:22:29.507  01 01 TDM2414A Ready to synchronize, reply CANCEL, or group USERG   .
05/15/2001 14:22:59.623  01 01 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:22:59.714  01 01 TDM3531I Completed a refresh of the target volume.
05/15/2001 14:23:59.814  01 01 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:23:59.912  01 01 TDM3531I Completed a refresh of the target volume.
05/15/2001 14:25:00.010  01 01 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:25:00.106  01 01 TDM3531I Completed a refresh of the target volume.
05/15/2001 14:27:30.490  01 01 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:27:30.851  01 01 TDM3531I Completed a refresh of the target volume.
05/15/2001 14:28:00.581  01 01 TDM2399I Group Synchronization was responded to via console TSO4    .
05/15/2001 14:28:00.582  01 01 TDM2285I The master system is starting the quiesce process for a volume.
05/15/2001 14:28:00.584  01 01 TDM3532I Starting the synchronization of the target volume.
05/15/2001 14:28:00.584  01 01 TDM3533I Completed the synchronization of the target volume.
05/15/2001 14:28:00.589  01 01 TDM3524I Successful movement of the source volume TMP247 to the target volume TDM318.
05/15/2001 14:28:01.820  01 01 TDM2349I The master system is starting the point-in-time migration process for a vol
05/15/2001 14:28:01.843  01 01 TDM2291I The master system is starting the resume process for a volume.
05/15/2001 14:28:01.986  01 01 TDM2697I Associated Address Space Facility (OVA) is waiting for an OVA job.
05/15/2001 14:41:34.584  01 01 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:43:35.475  01 01 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:44:05.569  01 01 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:44:35.666  01 01 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:45:05.770  01 01 TDM2695I An Associated Address Space Facility (OVA) job has become active.
   TRANSPARENT DATA MIGRATION FACILITY V2.3.0
        Fujitsu Softek
      Technical Support SITE 12456
   COPYRIGHT FUJITSU SOFTWARE TECHNOLOGY CORPORATION 1996-2003                    PAGE 004
SYS VOL MESSAGE                                SYSTEM  CDS
   DATE        TIME     ---- NUMBER ----                              MESSAGE TEXT

05/15/2001 14:46:05.984  01 01 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:48:36.453  01 01 TDM2695I An Associated Address Space Facility (OVA) job has
05/15/2001 14:49:36.679  01 01 TDM2696I Associated Address Space Facility (OVA) is now inactive.become active.
05/15/2001 14:50:06.784  01 01 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:50:36.881  01 01 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:58:38.473  01 01 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:59:08.620  01 01 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 15:00:39.622  01 01 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 15:04:42.508  01 01 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 15:19:52.727  01 01 TDM2698I Associated Address Space Facility (OVA) has completed all OVA functions.
05/15/2001 15:19:52.727  01 01 TDM2293I The master system is starting the termination process for a volume.
05/15/2001 15:19:52.930  01 01 TDM2303I The master system has completed the migration process for a volume.
05/15/2001 15:19:52.981  01 01 TDM2410I All storage frames to migrate this volume have been successfully page freed.
05/15/2001 13:58:03.288  01 02 TDM1481I A volume TermGrp option was changed to ON by a Migrate card option.
05/15/2001 13:58:03.288  01 02 TDM1578W The "Allow SWAP to non-PPRC" option isn't relevant for this Point-In-Time cop
05/15/2001 13:58:03.288  01 02 TDM1381I The volume Fast Copy option was requested by a Migrate input statement.
05/15/2001 13:58:03.288  01 02 TDM1468I The Associated Address Space Facility (OVA) was specified for a volume.
05/15/2001 13:58:03.861  01 02 TDM1177I The source volume TMP24A is mounted on device 024A on this system.
05/15/2001 13:58:03.862  01 02 TDM1180I At TDMF initialization, caching is not active on the source volume subsystem.
05/15/2001 13:58:03.862  01 02 TDM1183I At TDMF initialization, the source volume has caching (CFW) deactivated.
05/15/2001 13:58:03.862  01 02 TDM1185I At TDMF initialization, the source volume has DASD fast write deactivated.
05/15/2001 13:58:03.862  01 02 TDM1186I The target volume TDM319 is mounted on device 031D on this system.
05/15/2001 13:58:03.862  01 02 TDM1189I At TDMF initialization, the target volume has caching (CFW) activated.
05/15/2001 13:58:03.862  01 02 TDM1191I At TDMF initialization, the target volume has DASD fast write allowed.
05/15/2001 13:59:17.759  01 02 TDM2406I All storage frames to migrate this volume have been successfully page fixed.
05/15/2001 13:59:17.759  01 02 TDM2403I This volume successfully selected for initialization and is a group volume.
05/15/2001 13:59:17.973  01 02 TDM2281I The master system is starting the initialization process for a volume.
05/15/2001 13:59:18.174  01 02 TDM2283I The master system is starting the migration process for a volume.
05/15/2001 13:59:18.841  01 02 TDM3528I Starting the copy of the source volume.
05/15/2001 14:20:56.935  01 02 TDM3529I Completed the copy of the source volume.
05/15/2001 14:21:14.203  01 02 TDM3530I Starting a refresh of the target volume.
05/15/2001 14:21:14.653  01 02 TDM3531I Completed a refresh of the target volume.
05/15/2001 14:21:29.296  01 02 TDM2285I The master system is starting the quiesce process for a volume.
05/15/2001 14:28:00.607  01 02 TDM3532I Starting the synchronization of the target volume.
05/15/2001 14:28:00.607  01 02 TDM3533I Completed the synchronization of the target volume.
05/15/2001 14:28:00.613  01 02 TDM3524I Successful movement of the source volume TMP24A to the target volume TDM319.
05/15/2001 14:28:01.960  01 02 TDM2349I The master system is starting the point-in-time migration process for a vol.
