MICRO FOCUS

Administrator's Guide

Databridge Host

Version 6.6 Service Pack 1

Legal Notices

The only warranties for products and services of Micro Focus and its affiliates and licensors (“Micro Focus”) are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Patents

This Micro Focus software is protected by the following U.S. patents: 6983315,7571180, 7836493, 8332489, and 8214884

Trademarks

Micro Focus, the Micro Focus logo, and Reflection among others, are trademarks or registered trademarks of Micro Focus or its subsidiaries or affiliated companies in the United Kingdom, United States and other countries. RSA Secured and the RSA Secured logo are registered trademark of RSA Security Inc. All other trademarks, trade names, or company names referenced herein are used for identification only and are the property of their respective owners.

Third-Party Notices

Third-party notices, including copyrights and software license texts, can be found in a 'thirdpartynotices' file located in the root directory of the software.

Contents

About This Guide

This guide contains instructions for configuring and using Micro Focus Databridge Host. This preface includes information to help you use this guide.

To install, configure, and run Databridge, you should be a system administrator familiar with the following:

- Standard Unisys® operations for MCP-hosted mainframes such as the CS7xxx series, Libra series, ClearPath® NX/LX or A Series
- DMSII databases and Data And Structure Definition Language (DASDL)
- Transferring files between your host and the system that will use the replicated DMSII database

Conventions

The following conventions and terms may be used in this guide.

This convention	Is used to indicate this
menu > sub menu ...> menu item (dialog box item)	This font style/color shows mouse-clicks in the order required to access a specific function, window, dialog box, etc. The greater than symbol > indicates the next item to click in the series. The parentheses ( ) indicate the setting, option, or parameter being discussed. Note the font style reverts back to normal.
`this type style`	text that you type, filenames and directory names, onscreen messages
italic	variables, emphasis, document titles
square brackets ([ ])	optional items in a command. For example, [true | false]. (Do not type the brackets.) Buttons. For example, [OK], [Start], [Cancel]
pipe (|)	a choice between items in a command or parameter. When enclosed in braces ({ }), the choice is mandatory.
UPPERCASE	DMSII data set and data item names

This term	Is used to indicate this
MCP server host mainframe	Unisys ClearPath NX, LX or A Series mainframe
DBEngine	Databridge Engine
DBEnterprise	Databridge Enterprise Server
DBServer	Databridge Server

Abbreviations

The following abbreviations are used throughout this guide and are provided here for quick reference.

Abbreviation	Name
AA	Absolute Address
ABSN	Audit Block Serial Number
AFN	Audit File Number
API	Application Programming Interface
DASDL	Data and Structure Definition Language
DMSII	Data Management System II
IDX	Index
IPC	Inter-Process Communications
MCP	Master Control Program
RPC	Remote Procedure Call
SEG	Segment
WFL	Work Flow Language

Related Documentation

When using Databridge, you may need to consult the following resources.

Databridge product documentation	On the Databridge installation image, the DOCS folder contains guides for installation, error codes, and administrator's guides for each Databridge product. These documents require Adobe Reader for viewing, which you can download from the Adobe website (http://get.adobe.com/reader/). This documentation, and current technical notes, is also available on the Micro Focus Databridge support site (http://support.attachmate.com/manuals/databridge.html). Documentation for Databridge Enterprise Server and the Databridge Client Console is also available from the **Help** menu. A modern browser is required for viewing this documentation.
Unisys MCP server documentation	If you are not completely familiar with DMSII configuration, refer to your Unisys documentation.

1 Introducing Databridge Host

This chapter introduces the Databridge Host software, its replication methods, and their benefits. It also provides factors to consider when choosing a replication method for your business.

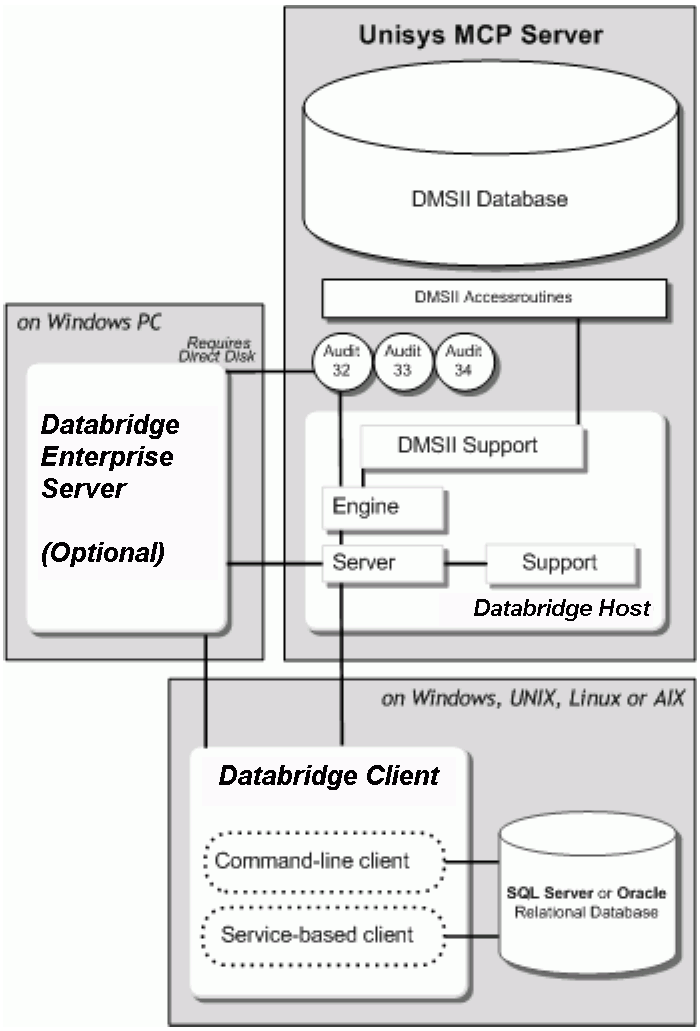
In this Chapter

About Databridge

Databridge is a combination of host software and optional client software that provides automated replication of DMSII databases and flat files. All replications occur while the DMSII database is active. After the initial clone, Databridge updates the secondary database, copying only the DMSII data changes from the audit trail.

By copying the DMSII data to another system, you can offload all queries, reporting, and decision support to a secondary database—such as a relational database on a remote UNIX or Windows server—and reduce the expense associated with mainframe resources. Alternatively, you can copy DMSII data to a secondary database on the host. During replication, the mainframe database can be available at all times instead of being shut down for extractions and downloads and then being reopened for updates.

The following diagram shows the Databridge architecture.



Advantages of Databridge

Databridge provides the following benefits:

- It only requires cloning once.
- It synchronizes data sets/tables in the primary and secondary database.
- You can select only the items you want to replicate from the primary database.
- You can replicate to many platforms.
- Databridge has low resource requirements.
- When you use the Databridge host and client software, no file transfer utility is required to move the replicated data.
- The secondary database provides a secure way to make data available to selected individuals, departments, or sites while protecting the primary database on the host.
- The data is available on the client system even when the host is down or experiencing long wait times, or when the data communication connection is broken.

- The client system provides the processing resources for database queries with no impact on the host. And since the data is typically filtered (that is, the client database contains only the necessary data items) there is less overhead.
- You can avoid the prohibitive time and expense of developing (or purchasing) custom data conversion and porting software, and you can avoid costly mainframe hardware purchases that might be necessary if all users accessed the host directly.

Security Considerations

When you run a Databridge Accessory, normal host security restrictions apply. For example, the Databridge Accessory must run under a usercode that has access to the appropriate DMSII DESCRIPTION files, database CONTROL files, and the audit files. All of the typical file access rules of the host apply to the Databridge Accessories. In other words, security is dependent on the usercode under which Databridge is run.

Usercodes

We recommend that you install and run Databridge under a privileged usercode. To run Databridge under a different usercode from the one under which you installed it requires that you copy Databridge Engine (DBEngine) to that usercode or establish DBEngine as a system library. For details, see "DBEngine Visibility to Accessories and Usercodes" on page 26.

Guard Files

Databridge supports MCP guard file validation on the host. Guard files provide controls that restrict access to specific files. For information about setting up guard files, see the *Databridge Installation Guide*.

Custom Filtering

Databridge provides a method for restricting access to certain data sets (or remaps) and certain records within the data sets. You can put these visibility restrictions in a tailored support library via the GenFormat program. Before Databridge returns a record to the secondary database system, it applies the visibility constraints in the tailored support library. See "Creating a Filter" on page 76.

Logical Databases

You can restrict access to data sets and records by creating a logical database and then running the Accessory against the logical database. DBEngine restricts access to data sets and remaps in the logical database. You can specify the logical database when you start the Accessories via the WFLs.

Databridge Support for DMSII Structure Types

Databridge supports data sets, remaps, and logical databases. Databridge Accessories can use a remap just like a data set.

For information on DMSII structure types supported by Databridge Clients, see the *Databridge Client Administrator's Guide*.