05/15/2001 14:28:01.984  01 02 TDM2291I The master system is starting the resume process for a volume.
05/15/2001 14:28:01.986  01 02 TDM2697I Associated Address Space Facility (OVA) is waiting for an OVA job.
05/15/2001 14:41:34.584  01 02 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:43:35.475  01 02 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:44:05.569  01 02 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:44:35.666  01 02 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:45:05.770  01 02 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:46:05.984  01 02 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:48:36.453  01 02 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:49:36.679  01 02 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:50:06.784  01 02 TDM2695I An Associated Address Space Facility (OVA) job has become active.
      TRANSPARENT DATA MIGRATION FACILITY V2.3.0
         Fujitsu Softek
         Technical Support SITE 12456
   COPYRIGHT FUJITSU SOFTWARE TECHNOLOGY CORPORATION 1996-2003                    PAGE 005
SYS VOL MESSAGE                                SYSTEM  CDS
   DATE        TIME     ---- NUMBER ----                              MESSAGE TEXT
05/15/2001 14:50:36.881  01 02 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 14:58:38.473  01 02 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 14:59:08.620  01 02 TDM2696I Associated Address Space Facility (OVA) is now inactive.
```

```
05/15/2001 15:00:39.622  01 02 TDM2695I An Associated Address Space Facility (OVA) job has become active.
05/15/2001 15:04:42.508  01 02 TDM2696I Associated Address Space Facility (OVA) is now inactive.
05/15/2001 15:19:52.727  01 02 TDM2698I Associated Address Space Facility (OVA) has completed all OVA functions.
05/15/2001 15:19:52.999  01 02 TDM2293I The master system is starting the termination process for a volume.
05/15/2001 15:19:52.999  01 02 TDM2426I Point-In-Time Migration completed successfully group USERG    .
05/15/2001 15:19:53.216  01 02 TDM2303I The master system has completed the migration process for a volume.
05/15/2001 15:19:53.262  01 02 TDM2410I All storage frames to migrate this volume have been successfully page freed.
```

Chapter 8 Recovery Procedures

This chapter describes different procedures for recovering data.

Topics

Introduction to TIBCO Object Service Broker Recovery

Purpose of TIBCO Object Service Broker Recovery

The recovery procedures assure the restoration of all data resident within one specific Pagestore. They do not deal with data external to the Pagestore, such as data residing in external database management systems and remote TIBCO Object Service Broker nodes that participated in database units of work coordinated by this specific TIBCO Object Service Broker system. You must administer their recovery through their own respective database management systems.

A recovery of the system is essentially a replacement of all or part of the current system with a backup copy. You normally perform a recovery procedure only when the current system is corrupted (data was inadvertently altered or destroyed) or when there has been a media failure.

When restoring data that participated in units of work traversing TIBCO Object Service Broker and external databases, it is your responsibility to ensure the integrity of the entire unit of work. That is, if you recover a TIBCO Object Service Broker table, page data set, or segment, you must consider the consequences on the entire unit of work and perform related recovery, if necessary, on the other participating external databases.

Validating the Backup

We recommend that you run the S6BBRPTR (Batch Pointer Check) utility against every new backup to ensure the integrity of the backup. If you did not run the S6BBRPTR utility against the backup you plan to use for recovery, you should do so before proceeding.

See Also *TIBCO Object Service Broker for z/OS Utilities* for more information on S6BBRPTR.

Deciding How Much To Restore

The first step in any recovery process is to determine the nature and scope of the damage from which you need to recover. There are several possibilities:

- The damage is limited to a specific table or a number of known tables.

  In this case, recover these tables from your most recent backup using the S6BBRULA (Recover TDS Table From Archive) utility. For information about the S6BBRULA utility, refer to *TIBCO Object Service Broker for z/OS Utilities*.

- The damage is isolated to a physical hardware device.

  Restore all page data sets on that device to the level they were before the hardware failure. For information about performing a partial system recovery, refer to Restoring A Segment on page 106.

- The damage cannot be isolated (for example, it was caused by a program error).

  You can either restore your entire system to its state just before the occurrence of the program error or restore your system to a specific time by the use of the S6BTLSRP (Select Recovery Pages) utility. Keep in mind that you must use sort exit S6BSPU35 in SPINMRG and BKUPCON instead of S6BSPX35.

  For information about a full system recovery, refer to Full Recovery Procedure on page 98.

- Damage is isolated to system control data sets—journals, redolog, or cache.

  Since there is no damage to the Pagestore in this case, no Pagestore recovery is required. However, you must repair any damaged system data sets before you resume normal application processing.

When restoring to a point in time with database products, you must ensure that all data within the database remains synchronized. If a table, page data set, or segment is restored to a point in time, you must consider the consequences of this on the rest of your system.

Recovery Preparation

Before doing system recovery work, decide on the appropriate recovery level according to what you need and what is possible. Depending on this decision, if necessary, perform some pre-recovery procedures. Your data restoration options include the following:

Option 1: Recover only the data included in your most recent backup copy

If the backup was taken three days ago, at the end of the restore process, the entire TIBCO Object Service Broker system is exactly as it was three days ago. None of the changes made to the system since the backup was taken are reflected. With this option, your latest backup copy is usable as it is. No pre-recovery steps are necessary.

Option 2: Recover data including your most recent backup copy and the latest unmerged

SPINOUT data sets

Run BKUPCON, which picks up all SPINOUT generations and merges them into the backup data sets. You can now use the updated backup copy.

Option 3: Recover data including your most recent backup copy, the latest unmerged SPINOUT data sets, and any unspun journal data

Complete the following steps:

1. Ensure that the latest active journal does not contain incomplete units of work.

   The easiest way to achieve this is to bring up the Data Object Broker to resolve any units of work that are pending in its recovery files.

2. Close the active journal data set.

   Use the **SPINSUBMIT**=I command. This forces an immediate spin to clean out the current active journal. For more information on the **SPINSUBMIT** command, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

3. Run **BKUPCON.**

   BKUPCON picks up all SPINOUT generations and merges them into the backup data sets.

You can now use the updated backup copy.

Option 4: Restore to a specific time using the S6BTLSRP (Select Recovery Pages) utility

You can use the S6BTLSRP utility to identify a specific time to which you want to recover. It then selects the pages that are time-stamped the closest to this date and time and creates a file from which you can restore data. Refer to Point in Time Recovery on page 102 and *TIBCO Object Service Broker for z/OS Utilities* for information on using the S6BTLSRP utility.

Recovery Overview

The basic recovery tasks are as follows:

1. Shut down the TIBCO Object Service Broker system.
2. Prepare the file from which you want to recover.

   For example, you can do this using the S6BTLSRP (Select Recovery Pages) utility or by retrieving an archived file.

3. Rebuild each segment by:
   a. Deleting and redefining the data sets in the segment using **`IDCAMS`**
   b. Initializing the segment page data sets
   c. Restoring the segment page data sets
4. Rebuild the recovery data sets.
5. Restart TIBCO Object Service Broker.

Full Recovery Procedure

The tasks to perform a full recovery are as follows:

These tasks are described in detail in the following sections.

Task A Shut Down the TIBCO Object Service Broker System

For detailed instructions about shutting down TIBCO Object Service Broker, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

Task B Create a Data Set Containing the Latest Page Images

These page images are used to restore the damaged segment.

For Recovery up to Your Last Checkpoint:

1. Offload the journal data sets using the appropriate SPINnn job.
2. Merge the journal data sets with the journal accumulations.
3. Use the S6BSPSEP utility to extract the pages for the required segment from the journal accumulations.
4. Merge the extracted pages with the segment backup
5. Run the S6BBRPTR (Batch Pointer Check) utility using the point-in-time restore file as input and check ERRLOG for any messages.
6. If the S6BBRPTR utility runs cleanly, delete, define, and reformat the page data sets in the segment to be restored and use the restore file to restore your segment.

For Recovery up to a Point in Time:

For each Segment you want to restore to a point-in-time:

1. If the restore point-in-time is very recent, SPIN the current Journal to include the updated pages on it.
2. Using the S6BTLSRP (Select Recovery Pages) utility and all necessary Journal accumulation files as input, create a file that contains the latest page images up to the designated point-in-time.
3. Use the S6BSPSEP utility to extract the pages for the required segment from the journal accumulations.
4. Merge the file created in the previous step with the appropriate segment backup file to create your point-in-time restore file.
5. Run the S6BBRPTR (Batch Pointer Check) utility using the point-in-time restore file as input and check ERRLOG for any messages.
6. If the S6BBRPTR utility runs cleanly, delete, define, and reformat the page data sets in the segment to be restored and use the restore file to restore your segment.

Task C Rebuild Each Segment

Delete and Redefine the Data Sets

1. Use **IDCAMS** DEL/DEF to delete and redefine all the data sets in the specified segment.
2. Refer to the S6A3ALOC member in the OSB.INSTALL data set.
3. Review this JCL carefully and modify it as required.

Initialize the Data Sets

1. Edit the S6A5FRMT member in the OSB.INSTALL data set and ensure that high-level qualifiers are set correctly.

   Follow the documentation within the JCL to specify this.
2. Ensure that all page data sets you want to initialize are listed in the DD statements.

To reduce elapsed time, you can initialize page data sets asynchronously by running parallel jobs that each contain a DD statement for a different page data set.

Task D Restore the Data Sets

1. Create JCL to restore a segment from a TIBCO Object Service Broker system backup.

   The member RESTORE in the JCL data set, distributed with TIBCO Object Service Broker, contains sample JCL to restore a segment from a TIBCO Object Service Broker system backup.

2. Use the parm='Seg=nn' parameter to indicate the segment number you want to restore.
3. Use the JOURNAL DD statement to point to the input data set for the restore process.
4. Set the output parameters to the correct TIBCO Object Service Broker high-level qualifiers.

   Follow the documentation within the sample JCL to specify this.

5. Execute the JCL.

After executing the JCL to restore the data sets, the data is copied from the backup tapes to the specified segment.

See Also For more information about parameters, refer to *TIBCO Object Service Broker Parameters*.

Task E Rebuild the Recovery Data Sets

The recovery data sets are the redolog, the contingency log, CACHE1, and CACHE2.

The following procedure destroys any recovery data that is still in recovery data sets. Be sure to assess the consequences of losing that data. If you are not able to do a clean shutdown of the Data Object Broker, you lose any updates since the last complete checkpoint.

Delete and Redefine the Data Sets

1. Use IDCAMS DEL/DEF to delete and redefine the files.
2. Refer to S6A3ALOC member in the OSB.INSTALL data set for sample JCL.
3. Review this JCL carefully and modify it as required.

Initialize the Data Sets

1. Edit the S6A5FRMT member in the OSB.INSTALL data set

2. Check that the correct data set names (the TIBCO Object Service Broker high-level qualifiers) are specified.
3. Submit the S6A5FRMT job in the OSB.INSTAL data set for execution.

   This job calls the following utilities:

   — S6BTLFCA (Format Cache Data Sets) to initialize the cache data sets

   — S6BTLFRL (Format Redolog) to initialize the redolog

   — S6BTLFJR (Format Journal Data Sets) to initialize the journals

   — S6BTLFCL (Format Contingency Log) to initialize the contingency log.