Databridge supports all DMSII data set structure types and data items, except the following:

Data Sets Not Supported

- Embedded data sets if INDEPENDENTTRANS is reset or if embedded within an ORDERED or COMPACT data set. If the EMBEDDED EXTRACTS option is set in the Databridge Span or Snapshot parameter file or the Databridge Client configuration, such embedded data sets can be extracted (cloned, but without any fixups) but not tracked.
- Partitioned structures

Data Items Not Supported

- POPULATION
- COUNT
- FILLER
- AGGREGATE

Even though the Databridge Engine (DBEngine) does not return layout information for the types listed above, by using RAWFORMAT, a host Accessory (Databridge Span, for example) can gain access to the data for the item types listed above. If the Accessory uses a format generated by DBGenFormat, though, these special data items are not available. The Databridge clients do not have access to these special data items.

Choosing a Replication Method

Databridge offers several ways to clone (page 210) a DMSII database. You can use any of these methods while the DMSII database is in operation and being updated.

Use this	For
Databridge Snapshot	Single DMSII data extraction or infrequent clones of a database. Snapshot clones all of the selected data sets each time you run it (not just the changes). Because cleanup and consolidation takes place on the host, Snapshot uses more host resources than Span. See "Span and Snapshot Compared" on page 108 for a comparison of these Accessories. For more information about Snapshot, see How Databridge Snapshot Works (page 147).
Databridge Span	A primary database that has frequent changes. Span gathers only changes to the primary database when it writes updates to its output files. For more information, see How Databridge Span Works (page 105). For a comparison of these Accessories, see "Span and Snapshot Compared" on page 108. **NOTE:** This Accessory does not apply the fixup records (page 212). In addition, the files must be manually transferred if used on a secondary system.

The following methods require Databridge Host and additional Databridge components.

Use this	For

Use this	For
Databridge Server and the Databridge Client	Cloning and transferring DMSII data and subsequent updates to a remote client. For more information, see the *Databridge Client Administrator's Guide* and the Databridge Client Console **Help**.
Databridge Enterprise Server and Databridge Client	Cloning and transferring DMSII data and subsequent updates to a remote client. Use this method to reduce mainframe usage and related expense by moving all cloning, tracking, and filtering to a Windows system. This method is similar to the preceding method except that Enterprise Server performs the functionality of Databridge Server and the Databridge Engine. For more information, see the Help included with the Databridge Enterprise Server.
Databridge FileXtract	Replicating flat files. For more information, see the *Databridge FileXtract Administrator's Guide*.
DMSII Client or Databridge Twin	Replication to a Unisys MCP Server. For more information, see the *Databridge DMSII Client Administrator's Guide* or the *Databridge Twin Administrator's Guide*.

Updating

Updating is an ongoing process for propagating changes from records in the DMSII database to the secondary database (page 213). Databridge uses the Audit Files (page 19) created by DMSII to extract and apply the updates to the secondary database.

NOTE: Databridge cannot read audit files copied to tape with the Unisys QuickCopy program.

NOTE: Databridge can clone an unaudited database but cannot track any changes, i.e. it cannot do any updating.

Updating can be accomplished using exclusively Databridge Host software. Alternatively, you can incorporate Databridge Client software to perform this function.

Host-based updating

In this method of updating, you run Databridge Span to retrieve updates from DBEngine and then write those records to one or more data files on the host. You then transfer those files to the new location and load the data into a secondary database.

An advantage of using host-based updating is that you can replicate DMSII data to any system, even if Micro Focus has not developed a Databridge Client for that particular system.

A disadvantage of this method is that it puts more responsibility on you to deliver and process the updates in the data files.

Client-based updating

In this method of updating, a Databridge Client program (typically running on another system) communicates with DBServer (or DBEnterprise), which retrieves information from DBEngine.

The Databridge Client requests updates from DBServer, which in turn calls the appropriate entry points (page 211) in DBEngine to retrieve the modified records. The Databridge Client program then updates the client database with those changes.

Loose vs. Tight Replication

You can determine how current you want to make your secondary database by choosing between a loose replication and a tight replication. Regardless of which you choose, only Databridge updates should be allowed on the secondary database. This maintains data integrity between the primary database (page 212) and the secondary database (page 213).

Loose replication

In the loose replication model, Databridge accesses only closed audit files for the DMSII changes. This results in a time delay between a change to the primary database and the corresponding change to the secondary database.

You can control the amount of time between secondary database updates in the following ways:

- Using the AuditTimer (page 179) utility to schedule audit file closes. For example, you could close the audit file every hour, every day, or even every week, depending on how crucial current information is.
- Using the Copy Audit (page 186) utility to schedule Databridge Span to start each time an audit file closes.
- Using the Copy Audit utility or the Notify (page 178) utility in conjunction with Databridge Server to notify the Client that additional audit file information is available.
- Running the Databridge Accessory each time you want updates.

Tight replication

In the tight replication model, Databridge accesses the current audit file for the DMSII changes. Updates to the primary database are performed almost immediately on the secondary database.

Databridge can read the current audit file and provide near real-time updates to the secondary database.

Audit Files

An audit file is created by DMSII and contains the raw format of changes made to the DMSII database by update programs. Audit file records contain the deletes, adds, and modifies that were made to the various structures. It can contain, for example, hours', days', or weeks' worth of information.

Databridge uses the audit file for the raw data of each database change to exactly replicate the primary database. Databridge records the audit location (AFN, ABSN, SEG, IDX) between runs, so it can restart without losing any records.

If you set the DBEngine Read Active Audit option, Databridge can access the current audit file. If you do not set Read Active Audit = true in the DBEngine parameter file, Databridge can access audit information from audit files up to, but not including, the current DMSII audit file. The audit file contains the update level at the time the audit file was created. The update level in the audit file and the update level in the DMSII DESCRIPTION file used by Databridge must match before Databridge will update a replicated database.

When an audit file is closed, DMSII creates the next one in the series. Audit files are closed for several reasons, including the following:

- An operator closes the audit file with the mixnumber SM AUDIT CLOSE command.
- The audit file reaches the file size set in its DASDL.
- There is an I/O error on the audit file.
- There is not enough disk space for this audit file.
- The database update level changes due to database definition changes
- The current audit file could not be found.
- A file reorganization was executed to modify the DMSII structure.

Forcing an Audit Switch

If you generate less than one audit file a day and you don’t want to set Read Active Audit = TRUE, you may want to force an audit switch daily and run the Span Accessory or DBServer so that the secondary (replicated) database is at most one day behind. In this case, you can use the following procedure instead of running AuditTimer.

This works only for databases that are currently open for updating (that is, one or more programs have done an OPEN UPDATE *databasename*).

To force an audit switch

1. From the ODT or action line in MARC, transmit the following:

   ```
   DBS
   ```

   A list of active databases and their associated mix numbers appears.
2. Transmit the following for the database audit file you want to close:

   ```
   mixnumber SM AUDIT CLOSE FORCE
   ```

where *mixnumber* is the mix number of an active database.

The current audit file is now closed, and a new one is created.

You can also use Audit Close utility to close the current audit file. See "Audit Close Utility" on page 188. For additional information on forcing an audit switch, see the Unisys DMSII Utilities documentation.

Replication Overview

The following steps outline the procedure for using host-based Accessories to replicate a DMSII database.

1. Run the Lister Accessory (page 212) to determine which DMSII data sets you want to clone or replicate with Databridge.
2. From the Lister report, make a list of the data sets you want to replicate.
3. Edit the DBGenFormat (page 212) file and create the record formats and filters that you want to use. When done, compile the Databridge Support library.
4. Choose the replication method (for example, using the Databridge Span or Snapshot Accessory).
5. Run the Accessory against the DMSII database. (Both Accessories generate a default parameter file the first time you run them.)
6. Edit the parameter file by selecting the data sets you want to replicate, the formatting and filtering routines to use, etc.
7. Replicate the DMSII data by running Databridge Span or Snapshot to write data files.
8. Transfer the data files for the replicated data sets to the secondary database system. This is site dependent. You might use a file transfer program, for example, or you might physically carry tapes to another location.
9. Update the replicated DMSII data.

   If you use Databridge Span to replicate your DMSII database, update the secondary database by running the Span Accessory to write data files that contain changes gathered from the audit trail. Then, use one of your user-written programs to read these data files and update the secondary database.

   If you use Databridge Snapshot to clone your DMSII database, update the secondary database in one of two ways:

   - Run Databridge Snapshot again. The entire collection of data sets is recloned and the old secondary database is reloaded by running one of your user-written programs with the new data files.
   - Insert the Snapshot audit location into a Databridge Span parameter file and then run Databridge Span.

NOTE: Databridge Accessories do not permanently reside in the mix. Instead, initiate the Accessory for the task you want to complete. Databridge then initiates and subsequently terminates any other programs, libraries, or utilities needed to complete the task. As long as you installed Databridge so it has visibility to your DMSUPPORT Library and DMSII DESCRIPTION and CONTROL files, you can run the Databridge Accessories whenever necessary.

Description of Databridge Components

Each Databridge product (for example, Databridge Host, Databridge Client) and the components they include are described in the following table.