Task F Restore TIBCO Object Service Broker

For detailed instructions about starting up TIBCO Object Service Broker, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

If the system was backed up using a DASD backup product such as DF/DSS or FDR, it must be restored in the same way as other DASD backed up using that product. This requires that it be restored using the same device type from which it was backed up.

Point in Time Recovery

This type of full recovery is typically required to remove programming type errors that applied updates against a segments.

Performing a point in time recovery can create inconsistencies in a distributed data environment. You must recover other data nodes to the same point to which you recover the affected segments. In addition, recovering one local segment to a specific point can cause inconsistencies in other local segments. For example, since table definitions are stored in the MetaStor (segment 0), table deletes or adds that affect other segments would also require recovery of segment 0.

Procedure

These are the tasks required to perform a successful point in time recovery:

- Develop a Continuous Backup Strategy, page 102
- Cleanse BACKUP File Images, page 103
- Vary Segments Offline or Shut Down Data Object Broker, page 103
- Consolidate Latest Backup Page Images, page 103
- Using the S6BTLSRP (Select Recovery Pages) utility and all necessary Journal accumulation files as input, create a file that contains the latest page images up to the designated point-in-time., page 99
- Verify Validity of the EXTRACT File, page 104
- Perform Recovery Using the S6BTLRPS (Restore TDS Segment) Utility, page 104
- Perform a Backup and Pointer Check, page 104
- Use the BACKUP File as the Basis for Next Recovery Cycle, page 105
- Vary the Segments Online or Restart the Data Object Broker, page 105

These steps are discussed in the following sections.

Task A Develop a Continuous Backup Strategy

A continuous backup strategy is required to retain all updated copies of journaled page images over a limited period of time. Point in time recovery requires that all updated page images be retained for the entire window of recovery. Depending on the frequency of updates and the length of the supported recovery window, the amount of recovery journal data can grow to a significant volume. Therefore,

the length of the recovery window must reflect a balance between keeping the volume of historical recovery data to a manageable size and providing for a large enough recovery period to be useful. For more information on developing a continuous backup, refer to Using the Continuous Backup Approach on page 48.

You must implement SORT exit S6BSPU35 rather than S6BSPX35 in all SPINMRG and BKUPCON jobs in your continuous backup. S6BSPU35 retains all non-identical page images, while S6BSPX35 retains only the latest image of each page based on date, time, and checkpoint number of the page. For more information, refer to Deciding How Much To Restore on page 94.

Task B Cleanse BACKUP File Images

This step is necessary to provide a base to continually collect ongoing duplicate page images for point in time recovery. Cleansing the BACKUP file can be accomplished with a special sort using S6BSPX35 or by using the S6BTLSRP (Select Recovery Pages) utility specifying a recovery date of the current runtime.

Task C Vary Segments Offline or Shut Down Data Object Broker

When a recovery is required, the affected segments must be varied offline or the Data Object Broker terminated with a controlled shutdown. If a DASD failure hits various files, you must recover the Data Object Broker using the procedures for various data set failures. Refer to Recovering From Non-Page Data Set Failures on page 112 for more information. After a successful Data Object Broker recovery, vary the segments to be recovered offline and spin the active journals to flush out any pending updates to the segments.

When a segment is recovered to a certain point, any previously journaled updated pages that were created after the point to which the recovery is performed must be removed from the continuous backup process. Failure to do so can result in corrupted Pagestores if invalid pages are used for subsequent recovery.

All online segments at the time of termination or error condition are candidates for recovery and should be checked afterwards with the S6BBRPTR (Batch Pointer Check) utility for consistency.

Task D Consolidate Latest Backup Page Images

After varying the segments to be recovered offline or successfully shutting down the Data Object Broker, ensure that the journal data sets are spun and consolidated into the backup cycle. This is accomplished by submitting jobs SPIN01, SPIN02, SPINMRG, and BKUPCON.

You must modify the INCLUDE SORT statements in BKUPCON to remove journal records for segments you do not wish to recover. The example below selects pages for Segment 1 only and the resultant file will be a back up of only that segment.

```
INCLUDE COND=(6,1,BI,EQ,X'01')
```

You must also modify the S6BBRPTR (Batch Pointer Check) step for the required segment.

Task E Input the BACKUP File from the Previous Step into the S6BTLSRP (Select Recovery Pages) utility

Specify the parameter 'DATE=YYYYMMDD,TIME=HHMMSS' to the S6BTLSRP utility, identifying the time up to which you want to restore your segments. The time selected should match the end of a checkpoint. Use the message S6BKX052L (end checkpoint) produced by the Data Object Broker for determining appropriate timestamps for recovery.

Task F Verify Validity of the EXTRACT File

Run the S6BBRPTR (Batch Pointer Check) utility to verify the validity of the EXTRACT file produced by the S6BTLSRP utility for each affected segment.

Task G Perform Recovery Using the S6BTLRPS (Restore TDS Segment) Utility

If the EXTRACT file is clean, use it in the S6BTLRPS utility to perform the point in time recovery. Ensure that all pages updated by the Data Object Broker subsequent to the point of recovery are rewritten with previously timestamped updated page images. The recovered segments must not contain any pages updated beyond the recovered point in time. This can best be accomplished by performing a **DELETE** and **FORMAT** of the affected segments prior to the restore with the S6BTLRPS utility.

Task H Perform a Backup and Pointer Check

After the segments are restored, perform a backup and pointer check using the S6BBRPTR utility to ensure that the recovery is successful. The results should match the output from Task F, Verify Validity of the EXTRACT File, on page 104.

Task I Use the BACKUP File as the Basis for Next Recovery Cycle

The BACKUP file output from step 8 can be used as the basis for the next point in time recovery cycle. You must ensure that no previously journaled page images with timestamps after the time of recovery remain in your continuous backup cycle. If any such page image remains in your backup cycle, page corruptions occur if the BACKUP file is used for subsequent recovery operations.

Task J Vary the Segments Online or Restart the Data Object Broker

If the output from step 8 is clean, vary the segments recovered back online to the Data Object Broker. If the Data Object Broker was shut down, restart the Data Object Broker.

Restoring A Segment

If your analysis indicates that the damage you need to recover from is isolated to a specific Pagestore segment, you can follow the steps in this section to recover just that segment. If you are not sure whether other segments are affected, the safest approach is to restore all segments.

To restore a segment, follow the procedure outlined below.

Task A Ensure That There Are No Pending Updates

You do this by varying the affected segment offline or shutting down the Data Object Broker. If segment 0 (the MetaStor) is to be recovered, you must shut down the TIBCO Object Service Broker system.

Be careful not to restore a segment to a point prior to the last physical definition change for any table held in that segment. Access errors can result when users attempt to use the table, since there can be a mismatch between the physical data and the recovered table definition.

Procedure

1. Execute the command to vary a segment offline.

   The command is as follows:

   `Modify` *dob_jobname*,Dboffline=*segmentname* or *segmentnumber*

   Examples:

   ```
   Modify dob_jobname,Dboffline=3
   Modify dob_jobname,Dboffline=sales
   ```

   You can also vary a segment offline using the Administration menu. For more information about the various options on the Administration menu, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

2. Force a journal spin

   When you vary a segment offline, a checkpoint is automatically taken. We recommend that you also force a journal spin to ensure that all database images are written out to the journal accumulation. You can force an immediate journal spin by issuing the following command from the z/OS operator console:

   `Modify` *dob_jobname*,Spinsubmit=Immed

For more information about TIBCO Object Service Broker operator commands, refer to *TIBCO Object Service Broker for z/OS Installing and Operating*.

Task B Extract Parts of the Backup Relating to the Segment

Use a recent backup of that segment, or use a sort filter to read in your most recent full system backup and extract those parts relating to the specific segment.

For example, you can **SORTIN** an entire system backup, and **SPINOUT** only those pages belonging to segment 1. To selectively sort out a specific segment or page data set within a segment, you add an INCLUDE statement to your sort control cards.

This example sorts images for segment 1:

```
INCLUDE COND=(6,1,BI,EQ,X'01')
```

This example sorts images for page data set 3 in segment 1:

```
INCLUDE COND=(6,1,BI,EQ,X'01',AND,7,1,BI,EQ,X'02')
```

Data sets are numbered from 0; therefore, the data set must equal hex '02' to be filtered out.

Task C Rebuild the Selected Page Data Sets

This is described in Full Recovery Procedure on page 98.

Task D Bring the Affected Segment or Full System Back Online

The command to vary a segment online is:

Modify *dob_jobname,*Dboffline=*segmentname* or *segmentnumber*

You can also bring the segment back online from the Administration menu.

Restoring Individual Page Data Sets

The following sections outline the steps required to recover individual page data sets in a number of different recovery scenarios. The steps you follow depend upon your overall approach to data recovery.

Recovering Page Data Sets

To restore the data:

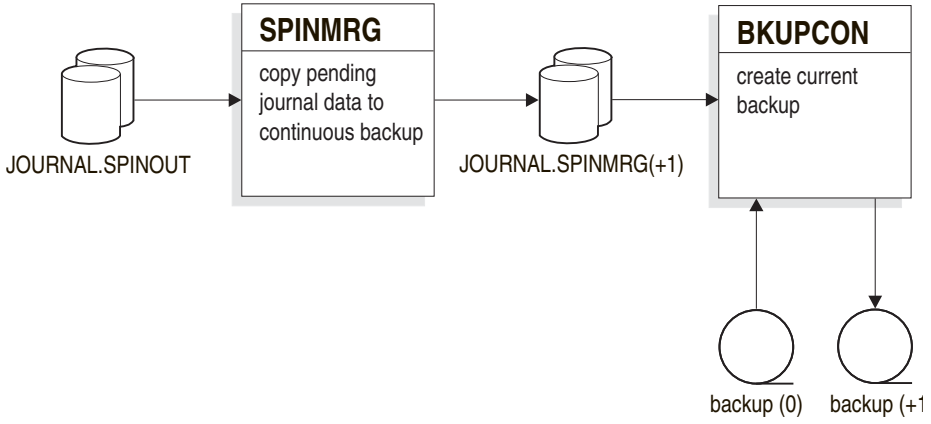
1. Delete and redefine the failing data set.
2. Initialize the data set.
3. Restore the data set using the continuous backup procedure outlined below.
4. Restart the Data Object Broker with the restored page data set and resume normal processing.

   If updates are pending (via the redolog and cache data sets), the Data Object Broker automatically applies the updates as part of initialization.