Databridge Host (installed on the mainframe)

Component	Description
Engine (DBEngine)	The main component of the Databridge software, the Databridge Engine is a host library program that retrieves structural information, layout information, and data from the DMSII database and audit file and passes the information to Databridge Server.
DMSII Support Library (DBDMSIISupport)	DMSII Support is a Databridge library that retrieves data records from the DMSII database for cloning. The Databridge Engine links to this library to perform database functions such as reading records, switching the audit file, and getting database statistics.
Server (DBServer)	An Accessory that provides communications between DBEngine and the Databridge Client, and also between DBEngine and Databridge Enterprise Server. DBServer responds to Databridge Client requests for DMSII data or DMSII layout information. It retrieves updates by asking DBEngine to read the audit files on the host and send the changes to the Client.
Support Library (DBSupport)	A library that provides translation, formatting, and filtering to the DBServer and other Accessories. After DBServer receives data from the Databridge Engine, it calls the Support Library to determine if the data should be replicated, and if so, passes the data to the Support Library for formatting.
GenFormat (DBGenFormat)	A host utility that creates translation, filter, and format routines. The GenFormat utility interprets the GenFormat parameter file to generate ALGOL source code patches, which are included in the tailored Support Library.
Span (DBSpan)	Produces a replication of one or more data sets into flat sequential disk files that can be extended when more audit becomes available. The Databridge Span Accessory updates the extracted flat files by appending the changes to the end of the flat files (unlike the Databridge Snapshot Accessory, which replaces the changed records).
Snapshot (DBSnapshot)	Produces a clone of one or more data sets into tape or flat sequential disk files that consist of records suitable for bulk loading into a client application (for example, a spreadsheet or a relational database). The Snapshot Accessory clones the selected data sets each time you run it.
Lister (DBLister)	Produces a report of the layout of the structures in your DMSII database, including structure numbers and key sets.
Info Utility (DBInfo)	Produces a report of your DMSII database timestamps, update levels, DMSII release levels, etc.
WFL (Work Flow Language) Jobs	Provide customizable ways to run Databridge Accessories. For example, the Notify WFL makes DBServer notify the Client whenever audit files are available for processing.
AuditTimer Utility	Schedules times for closing an audit file.

Component	Description
Copy Audit Utility	Enables you to specify the number of closed audit files that should be saved on disk, automatically have DBServer notify the Client each time an audit file becomes available, and run the Span Accessory each time an audit file becomes available.
Audit Close Utility	Closes (switches) the current audit file.
Audit Remove Utility	Removes processed audit files.
Sample Source Code and Accessories	Illustrate how to use the Databridge API to write your own Accessories. These sample Accessories are not supported. For more information, see the *Databridge Programmer's Reference*.
APIs (Application Program Interfaces)	Provide access to DBEngine from any Databridge or user-written Accessory (for example, Span) for the purpose of retrieving information for a DMSII database. For more information about the Databridge API, see the *Databridge Programmer's Reference*.

Databridge Enterprise Server

A Windows-based product that provides the same functionality as the Databridge Engine (DBEngine) and Databridge Server (DBServer) on the host. Enterprise Server offloads much of the replication workload from the Unisys mainframe to a Windows computer, reducing mainframe resource utilization and initial load time.

Databridge Clients can connect directly to Enterprise Server, which in turn connects to DBServer on the mainframe. If MCP disks are directly accessible from the Windows server, Enterprise Server extracts the DMSII data directly. Enterprise Server reads the audit trail on the host to retrieve updates that occurred during the extraction and sends the changed information from the audit file to the Client. If MCP disks are not directly accessible, Enterprise Server uses DBServer to retrieve blocks of data from DMSII data sets or the audit files. Enterprise Server provides high-speed file transfer between the host and the Windows environment and audit file mirroring.

Component	Description
Enterprise Server (DBEnterprise)	A Windows program that provides replication services to Clients.
Director (DBDirector)	A Windows Service that listens for Client connection requests and starts Enterprise Server whenever a connect request is received.
EnumerateDisk (EnumDisk)	A command-line program that lists MCP disks that are visible to Enterprise Server.

Databridge Client

The Databridge Client is a product that replicates the DMSII database to a relational database. The Client initiates a connection with DBServer on the MCP server (or DBEnterprise on a Windows server) and then specifies the DMSII data sets to be replicated from a DMSII database.

Component	Description

Client Service	The service (Windows) or daemon (UNIX) that automates most Client operations. It handles operator requests from the Client Console and routes all log and informational messages to the consoles.
DBClient	A Client program that is launched by the service. DBClient handles the processing of DMSII data and updates the same as the command line client (dbutility), except that it runs as a background run and uses the Client Console to display its output and interact with the operator.
DBClntCfgServer	A program that handles all requests from the Client Console specific to a data source. These requests include updating the client configuration file, providing access to the client control tables, and handling the Client Configurator. Like DBClient, this program is run by the service as a background run.
dbutility	A program that runs the Databridge Client from a command line.
Batch Console (bconsole)	A program that allows Windows command files (UNIX shell scripts) to issue process-related requests to the Client Service. The Batch Console interprets (and runs) scripts that are written in a language that vaguely resembles Visual Basic.
Client Console	A graphical user interface from which you can connect to the Client service. From the Client Console you can start the **Client Configurator**, which lets you customize the layout of the relational database.

Databridge FileXtract

Databridge FileXtract is an application that allows you to clone and update flat files (page 212) that reside on Unisys MCP Servers. You can also use FileXtract with the Databridge Client to replicate this data. From the Client perspective, FileXtract data sources look like DMSII data sources.

FileXtract is bundled with Databridge Host software and includes several Reader libraries and other associated files.

Databridge DMSII Client

The Databridge DMSII Client is a product that installs on the MCP Server. DMSII Client clones and updates the DMSII database as another DMSII database. With DMSII Client, you can selectively replicate by filtering on both rows and columns.

Databridge Flat File Client

The Flat File Client (also known as PCSPAN) is a Windows implementation of the DBSPAN Accessory on the MCP. As is the case with DBSPAN, rather than update the secondary database, the Flat File Client creates data files that contain the data records for the updates. This approach is useful when a Databridge Client does not exist for a particular database or platform or when the data has to be transformed before being loaded into a secondary database.

The Flat Client has a very similar architecture to the relational database clients, such as the SQL Server and the Oracle Clients.

Databridge Twin

A mainframe program that replicates a DMSII database as another DMSII database. Twin typically runs on a development or departmental mainframe. As DMSII audit becomes available, updates to the primary database are applied to the secondary database.

2 Chapter 2: Engine

This chapter describes the Databridge Engine: what it is, how to configure it, and the commands you can use with it.

In this Chapter

How DBEngine Works

The Databridge Engine (also referred to as DBEngine) is a host library program that uses the DMSII Support Library to retrieve data records from the DMSII database for cloning.

Sources of information for database records for DBEngine are as follows:

- Database data sets
- DMSII audit trail

When an Accessory requests a clone (initial extraction), DBEngine reads each record in the specified data set and sends it to the Accessory. This is called data set extraction.

DBEngine's source of information about updates to database records is the DMSII audit trail, which contains before and after images for all changed records. DBEngine reads the audit trail and selects information for data sets the Accessory requested. It sends a copy of each updated record back to the Accessory. By default, DBEngine processes only closed audit files. If the Read Active Audit option is set to true, it will also access the current (in use) audit file.

DBEngine initializes when Databridge Server and other Accessories call it (a single copy of DBEngine is linked to each Accessory). DBEngine shares the same code stack as other copies of the Engine, but it does not share the same data stack.

DBEngine Visibility to Accessories and Usercodes

If you install Databridge Host and its Accessories under a privileged usercode as recommended, Accessories you use for replication can access DBEngine by title as needed. Accessories call entry points in DBEngine as specified in the Databridge API. The DBEngine component is shipped as a privileged program so that it can access guard files under all usercodes without any additional configuration.

However, if you need to run Databridge Host under multiple usercodes, or if the Databridge Accessories are installed under a different user code from DBEngine, you must do one of the following:

- Include a copy of DBEngine and its parameter file under each Accessory's usercode.
- Install Databridge Host without a usercode to a common disk family and use family substitution statements that reference that disk family. For information, see the Unisys documentation.
- Establish DBEngine as a system library. (See "Establish DBEngine as a System Library (SL)" on page 26.)

Establish DBEngine as a System Library (SL)

Use the following procedure to make DBEngine visible to Databridge Accessories and programs running under different user codes. For more information, see "DBEngine Visibility to Accessories and Usercodes" on page 26.

To establish DBEngine as an SL

1. Enter the following command from the Action line in MARC:

   ```
   SL DBENGINE = (usercode)OBJECT/DATABRIDGE/ENGINE ON familyname
   ```

2. To make the DBEngine parameter file available to Accessories under a different usercode, copy the DBEngine parameter file to the usercode of the Accessories.