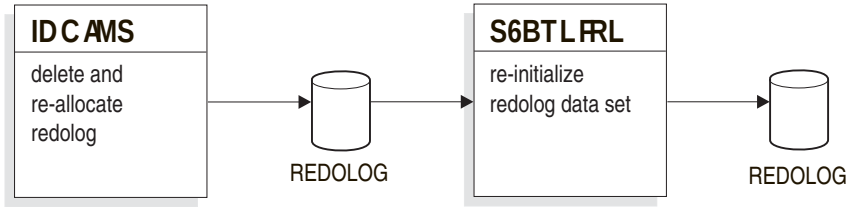
Restoring a Page Data Set Using a Continuous Backup

To restore a page data set that is backed up using the continuous backup method, you must restore the data set from the most recent backup. You must then apply all outstanding journaled page images from the time of the backup to the point of failure. The job flow is as follows:

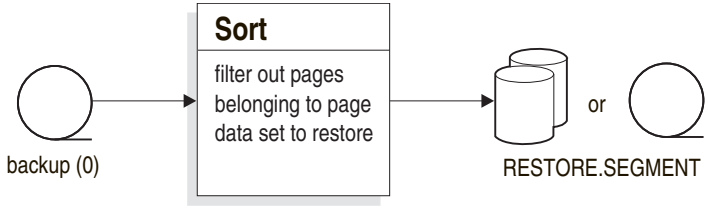
1. Flush all pending journal update information from the Data Object Broker into the continuous backup process as shown below:



2. Delete, redefine, and re-initialize the page data set as shown below:

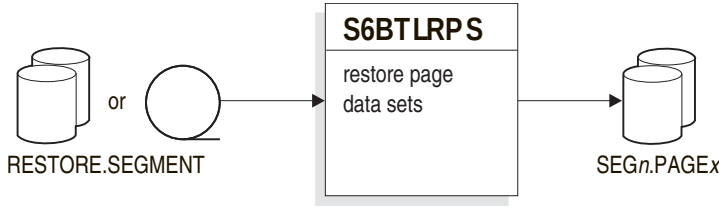


3. Filter out the page images for the particular page data set being restored as shown below:



You must always run the S6BBRPTR (Batch Pointer Check) utility, against newly created data sets to ensure their integrity.

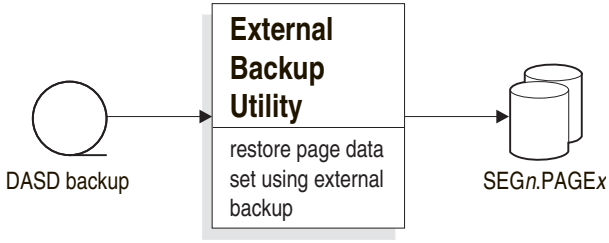
4. Restore the page data set using the filtered backup images as shown below:



Restoring a Page Data Set with an External DASD Backup

To restore page data sets by using an external DASD backup utility:

1. Restore the failing page data set using the external backup utility as shown below:



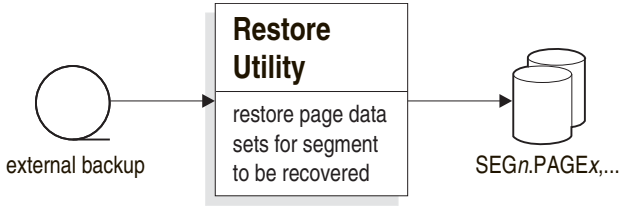
2. Run the S6BBRPTR (Batch Pointer Check) utility, against newly created data sets to ensure their integrity.

Recovery to a Previous Full External Backup

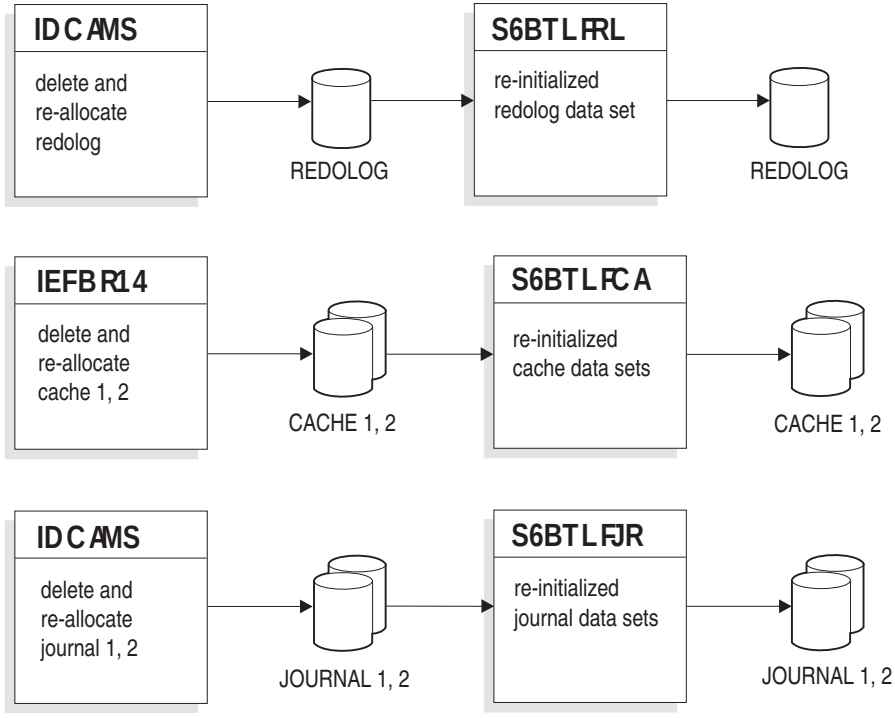
External DASD backup utilities can be used to perform backups of the Pagestore provided records are not relocated within VSAM control intervals. All segments should be offline when the backups are taken and no updates should be pending.

With the Data Object Broker down, a typical recovery procedure would be as follows:

1. Restore all page data sets for the segment to be recovered as shown below:



2. Clear residual update information in all Data Object Broker operational data sets (the redolog, caches, and journals) as shown below:



3. If there are transactions in doubt on the contingency log, evaluate whether you need to abort these transactions.

   This decision depends on whether the external database server or the peer TIBCO Object Service Broker is also restored. You can abort in-doubt transactions from the S6BTLADM (Administration Menu) utility.