To remove the function name assignment

Use this procedure after previously establishing it as a system library in the following situations:

- To access DBEngine by title (as per our recommendations)
- To install a new version or patch of DBEngine

1. From the Action line in MARC, enter the following command:

   ```
   SL - DBENGINE
   ```

   The host responds with the following:

   ```
   FUNCTION "Databridge Engine" IS NO LONGER ESTABLISHED
   ```

2. If you're installing a new version or patch and want to re-establish DBEngine as a system library, from the Action line in MARC enter the following command:

   ```
   SL DBENGINE = (usercode)OBJECT/DATABRIDGE/ENGINE ON familyname
   ```

Validation Checking

DBEngine validates the following:

- The property level in the DESCRIPTION file. This prevents the Databridge software from running with incompatible levels of the DMSII software.
- The audit level of each audit file. This ensures that it is compatible with the level of DMSII software for which DBEngine was compiled.
- The audit update level when the requested audit file is opened for tracking. This ensures that Databridge does not return records that have a different format than what the Accessory is expecting. Databridge starts by opening the oldest requested audit file first, and then moving forward; this ensures that the Accessory receives the updates in the proper order.
- That the update levels of the DMSUPPORT library and DESCRIPTION file match the audit file. When DBEngine must look for a DESCRIPTION file at a new update level, it checks for the file in the following sequence:
  - DESCRIPTION/*databasename/updatelevel*
  - *updatelevel*/DESCRIPTION/*databasename*
  - DESCRIPTION/*databasename*

  Likewise, when DBEngine is searching for DMSUPPORT, it will try:
  - *dmsupporttitle/updatelevel*
  - *updatelevel/dmsupporttitle*
  - *dmsupporttitle*

NOTE: To ensure that the DMSII DESCRIPTION file and the Support Library file include the correct update level for the database, run WFL/DATABRIDGE/BACKUPTAILORED after every DASDL update or reorganization. This WFL adds the update level to the last node of the DESCRIPTION and Support Library filenames. See "Chapter 10: Utilities" on page 177.

Configuring DBEngine Parameters

There are three types of DBEngine parameter files that you can configure, depending on your needs:

The common parameter file (DATA/ENGINE/CONTROL)	Use to specify availability, checkpoint frequency, etc., for all databases that do not set the parameter value in a database-specific parameter file or a logical database-specific parameter file.
A database-specific parameter file (DATA/ENGINE/*databasename*/CONTROL)	Use to specify availability, checkpoint frequency, etc., at the database level. Options in the database-specific parameter file override the options in the common parameter file.
A logical–database-specific parameter file (DATA/ENGINE/*databasename*/*logicaldatabasename*/CONTROL	Use this type of DBEngine parameter file to specify availability, checkpoint frequency, etc., at the logical database level. Options in the logical database-specific parameter file override the options in the database-specific parameter file and the options in the common parameter file.

DBEngine reads the database-specific parameter file and the logical–database-specific parameter file, if they exist, in addition to the common parameter file (DATA/ENGINE/CONTROL). The database-specific parameter file and the logical–database-specific parameter file can have the same options as the common parameter file. Options in the database-specific parameter file and the logical–database-specific parameter file override the options in the common parameter file. This allows you to configure different options for different databases (for example, to have different checkpoint frequencies).

Parameter files must be visible to DBEngine and in one of the following locations:

- Under the current usercode (that is, the Accessory's usercode)
- Under the DBEngine codefile's usercode and pack name

Parameter Change Limits

The Databridge Client and Enterprise Server can change the values of many of the parameters in this file. The parameter declaration can specify the allowable changes.

Boolean Parameter Change Limits

The syntax for Boolean parameters will have the following form.

```
parameter [ [ = ] default [ ONLY ] ]
```

where *default* is the default value before any changes by the Databridge Client or Enterprise Server. The default value can be any of the following:

```
TRUE
FALSE
YES
NO
```

If the parameter is specified without a default value it is equivalent to *parameter* = TRUE.

The ONLY keyword makes the parameter read-only.

Numeric Parameter Change Limits

The syntax for numeric parameters has the following form:

parameter [=] *default* [*limits*]

where *default* is the default value before any changes by the Databridge Client or Enterprise Server.

The optional *limits* clause can be any of the following. (The parentheses are required.)

```
(ALLOW min - max)
(ALLOW ANY)
(MAX max)
```

where *min* is the minimum value the parameter can be set to and *max* is the maximum. ALLOW ANY means there is no minimum or maximum restrictions. For the MAX *max* form there is no minimum value limitation.

If no *limits* clause is specified, there is no minimum value limitation but the maximum value is *default*.

Audit and Property Levels

CAUTION: Do not make changes to Audit level or Property level unless you are instructed to do so by Micro Focus. This information is for reference only.

Syntax:

```
Audit level = number

Property level = "numberstring"
```

Where	Is
number	An integer that represents the audit level DBEngine can use even if it was compiled with a different audit level.
"*numberstring*"	The value that represents the property level of the DESCRIPTION file that DBEngine can use even if it was compiled with a different property level.

Example:

Audit levels and property levels can have multiple entries, as in this example:

```
Audit level = 7
Property level = "040230"
Property level = "040710"
```

The DBEngine options that follow are available for these circumstances:

- When an audit file has a different audit level than the one with which DBEngine was compiled
- When the DESCRIPTION file has a different property level than any of the values in the Databridge internal property level list

In either of these cases, DBEngine checks its parameter file for an entry that indicates the software level for which DBEngine is compatible. If an audit level or property level mismatch remains after DBEngine checks the parameter file, DBEngine displays a message about the mismatch and returns an error.

Available From... To...

Syntax: AVAILABLE *day* FROM *time* TO *time*

Where	Is	
day	One of the following: ◆ Daily ◆ Weekdays ◆ day_of_week (Saturday, for example)	
time	One of the following:	
	hh:mm	Hours and minutes in 24-hour format, for example, 13:15 for 1:15 PM
	hh:mm AM	Hours and minutes in clock time, for example, 10:00 AM
	hh:mm PM	Hours and minutes in clock time, for example, 10:00 PM

Default: Disabled

The Available From…To…option limits when a Databridge Accessory can use DBEngine to access a database. If an Accessory attempts to access the database outside of the specified time interval, error 41, “Access to database denied” is returned.

Example:

The following example shows how you can specify multiple Available From...To... options:

```
Available weekdays from 9:20 PM to 11:50 PM
Available Saturday from 4:00 AM to 10:50 PM
```

If there are conflicting entries, DBEngine will use the least restrictive (most available) entry.

Checkpoint Frequency

The Checkpoint Frequency option can be based on the following:

- Size of a transaction group
- Elapsed time
- Pseudo-quietpoints
- Idle database

NOTE: The Checkpoint frequency option applies to all Accessories. Databridge Clients can override the DBEngine Checkpoint values using parameters in the Client configuration files.

Elapsed Time

Syntax: `CHECKPOINT [ CLIENT ] [ EVERY ] n [ limits ] [ SECONDS | MINUTES ]`

Default: `Checkpoint client every 0 seconds`

(disabled)

This Checkpoint option specifies the elapsed time (in seconds or minutes) when Databridge is processing the audit trail, not the time difference when the updates actually occurred.

When this option is specified, DBEngine checks the current time of day at each QPT. If that amount of time has elapsed since the last commit, DBEngine causes another commit.

Elapsed-time checkpoint frequency is useful for making transactions available to the client sooner when the mainframe is heavily loaded and replication slows.

Example:

```
Checkpoint client every 5 minutes
Checkpoint every 20 seconds
```

Idle Database

Syntax: `CHECKPOINT [ CLIENT ] [ DURING ] IDLE DATABASE [ [ = ] value [ ONLY ]`

Default: False

This option determines if DBEngine should do a commit if the audit trail ends with an End Transaction. If the database is idle (open for update but no programs have a transaction in progress) the last transaction group will not be committed because no Begin Transaction follows the last End Transaction. As soon as a program starts a transaction or DMSII does a syncpoint or controlpoint or the database is closed, that last transaction group will be committed.

If the site wants to ensure that the last transaction group before the database became idle is replicated, the recommended procedure is to run a program that forces a syncpoint. For databases that do not have the DASDL option REAPPLYCOMPLETED set, this will also ensure that the last transaction is actually written to the audit trail and the database.

Again, this procedure is necessary only if one or more programs have the database open update but none of them are actually doing any updates. If running a special program to force a syncpoint is not desirable in this situation, the site can set the following option in the DBEngine parameter file:

```
Checkpoint client during idle database = true
```

This will cause a commit if the last available audit record is an End Transaction and no other transactions are in progress.

Setting this option runs the risk that DMSII will not treat that position in the audit trail as a quiet point because it (incorrectly) writes the next Begin Transaction record with a transaction count other than 1. A halt/load recovery could then discard those committed transactions and the client database would not match the DMSII database.

Pseudo Quiet Points

Syntax:

```
CHECKPOINT [ CLIENT ] [ DURING ] LONG TRANSACTIONS [ = value [ ONLY ] ]
```

where *value* is `True`, `Yes`, `False`, or `No`.

Default:

```
Checkpoint client during long transactions = false only
```

When this option is true, DBEngine treats a pseudo quiet point as a normal quiet point and a possible place to commit a transaction group.

A pseudo quiet point occurs when DMS forces all programs out of transaction state in order to perform a sync point when the SYNCWAIT time specified in the DASDL is exceeded. If a program terminates (or there is a halt/load) before finishing a long transaction, DMS rolls back the entire long transaction, but the client database still has the committed portions of the partial transaction. DBEngine eventually reverses these updates when the recovery region of the audit is processed by sending update reversals. In these cases, the history tables will contain both the original update and its reversal.

The `only` option prevents Databridge Clients or Accessories from changing the LONG TRANSACTIONS option.

Transaction Group Size

Syntax:

```
CHECKPOINT [ CLIENT ] [ EVERY ] n [ limits ] [ AUDIT ] BLOCKS
CHECKPOINT [ CLIENT ] [ EVERY ] n [ limits ] TRANSACTIONS
CHECKPOINT [ CLIENT ] [ EVERY ] n [ limits ] [ UPDATE ] RECORDS
```

Checkpoint options can be combined with "or." For example:

```
CHECKPOINT [ CLIENT ] [ EVERY ] n [ limits ] [ AUDIT ] BLOCKS
     OR [ EVERY ] n [ limits ] TRANSACTIONS
     OR [ EVERY ] n [ limits ] [ UPDATE ] RECORDS
```

Where	Is
n	The default number of audit bocks or updates or transactions to be allowed in a transaction group. *n* can be any valid integer.
`limits`	See “Numeric Parameter Change Limits” on page 29 for an explanation of the limits option.

`CHECKPOINT`	A required word
`CLIENT`	An optional word
`DURING`	An optional word
`EVERY`	An optional word
`AUDIT`	An optional word

Where	Is
BLOCKS	This indicates that *n* refers to the number of audit blocks processed in a transaction group.
OR	A required word if you are checking combinations of checkpoint options.
UPDATE	An optional word
RECORDS	This indicates that *n* refers to the number of record updates in a transaction group.
TRANSACTIONS	This indicates that *n* refers to the number of transactions in a transaction group.

Default:

```
CHECKPOINT CLIENT EVERY 100 (ALLOW any) AUDIT BLOCKS
CHECKPOINT CLIENT EVERY 1000 (ALLOW any) RECORDS
CHECKPOINT CLIENT EVERY 0 TRANSACTIONS
```

(i.e. disabled)

NOTE: DATA/ENGINE/SAMPLE/CONTROL may have different settings.

This Checkpoint option specifies the size of a transaction group. DBEngine ends a transaction group at the first quiet point (natural or super) after reaching any Checkpoint value. This ensures that transaction groups are roughly the same size and reduces the overhead associated with ending transaction groups due to frequent quiet points.

Updated records in a transaction group are not available to Accessories until the end of the transaction group is reached and the whole group is committed to the client database. Therefore, smaller transaction groups make the individual updates available sooner.

The disadvantage of smaller transaction groups is that DBEngine must send state information records at the end of each group to reflect the new location in the audit trail. The Accessory must update its control tables with this state information, which is additional overhead. The target database might have some constraints on how many records can be updated in one group. If so, the Checkpoint option would need to be low enough to stay within those constraints.

Convert Reversals

Syntax: `CONVERT REVERSALS TO UPDATES = value [ ONLY ]`

where *value* is `True, Yes, False,` or `No`.

Default: `False`

If this option is true, reversals are treated as normal updates such that the original update and its reversal will both be sent to the Databridge Client or the Accessory. This removes the need to abort and reposition the audit trail when an aborted transaction is encountered. The Client database will show the original update and its reversal. A history table will include both.

DMSII Program Titles

Syntax:

```
DMCONTROL [ = ] "filetitle"
DMUTILITY [ = ] "filetitle"
DMRECOVERY [ = ] "filetitle"
```

Default:

Titles as specified in WFL/DATABRIDGE/INCLUDE/SSRTITLES

The DMSII Program Titles option allows you to specify a different title for the DMCONTROL, DMUTILITY, and DMRECOVERY programs. This option allows you to use nonstandard names for either program, in addition to specifying usercodes and packnames.

We strongly suggest, however, that you use WFL/DATABRIDGE/INCLUDE/SSRTITLES to specify these titles instead, as this allows you to specify different titles for different MCP and DMSII releases.

Example:

```
DMUTILITY "(581)SYSTEM/DMUTILITY ON SYSPACK"
```

Dynamic Names

Syntax: `DYNAMIC NAMES [[ = ] value ]`

where *value* is either `True, Yes, False,` or `No`.

Default: `False`

If false or no, names of the Extract Workers are EXTRACT/WORKER/*n* where *n* is a relative Worker number.

If true or yes, the names of the Extract Workers will include the database name and the current data set name in this form: EXTRACT/WORKER*n/databasename/structurenumber*. *Note there is no slash between WORKER and n.*

Enterprise Workers

Syntax: `ENTERPRISE WORKERS = number [ limits ]`

where *number* is the default number of Databridge Enterprise extract threads. For an explanation of limits see “Numeric Parameter Change Limits” on page 29.

Default:

```
Enterprise Workers = 20 (allow any)
```

The Enterprise Workers option enables Databridge Enterprise to process multiple Worker threads to perform data set extracts during a clone. When more than one Worker thread is running, the Databridge Client receives records interspersed from multiple data sets.

The actual number of Databridge Enterprise threads will not exceed the number of data sets to extract.

Links

Syntax: `LINKS [ = ] value [ ONLY ]`

where *value* is either `True`, `Yes`, `False`, or `No`.

Default: `False`

The LINKS option enables DBEngine to process link data items for unsectioned data sets.

If you enable the `enable_dms_links` parameter in the Databridge Client configuration file, you must also enable the DBEngine LINKS option.

NOTE: The Span and Snapshot Accessories do not support links and will discard any link values they receive.

Manual Compile

Syntax: `MANUAL COMPILE [[ = ] value ]`

where *value* is either `True, Yes, False`, or `No`.

Default: `False`

If false or no, DBEngine will automatically recompile DBSupport and DBDMSIISupport when needed for a particular database update level.

If true or yes, DBEngine will return the error message "Compile of *title* failed", and you will need to manually initiate the compile of that program. This is useful when the DMALGOL compiler is not resident on the system where DBEngine is running.

Mirrored Audit

Syntax: `MIRRORED AUDIT [ AND DATABASE ] FOR sourcename AT hostname PORT "portnumber"`

Where	Is
sourcename	The name of a DBServer SOURCE on the primary system (where the live database is)
hostname	The primary system (where the live database is)
portnumber	The port number for communicating with DBServer on the primary system

The Mirrored Audit option allows an Accessory to process transaction groups on a secondary system containing the mirrored audit files with minimal impact on the primary system. Special disk hardware or software duplicates the audit files on the secondary system. DBEngine will communicate with DBServer on the primary system to determine the current (logical) end of the audit trail.

Put the MIRRORED option in a *database-specific* DBEngine parameter file (DATA/ENGINE/*database*/CONTROL). If you put it in DATA/ENGINE/CONTROL, then no matter what database the Accessory wants to use, it will always try to get the information from the specified remote source.

Examples:

```
Mirrored Audit and Database for BANKDB at PROD port "3000"
Mirrored Audit for CUSTOMERDB at GREENHOST port "4101"
```

Print Statistics Report

Syntax: `[ PRINT ] STATISTICS [ = ] value [ ONLY ]`

where *value* is either True, Yes, False, or No.

Default: `False`

If true or yes, DBEngine will print a statistics report when it finishes. The optional word "only" ensures that the Databridge Client or Accessory cannot change this option.

ReadAhead

Syntax: `READAHEAD [ = ] value [ ONLY ]`

where *value* is either `True`, `Yes`, `False`, or `No`.

Default: `False`

The ReadAhead option can be used to increase throughput when DBEngine accesses a remote audit file via DBServer, such as when running Databridge Span with the “SOURCE *sourcename* ...” option, or the “MIRRORED AUDIT ...” DBEngine option. If the ReadAhead option is enabled, DBEngine will request the next audit region before the Accessory actually needs it. This overlaps the network time and the processing time for greater throughput. The optional word "only" ensures that the Databridge Client or Accessory cannot change this option.

Read Active Audit

Syntax: `READ ACTIVE AUDIT [ = ] value [ ONLY ]`

where *value* is either `True, Yes, False, or No.`

Default: `False`

If True or Yes, Databridge will return updates during tracking from the active audit file. Databridge reads the active audit file as needed during the extract and fixup phases. This parameter may only be set to True on installations running SSR 50.1 or later.

The optional word "only" ensures that the Databridge Client or a Databridge Accessory cannot change this option.

IMPORTANT: This option must be set to true for Databridge Enterprise Server to read the active audit file during tracking.

Workers

Syntax:

`WORKERS = number [ limits ]`

where *number* is the default number of Workers. For an explanation of limits see "Numeric Parameter Change Limits" on page 29.

Default:

```
Workers = 10 (allow 1 - 4095)
```

The Workers option enables DBEngine to process one or more Extract Worker stacks to perform the data set extracts during a clone. On large systems with multiple processors and/or extra available resources, this might increase the speed of the cloning process. When there is more than one Worker running, the Accessory or Databridge Client receives records interspersed from multiple data sets.

The AX `WORKERS = number` command can dynamically change the number of Extract Workers allowed, but must not exceed *limits*. An Accessory or Databridge Client can also specify the number of Workers by calling the DBParameters entry point, but it must not exceed the specified *limits*.

The actual number of Workers running will not exceed the number of data sets to extract.