4. At this point, restart the Data Object Broker.

   Refresh your continuous backup, if you are using it, from a complete backup taken after the restore completed. This stops any invalid page images created after the point of full backup but prior to the restore process being completed, from being circulated within the continuous backup process.

Recovering From Non-Page Data Set Failures

While journal and backup information can be used to recover page data sets, you must also be prepared to recover from other failures you encounter. Other data sets that can fail include the journals, redolog, CACHE1 and CACHE2 data sets, and the contingency log. Each situation is described in the following sections.

Recovering From Redolog Failure

If the redolog data set becomes damaged, it is possible that you cannot automatically recover the system.

Recovering With a Duplex Data Set

If the Data Object Broker fails due to either the redolog or its duplex becoming damaged, restart the Data Object Broker using a copy of the good data set as the redolog.

Procedure

1. Run IDCAMS to delete and re-allocate the affected redolog.
2. Run the S6BTLFRL (Format Redolog) utility to re-initialize the redolog data set and copy the contents of the good data set into the new redolog.

   Refer to *TIBCO Object Service Broker for z/OS Utilities*.
3. Restart the Data Object Broker, using the copy as the new redolog data set.

IDCAMS cannot be used to back up or copy TIBCO Object Service Broker redolog data sets because it changes the placement of records within the VSAM control interval.

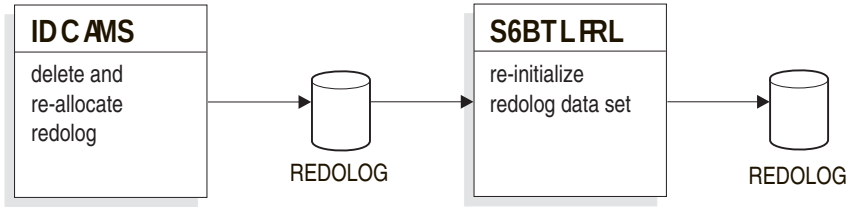
Recovering With No Duplex Data Set

If no duplex exists, you must delete, redefine, and re-initialize it. Loss of the redolog with no duplex causes the loss of commits received by the Data Object Broker since the last successful checkpoint.

Procedure

1. Run IDCAMS to delete and re-allocate the affected redolog.
2. Run the S6BTLFRL (Format Redolog) utility to re-initialize the redolog data set.

The following illustrates these steps:



3. Capture any data from the active journal by manually submitting a spin for the active journal. Refer to Spinning the Journals on page 29.

   You do this so as to have this data still available, if needed. This is necessary because, at restart, TIBCO Object Service Broker sees that you re-initialized the redolog and overwrites the existing journal data.

4. Restart the Data Object Broker.

5. Reply "G" to message S6BKR098, which should appear, along with message S6BKR011, because you have both a reformatted redolog and a cache data set available at this time.

If you have segments online when the Data Object Broker fails, and error messages S6BKX067 and S6BKX022 appear, modify the DBDGEN to include the segment shown in S6BKX067 as online at initialization.

Recovering From Contingency Log Failure

To recover from contingency log failure:

1. Use the S6BTLFCL (Format Contingency Log) utility to delete, redefine, and re-initialize the contingency log data set (REDOLOG.PENDING).

   Loss of the contingency log could result in data inconsistency in a distributed data environment. For more information about the role of the contingency log in distributed data environments, refer to Chapter 3, Understanding Fail Safe Processing, on page 15.

2. Restart your Data Object Broker.

Recovering From Cache Failure

The CACHE1 and CACHE2 data sets are used to record checkpoint information quickly prior to actual segment updates. Loss of either of these data sets is detected during checkpoint processing.

If a cache data set fails, you must delete and redefine it. Information is not lost because all information is recoverable from the redolog. Pending update information is reconstructed from the redolog data set when the Data Object Broker is restarted.

Procedure

To create new cache data sets:

1. Run the S6BTLDBR (Database Recovery Report) utility to print the redolog information.

   Check the report to see the status of checkpoints. For more information on S6BTLDBR, refer to *TIBCO Object Service Broker for z/OS Utilities*.

We recommend that, before proceeding with the rest of this procedure, you contact TIBCO Support for assistance.

2. If all checkpoints are complete, that is, all checkpoint starts have a corresponding checkpoint end, run IEFBR14 to delete and reallocate both cache data sets, run the S6BTLFCA (Format Cache Data Sets) utility to re-initialize the cache data sets, and restart the Data Object Broker.
3. If a checkpoint is outstanding and its data is in the good cache, that is, the checkpoint number in the cache header for the good cache matches the number of an incomplete checkpoint in the redolog information, run IEFBR14 to delete and reallocate the bad data set, run the S6BTLFCA (Format Cache Data Sets) utility to re-initialize this data set, and restart the Data Object Broker, replying G (go) to the S6BKR098A message.
4. If a checkpoint is outstanding and the good cache contains the data for the previous checkpoint, contact TIBCO Support.

Recovering From Journal Failures

The journal data sets provide an audit trail of changed physical pages.

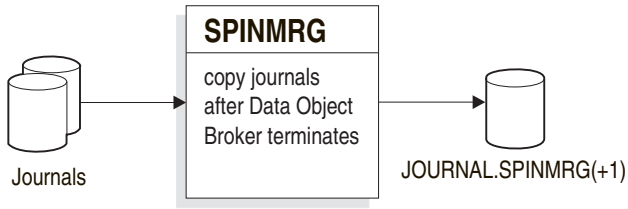
Recovering Journals

If one of your journal data sets fails, you are not able to recover to the point of failure for subsequent page data set failures. For this reason, you should immediately reset the continuous backup process with a new master backup.