Sample DBEngine Parameter File

```
%----------------------------------------------------------------------

%
% (C) Copyright 2019 Micro Focus or one of its affiliates.
%
% Module: DATA/ENGINE/SAMPLE/CONTROL
%
% Project: Databridge
%
% Description: Databridge Engine Parameter File
%
% (C) Copyright 2019 Micro Focus or one of its affiliates.
%
%----------------------------------------------------------------------


        % Software compatibilities

   Audit level = 10
   Property level = "050710"

        % Frequency of commits

   Checkpoint client every 100 (allow any) audit blocks
   Checkpoint client every 1000 (allow 1 - 9999999) records
 % Checkpoint client every 200 (allow any) transactions
 % Checkpoint client every 10 (allow 1 - 36000) seconds
   Checkpoint client during long transactions = false only
```

```
  Checkpoint client during idle database = false

       % Number of concurrent Extract workers

  Workers = 4 (max 10)
  Dynamic names = false    % Workers named .../database/dataset

  Enterprise Workers = 20 (max 100)   % threads reading datasets

       % Restrict access to Databridge to certain times

% Available daily from 11:30 PM to 7:00 AM
%          Monday from 12:00 PM to 1:30 PM
% Available Saturday from 4:00 AM to 10:50 PM
% Available weekdays from 9:20 PM to 3:00 AM

       % When retrieving audit regions from another host
       % across a network, ReadAhead will send a request for
       % the next region before the Accessory asks for it.

  ReadAhead = true

       % Print statistics report at EOJ

  Print Statistics = false

       % Include/exclude link data items

  Links = false

       % Access/don't access the active (current) audit file

  Read active audit = false

       % Treat reversals as normal updates

  Convert reversals to updates = false

       % Enable/disable automatic recompile of DBSupport and
       % DMSIISupport

  Manual compile = false

       % DMSII program titles
       % (We recommend using WFL/DATABRIDGE/INCLUDE/SSRTITLES instead.)

  %- DMCONTROL  "*SYSTEM/DMCONTROL ON DISK"
  %- DMUTILITY  "*SYSTEM/DMUTILITY ON DISK"
  %- DMRECOVERY "*SYSTEM/DMRECOVERY ON DISK"
```

```
        % Mirroring
        %  WARNING: The MIRRORED option should only appear in a
        %           database-specific DBEngine control file, i.e.
        %           DATA/ENGINE/<database>/CONTROL.

%         MIRRORED AUDIT [ AND DATABASE ] FOR <sourcename>
%                 AT "<hostname>" PORT "<portnumber>"

% <hostname> is the primary system, i.e. where the live database is.
% <sourcename> is name of a DBServer SOURCE on that primary system.

% Mirrored Audit and Database for BANKDB at PROD port "3000"
%
% Mirrored Audit for CUSTOMERDB at GREENHOST port "4101"
```

DBEngine Commands

DBEngine responds to the following AX commands. The mix number in the AX command must be the job number of the DBEngine itself, not an Accessory calling DBEngine. You can use the ODT or MARC command `LIBS NAME =ENGINE=` to find the DBEngine job number

After entering any AX command, enter SMSG from MARC or MSG from the ODT to see the response.

The DBEngine responses appear in the system messages.

`AX CLOSE`	If DBEngine has the database open during tracking, this command will close it. This allows other programs exclusive use of the database.
`AX COMMITS`	Displays the same commit-related statistics that would appear on the printed statistics report. The number after the "#" is the count of samples of that category. The numbers at the end of the displayed line are the minimum, maximum, and average.
`AX DEBUG`	Displays whether debug tracing is enabled. `AX DEBUG ON` enables debugging. `AX DEBUG OFF` disables it. The `AX DEBUG` command is available only when using the debug version of DBEngine.
`AX HELP`	Displays the available AX commands. The mix number of an Extract Worker can be used in the `AX HELP` command to display the valid AX commands for an Extract Worker.
`AX QUIT`	Terminates tracking. It causes the DBReadTranGroup entry point to exit at the next quiet point. Other copies of the DBEngine library, however, are still available for other Databridge Accessories. If DBReadTranGroup is waiting for more audit to become available, the `QUIT` command will cause it to quit waiting immediately.
`AX SIZES`	Displays the same size-related statistics that would appear on the printed statistics report. The number after the "#" is the count of samples of that category. The numbers at the end of the displayed line.
`AX STATUS`	Displays status information such as the current audit location being tracked, number of audit blocks, updates, and checkpoints. When the mix number of an Extract Worker is used in the `AX STATUS` command, the response shows the number of records extracted in the current data set, the number of data sets completed, and the total number of records extracted by the Extract Worker

DBEngine API

This API provides access to DBEngine and the Support Library to retrieve structural information, layout information, and data from audit files and a DMSII database. All Databridge Accessories use the Databridge API. Following are examples of what you can do with the Databridge API at your site:

- EDP Auditors can develop Accessories to monitor changes being made to all or selected records.
- Applications can be developed to generate summary reports of changes made to the database (for example, net change to a dollar amount field).
- Applications can be developed to report history records of all changes made to the primary database.

Sample Accessories for the Databridge API include the following:

- SQLGen (ALGOL)
- COBOLGen (ALGOL)
- DASDLGen (ALGOL)
- AuditClose (ALGOL)
- ReadDoc (ALGOL)
- ExtractAddress (COBOL)

Source is available for each of these. Each serves as an example of how you can use the Databridge API. For more about the Databridge API, see the *Databridge Programmer's Reference*.

Database Definition Retrieval

DBEngine provides several entry points for retrieving database definition information. These entry points are useful for programs that need to format data records or facilitate mapping from the DMSII database to the client database. The predefined formatting procedures in the Support Library use these entry points to load the arrays used for driving the formatting routines. For more information about entry points, refer to the *Databridge Programmer's Reference*.

3 Chapter 3: DMSII Support

This chapter describes everything that pertains to the Databridge DMSII Support Library.

In this Chapter

What is DMSII Support?

DMSII Support is a Databridge library that retrieves data records from the DMSII database for cloning. The Databridge Engine links to this library to perform database functions such as reading records, switching the audit file, and getting database statistics.

This library is a more standardized approach for accessing the DMSII database than the previous approach which used procedure references that pointed directly at procedures in the actual DMSII database stack (Accessroutines). The latter approach left Databridge vulnerable to any changes made to the DMSII software architecture.

Recompiling the DMSII Support Library

The Databridge Engine will automatically start WFL/DATABRIDGE/COMP to compile DMSII Support as needed, or you can do it manually using WFL/DATABRIDGE/COMP. For example, the Engine will recompile the DMSII Support Library if the DASDL is updated.

The advantage of compiling it manually is that you can compile all of the Databridge software at once for a new update level. For example, you can compile the DMSupport Library, a tailored DBSupport Library, and the DMSII Support Library under a different usercode or disk family and then switch them all over at once.

To recompile the DMSII Support Library, use the following sample command:

```
START WFL/DATABRIDGE/COMP ("DMSIISUPPORT", "DASDLname")
```

where "*DASDLname*" is the title of your DMSII database DESCRIPTION file without the DESCRIPTION node.

NOTE: Be sure you have updated WFL/DATABRIDGE/INCLUDE/SSRTITLES with the correct titles of the compilers for each system software release. Any compiler titles in the DBEngine parameter files are ignored.

Modifying the DMSII Support Library

The DMSII Support Library has no parameters or configuration files. The source code must not be modified in any way or Databridge will not work as expected.

Source and Object Code

The source code for the DMSII Support Library is SYMBOL/DATABRIDGE/DMSIISUPPORT.

The object code is OBJECT/DATABRIDGE/DMSIISUPPORT/*dbname/usercode*

where *dbname* is the name of your database and *usercode* is the usercode of the DESCRIPTION file.

For example:

```
OBJECT/DATABRIDGE/DMSIISUPPORT/BANKDB/PROD
```

4 Chapter 4: Support Library

This chapter describes the Databridge Support Library and the GenFormat program and how you can use them to create tailored support libraries.

In this Chapter

Understanding the Support Library

The Support Library provides translation, filtering, and formatting services for Databridge Accessories. The Accessories pass the records they receive from DBEngine to entry points in the Support Library for filtering and then, if the record successfully passes through the filter, for formatting.

Although you can use the Support Library "as is" or with minor formatting customizations, it is highly recommended that you create a tailored support library for each database you plan to clone and update.

IMPORTANT: In a nontailored support library, you cannot use any data set or data item names, nor can you use any SELECT statements. Therefore, you must create a tailored support library to create effective filters. For example, if a filter refers to specific data sets, such as BANK1 or OHIOCUSTOMERS, you must compile the Support Library as a tailored library.

A tailored support library is a database-specific version of the Support Library and works the same as the generic Support Library. However, tailoring a library for a specific database enhances performance by increasing throughput. A nontailored Support Library uses interpretive routines for each data set. You can tailor the Support Library for the following:

- Translation
- Filtering
- Formatting
- Reformatting (ALTERations)
- Transforming (VIRTUAL data sets)

You can customize the Support Library by using the GenFormat program explained in the next section. GenFormat generates source code for filters and formats that you can include when you compile the Support Library. It is highly recommended that you use GenFormat to generate a tailored (database-specific) support library for each database you plan to replicate.

Tailoring a Support Library

A tailored support library is a database-specific version of the Support Library. Tailoring a support library enhances performance by increasing throughput.

To tailor a support library

1. Use CANDE or another editor to get the GenFormat parameter file DATA/GENFORMAT/SAMPLE/CONTROL.

   For a general description of the GenFormat parameter file, see "GenFormat Parameter File" on page 46.

2. Save the file with a new name, as follows:

   `DATA/GENFORMAT/`*`databasename`*`/CONTROL`

   where *databasename* is the name of the database for which you are creating the tailored support library.

   Example: `DATA/GENFORMAT/PRODUCTDB/CONTROL`

   NOTE: You can also create an alternate parameter file when you compile the support library, as explained in step 5.

   ---

3. Modify this new file (`DATA/GENFORMAT/`*`databasename`*`/CONTROL`) for any of the following:
   - Formatting routines. See "Creating a Format" on page 66.
   - Filtering routines. See "Creating a Filter" on page 76.
   - Transforms. See "Transforms" on page 83.
   - Error manager. See "Error Manager" on page 83.
   - Data item conversion. See "ALTER and VIRTUAL Data Sets" on page 83.
   - Virtual data sets. See "ALTER and VIRTUAL Data Sets" on page 83.
   - Primary keys. See "Creating a Primary Key" on page 84.
   - Translation tables. See "Creating and Using Translation Tables" on page 85.
4. Save `DATA/GENFORMAT/`*`databasename`*`/CONTROL`.
5. To create and compile the new support library, including layout tables for each data set in the designated databases, start `WFL/DATABRIDGE/COMP`:

   ```
   START WFL/DATABRIDGE/COMP ("SUPPORT", "databasename"
   [,"logicaldatabasename"])
   ```

Where	Is
"*SUPPORT*"	The literal that represents the Support Library program. The quotation marks are required.
"*databasename*"	The name of the database for which you are creating the tailored support library. The database name can include a usercode and pack, which are used to locate the database DESCRIPTION file, as follows: "(*usercode*)*databasename* ON *packname*" The quotation marks are required.
"*logicaldatabasename*"	An optional name of an alternate database when you want to create one of the following: ◆ A tailored support library for the logical database. ◆ An alternate parameter file for the physical database, which is useful when you want to generate multiple support libraries for the same database but with different parameter files. In either case, the GenFormat file must be titled DATA/GENFORMAT/*databasename/logicaldatabasename/CONTROL*

CAUTION: You cannot compile tailored support libraries for more than one database of a given name in one usercode—the second tailored support library will replace the first.

This results in the new tailored support library titled as follows:

OBJECT/DATABRIDGE/SUPPORT/*databasename*

— or —

OBJECT/DATABRIDGE/SUPPORT/*databasename/logicaldatabasename*

The data set-specific layout tables in the tailored support library have the offsets and sizes of individual data items hardcoded. In addition to creating this file, the WFL will create a copy that adds the update level as the last node. This makes the library more easily identifiable and helps prevent accidental deletion of a version of the Support Library that may still be needed. For example, the Support Library for BANKDB update level 4 will be copied as follows:

OBJECT/DATABRIDGE/SUPPORT/BANKDB/4

To use the tailored support library

- Specify the new tailored support library file title for the SUPPORT parameter in the appropriate Accessory parameter file:
  - “Lister Parameter File” on page 167
  - “Snapshot Parameter File” on page 150
  - “Span Parameter File” on page 113
  - “DBServer Parameter File” on page 90

Understanding GenFormat

Use the GenFormat program to create formats, filters, and translation tables—without programming—that you can compile into a tailored support library. Specifically, you can use the GenFormat program (also referred to as DBGenFormat) to include the following in your tailored support library:

- Alter data sets*
- Virtual data sets*
- Primary keys
- Reformatting
- Translation tables
- Transforms*
- Formatting
- Filtering
- Error management*
- Startup and shutdown*

* These items require additional programming. Refer to the *Databridge Programmer's Reference* for instructions.

To use the GenFormat program, edit the GenFormat parameter file using CANDE or some other text editor, and then compile the Support Library using WFL/DATABRIDGE/COMP. The parameter file can contain as many formats and filters as you want.

The GenFormat program interprets your textual descriptions of filters, formats, etc., in its parameter file, generates ALGOL source code patches, and includes them in a tailored support library. The tailored support library implements the filter and format routines you specify in a Databridge Accessory parameter file (or in your user-written Accessories).

For instructions, see "Tailoring a Support Library" on page 44.

GenFormat Parameter File

Each part of the GenFormat parameter file is described on the following pages, along with an excerpt from the actual sample GenFormat parameter file (DATA/GENFORMAT/SAMPLE/CONTROL). This topic describes the format that the GenFormat parameter file requires.

- Declarations in the parameter file must be in the following order:

  1—ALTERs and VIRTUALs

  2—PRIMARY KEYs

  3—REFORMAT, TRANSLATIONs, TRANSFORMs, FORMATs, FILTERs, ERROR MANAGER, STARTUP, and SHUTDOWN
- You can list the options for a reformat, translation, transform, format, filter, error manager, startup, or shutdown in any order.
- You can list multiple options on a single line.

- You can split options across multiple lines.
- There is no termination character.
- There is no continuation character.
- The comment character is the percent sign (%). The comment character can appear anywhere on a line and anything after the comment character is ignored.
- The default format for floating point real numbers is SCIENTIFIC (11), which has the following format:

  ```
  #.#####E-##
  ```

- You can use GenFormat keywords as filter or format names. This includes, for example, the keywords TRANSLATION, FORMAT, and FILTER. To use keywords as filter or format names, enclose the name in quotation marks. To use FORMAT as a format name, you would enter the following:

  ```
  FORMAT "FORMAT"
  ```