After the Data Object Broker is started, you can use subsequent journal images, along with your backup, to recover any page data set failures up to the time of failure. Although the page images contained within the failing journal are lost, your repository and other control data sets are still intact. The steps for recovery are:

1. Back up your segments immediately to prime your continuous or discrete backup process.
2. Delete, reallocate, and re-initialize the failed journal data set.
3. Spin any residual data from the last current journal into your SPINOUT generation data sets.
4. Restart the Data Object Broker.

The following illustrates these steps:



Chapter 9 Errors in Journal Spinning

This chapter describes errors messages and the required actions when spinning the journal.

Topics

Spin Job Completion

Sample Console Messages

A successful spin produces operator console messages resembling the following examples:

```
22.38.19 STC01821   S6BKX008A-TEST JOURNAL 001 SPIN IN PROCESS
22.38.19 STC01821   START HRUNTS1,JRNLDSN=HUR03.HURON.JRNL1
22.38.32 STC01821   S6BKC016L-TEST HRUNTST  LOGGING ON
(OPERATOR/XMS)
22.38.32 STC01821   S6BKX009A-TEST JOURNAL SPIN COMPLETE
22.38.32 STC01821   S6BKC017L-TEST HRUNTS1  LOGGED OFF
```

You can also view these messages using the Browse Alert Messages option of the S6BTLADM (Administration Menu) utility.

Journal Spin Complete Message

If you do not see the `JOURNAL SPIN COMPLETE` message or if you receive an alternate message stating that the current journal is filling and has to be spun, investigate the status of your spin job. If you want make changes to the spin JCL at this time, you can make these changes and resubmit the job manually.

Refreshing the Data Object Broker

The Data Object Broker keeps an image of the spin JCL. Therefore, if you do make changes to the JCL, refresh the Data Object Broker copy. Use the **`SPINLOAD`** operator command to refresh the Data Object Broker copy.

Active Journal Filling – Spin Required

Journal Filling Message

This is an informational message that informs you that the current journal is reaching capacity and that all other journals are unavailable:

```
S6BKX007A - ACTIVE JOURNAL XX% FULL - SPIN REQUIRED FOR JOURNAL
```

This message can be indicating that the journal spin is taking longer than expected. It can also be warning you of a potential problem (for example, the spin job is waiting for a tape mount or has not yet executed). If there is a problem with the spin job, you should take corrective action immediately. This can involve mounting a tape or increasing the priority of the spin job so that it executes immediately.

TIBCO Object Service Broker issues this message to the operator console at the point when the current journal is 50% full and the other journals are still offline. It then repeats this message at 10% capacity intervals until either another journal becomes available or the current journal reaches 100% capacity.

You can also elect to run the spin jobs as a started task. To do this, select **`Spinoption=S`** as an initialization option and move the spin JCL to a system procedure library.

Journal Full Message

If the current journal fills before another journal is available, the following message appears on the operator console:

```
S6BKX006A - ALL JOURNALS FULL - SYSTEM QUIESCED FOR COMMITS
```

This error message informs you that TIBCO Object Service Broker is no longer able to accept commits because all journals are unavailable and therefore nothing can be logged.

When the System Is Quiesced

Error Messages

When the system is quiesced, only query operations from users currently logged in can proceed. Users trying to make updates when the system is quiesced receive error messages such as:

```
TABLE ACCESS ERROR
```

or

```
EXECUTOR ERROR - END OF TRANSACTION FAILED.
```

Enabling Offloaded Journal

When the spinning journal is offloaded, use the operator command **JOURNALON**=*nn* to enable this journal. When the journal is enabled, the system comes out of the quiesce state and the other full journals spin.

The S6BTLADM (Administration Menu) utility can be used while the Data Object Broker is in quiesce mode.

Appendix A Sort Control Manipulation

This appendix describes how to manipulate sort control statements to filter data.

Topics

Manipulating Sort Control Statements

Page Header Format

To manipulate sort control statements for filtering data, you must know the format of the header information. The following illustrates page header format for different SORT offsets:

SORT Offset	Field Length	Field	Description
1	2	ll	Not a page header field; this field is added on for sorting.
3	2	00	Not a page header field; this field is added on for sorting.
5	2	Seg#	Pagestore segment number in binary.
7	1	DS#	The data set number in binary where the value is one less than the number appearing in your JCL. SEG*n*.PAGE1 would be X'00'.
8	3	Page#	The binary page number within the particular data set and segment.
11	4	Prev PTR	Internal chain pointer consisting of a DS# and PAGE#.
15	4	Next PTR	Internal chain pointer consisting of a DS# and PAGE#.
19	1	Type	Page type flag used internally.
20	1	RSVD	Reserved for future use.
21	3	Date	Date of last update in YYMMDD or OYYDDD format. (See note.)
24	3	Time	Time of last update in HHMMSS format.
27	3	Trns ID #	Transaction identification of last update.
30	3	CHKPID#	Checkpoint identification causing last page update.

Note Pages updated or created on or after January 1, 2000 contain a new packed non-signed Julian date format, defined as 'OYYDDD', where ...

O	Is the century indicator relative to the first millennium; that is, `X'A'` is year 2000, `X'B'` is 2100, and so on.
YY	Is the year, 0 to 99.
DDD	Is the day of the year.

On the change of the millennium (1999 December 31 at midnight), the date format changed to the TIBCO Object Service Broker extended Julian page header date. The sort is still able to handle pages updated or created before that date that make use of the YYMMDD format, where ...

YY	Is the year, 0 to 99.
MM	Is the month, 0 to 12.
DD	Is the day of the month.

Existing sort control cards do not require changes.

Index

Numerics

A

B

C

D

E

F

G

I

J

L

M

N

O

P

Q

R

S

T

U

V

X