Sample DBGenFormat Parameter File

```
%----------------------------------------------------------------------
%
% (C) Copyright 2019 Micro Focus or one of its affiliates.
%
% Module: DATA/GENFORMAT/SAMPLE/CONTROL
%
% Project: Databridge
%
% Description: Databridge GenFormat Sample Parameter File
%
% (C) Copyright 2019 Micro Focus or one of its affiliates.
%
%----------------------------------------------------------------------%
% This is the input to the Databridge Format Generator program.
%
% (Upper/lower case is not significant.)
%
% This file has these main sections:
%       Alterations
%       Virtuals
%       Primary keys
%       Reformat
%       Translations
%       Transforms
%       Formats
%       Filters
%       Error Manager
%       Startup
%       Shutdown
% Alterations and Virtuals must precede the other sections.
% Translations, Transforms, Formats, and Filters can be intermixed with
% each other but cannot precede any Primary keys.
%
% When processing an update, the sequence is:
```

```
%
% 1. Apply the row filter (the FILTER's WHERE ... condition) and stop
%       processing if the condition is false.
%
% 2. Call the TRANSFORM.
%
% 3. The TRANSFORM calls the FORMAT zero or more times for real and/or
%       VIRTUAL records.
%
% 4. The FORMAT steps through the list of data items of the column
%       filter (the FILTER's SELECT ...).
%
% 5. For each data item, if the data item is ALTERed, call the REFORMAT
%       routine, otherwise format it according to its data type. If it
%       is an ALPHA item, use the TRANSLATION table, if any. The
%       reformatted values for ALTERed items are placed at the end of
%       the record and formatted according to their data type just like
%       unaltered items.
%
%
% Lines below beginning with %* indicate example declarations that you
% can uncomment and use immediately if you are compiling a tailored
% Support Library for the sample database, BANKDB.
%
%=============================================================================
%                               Alterations
%                         (Data item reformatting)
%                         ========================
%
% In this section, list any data items that require transformation.
% When the formatting routines encounter a data item in the list, they
% will call the REFORMAT routine, which is either a procedure in the
% Support Library (see PATCH/DATABRIDGE/SAMPLE/SUPPORT/REFORMAT) or an
% entry point in a library (see SYMBOL/DATABRIDGE/SAMPLE/REFORMAT),
% before performing normal formatting. (The "Reformat" section below
% specifies where the REFORMAT routine resides.)
%
% For example, if dates are stored in a non-standard layout, the
% formatting routines could call REFORMAT to reformat the dates into
% a standard layout.
%
% The formatting routines will pass the number in brackets for the data
% item to REFORMAT, which can use the value to determine the type of
% formatting required.
%
%       ALTER <dataset>
%               (
%               [<uservalue1>]
%                       <original dataname1> <new DASDL definition>
%
%               [<uservalue2>]
%                       <original dataname2> <new DASDL definition>
%
%               [REDEFINE]
%                       <original dataname3> <new DASDL definition>
```

```
%
%               [DEFINE <uservalue3>]
%                       <new dataname4> <new DASDL definition>
%
%                       ...
%               );
%
%
% Note that the numbers do not have to be unique. In fact, all data
% items of a particular type, e.g. days-since-1900, would probably have
% the same value since a single routine could convert any of them.
%
%       Example:
%*          ALTER BRANCH
%*              (
%*              [1]    TS          ALPHA (30);  % was REAL
%*              [2]    BRANCH-ID   NUMBER (6);  % was NUMBER (4)
%*              [3]    BRANCH-AD1  GROUP        % was ALPHA (40)
%*                              (
%*                              BR-ADD1   ALPHA (30);
%*                              BR-ADD2   ALPHA (30);
%*                              BR-CITY   ALPHA (20);
%*                              BR-REGION ALPHA (15);
%*                              );
%*              );
%*
%*          ALTER BANK
%*              (
%*              [REDEFINE] BANK-ADDR3 GROUP     % was ALPHA (30)
%*                              (
%*                              BR-CITY   ALPHA (18);
%*                              BR-STATE  ALPHA (2);
%*                              BR-ZIP    ALPHA (10);
%*                              );
%*              [DEFINE 6] BANK-PRES  ALPHA (40); % new item
%*              [DEFINE 7] BANK-PHONE ALPHA (12); % new item
%*              );
%*
%*          ALTER HISTORY
%*              (                                           % was:
%*              [REDEFINE]      TRAN-DATE NUMBER (YYMMDD); % NUMBER (6)
%*              [REDEFINE]      TRAN-TIME NUMBER (HHMMSS); % NUMBER (6)
%*              [REDEFINE]      PROC-DATE NUMBER (YYMMDD); % NUMBER (6)
%*              [REDEFINE]      TS        TIME_6;          % REAL
%*              );
%*
%*
%*        ALTER EMPLOYEES
%*              (
%*      % Alter multiple adjacent data items as a single item.
%*
%*              [REDEFINE]      EMP-LAST-NAME,
%*                              EMP-FIRST-NAME
%*                          AS EMP-PERSONAL    ALPHA (32);
%*              );
```

```
%
% alterations go here ...
%=============================================================================
%                               Virtuals
%                          (Virtual datasets)
%                          ==================
%
%       These datasets will appear as normal datasets to Accessories
%       but do not actually exist in the DMSII database. Special
%       Transforms must "create" records in them.
%
%       VIRTUAL <datasetname> # <strnum> POPULATION <est. recs.>
%                DERIVED FROM <datasetlist>
%               (
%               <dataitems>
%               );
%
%       Example:
%
%*              VIRTUAL ADDRESS #79 POPULATION 100000
%*
%*                      DERIVED FROM BANK, CUSTOMER
%*
%*                      (ADDR-BANK-ID   NUMBER (4);
%*                       ADDR-CUST-ID   NUMBER (8);
%*                       ADDR-LINE-NBR  NUMBER (1);
%*                       ADDR-LINE      ALPHA (30);
%*                      );
%
%
%       The POPULATION clause is optional but strongly recommended. The
%       default value is 1000000.
%
%       The DERIVED FROM clause causes GenFormat to generate defines
%       and variables that the transform can use to build virtual
%       records. We recommend that you use the sample transform
%       PATCH/DATABRIDGE/SAMPLE/SUPPORT/VIRTUAL
%       as a starting point for writing the transform.
% virtuals go here ...
%=============================================================================
%                       Primary Keys
%                       ============
%
% The PRIMARY KEY declarations below provide Databridge with a unique
% key for the designated datasets. This is useful for datasets that do
% not have a NO DUPLICATES set declared (but *could* have one declared).
%
% Syntax:
%
%       PRIMARY KEY OF <dataset> IS ( <key items> )
%
% (The keywords PRIMARY, OF, and IS are optional.)
%
%       Examples:
%
```

```
%*              Primary Key of CUSTOMER is (CUST-ID)
%*
%*              Key ADDRESS (ADDR-BANK-ID, ADDR-CUST-ID, ADDR-LINE-NBR);
%
% Databridge will *not* confirm that the specified key is unique.
%
%       Primary Key declarations go here ...
%
%
%=====================================================================
%                       Reformat
%                       ========
%
%       This section specifies the location of the REFORMAT routine
%       that will convert any ALTERed data items.
%
%       An INTERNAL REFORMAT specifies a patchfile containing a REFORMAT
%       routine that converts ALTERed data items. This patchfile will be
%       compiled into the Support Library and called directly from the
%       formatting routines. The patchfile can contain "global" as
%       variables needed. This method makes the DBInterface definitions and
%       Engine entry points available to the REFORMAT routine without
%       having to declare them ($ INCLUDE "SYMBOL/DATABRIDGE/INTERFACE)
%       or explicitly linking to DBEngine. The patchfile should not
%       include an EXPORT declaration.
%
%       Syntax:
%               INTERNAL REFORMAT IN "<patchfiletitle>"
%
%       Example:
%*              INTERNAL REFORMAT
%*                      IN "PATCH/DATABRIDGE/SAMPLE/SUPPORT/REFORMAT"
%
%       An EXTERNAL REFORMAT indicates that the REFORMAT routine
%       resides in a separate library program, where it is declared as
%       an entry point and EXPORTed. In this method, the REFORMAT
%       routine must have $ SET INCLUDE_ENGINE (and INCLUDE_SUPPORT if
%       needed) as well as $ INCLUDE "SYMBOL/DATABRIDGE/INTERFACE".
%       The Support Library will ensure that the external REFORMAT
%       library is linked to the same instance of DBEngine as the
%       Accessory and Support.
%
%       Syntax:
%               EXTERNAL REFORMAT IN "<librarytitle>"
%
%       Example:
%               EXTERNAL REFORMAT IN "OBJECT/DATABRIDGE/REFORMAT"
%
%       You can specify only one REFORMAT declaration.
%
%       The default is
%               EXTERNAL REFORMAT IN "OBJECT/DATABRIDGE/REFORMAT".
%
%       The recommended method is to use INTERNAL REFORMAT ....
%
```

```
%
%====================================================================
%                       Translations
%                       ============
%
% The formatting routines use these translation tables to translate
% individual EBCDIC characters in text (ALPHA) data items. The
% format (declared below) specifies which table to use.
%
% Syntax:
%       TRANSLATION tablename
%               sourcevalue -> destinationvalue
%               ...
%
%       sourcevalue and destinationvalue can be the decimal value of
%       a character, or the hexadecimal value of a character, or a
%       quoted character. The following are equivalent.
%               193  -> 194
%               0xC1 -> 194
%               "A"  -> 194
%               193  -> 0xC2
%               0xC1 -> 0xC2
%               "A"  -> 0xC2
%               193  -> "B"
%               0xC1 -> "B"
%               "A"  -> "B"
%
% You can comment out each of the TRANSLATION tables declared below
% unless you have a FORMAT that explicitly references it.
%====================================================================
Translation CRToLF   % Convert Carriage Return to Line Feed
037 -> 013  % cr -> lf
Translation UnitedKingdom   % United Kingdom
123 -> 068  % # -> ú
Translation Denmark   % Denmark
161 -> 087  % ~ -> »
124 -> 138  % @ -> -
090 -> 139  % ] -> +
074 -> 140  % [ -> ¦
224 -> 176  % \ -> +
095 -> 180  % ^ -> _
121 -> 188  % ` -> S
208 -> 189  % } -> s
192 -> 190  % { -> µ
106 -> 238  % | -> °
161 -> 252  % ~ -> n
Translation Finland   % Finland
091 -> 069  % $ -> ñ
074 -> 138  % [ -> -
090 -> 139  % ] -> +
124 -> 143  % @ -> +
224 -> 174  % \ -> +
095 -> 180  % ^ -> _
192 -> 188  % { -> S
208 -> 189  % } -> s
```

```
121 -> 203  % ` -> T
106 -> 236  % | -> ÷
161 -> 252  % ~ -> n
Translation France   % France
123 -> 068  % # -> ú
090 -> 072  % ] -> º
161 -> 073  % ~ -> ¿
074 -> 088  % [ -> ¦
124 -> 184  % @ -> a
224 -> 191  % \ -> t
208 -> 202  % } -> F
192 -> 203  % { -> T
106 -> 239  % | -> ·
Translation Germany   % Germany
036 -> 192  % ? -> {
124 -> 072  % @ -> º
074 -> 138  % [ -> -
224 -> 174  % \ -> +
090 -> 180  % ] -> _
161 -> 183  % ~ -> ¯
192 -> 188  % { -> S
106 -> 236  % | -> ÷
208 -> 252  % } -> n
Translation Italy   % Italy
224 -> 123  % \ -> #
123 -> 068  % # -> ú
124 -> 072  % @ -> º
074 -> 088  % [ -> ¦
192 -> 184  % { -> a
224 -> 191  % \ -> t
208 -> 202  % } -> F
090 -> 203  % ] -> T
161 -> 206  % ~ -> 8
106 -> 222  % | -> =
121 -> 239  % ` -> ·
Translation Norway   % Norway
238 -> 106  % ° -> |
252 -> 161  % n -> ~
161 -> 087  % ~ -> »
124 -> 138  % @ -> -
090 -> 139  % ] -> +
074 -> 140  % [ -> ¦
224 -> 176  % \ -> +
095 -> 180  % ^ -> _
121 -> 188  % ` -> S
208 -> 189  % } -> s
192 -> 190  % { -> µ
106 -> 238  % | -> °
161 -> 252  % ~ -> n
Translation Spain   % Spain
074 -> 066  % [ -> í
123 -> 068  % # -> ú
124 -> 072  % @ -> º
192 -> 088  % { -> ¦
090 -> 117  % ] -> +
```

```
224 -> 160  % \ -> -
208 -> 191  % } -> t
106 -> 221  % | -> ±
Translation Sweden   % Sweden
091 -> 069  % $ -> ñ
074 -> 138  % [ -> -
090 -> 139  % ] -> +
124 -> 143  % @ -> +
224 -> 174  % \ -> +
095 -> 180  % ^ -> _
192 -> 188  % { -> S
208 -> 189  % } -> s
121 -> 203  % ` -> T
106 -> 236  % | -> ÷
161 -> 252  % ~ -> n
Translation SwissFrench   % SwissFrench
123 -> 068  % # -> ú
090 -> 072  % ] -> º
161 -> 073  % ~ -> ¿
074 -> 088  % [ -> ¦
124 -> 184  % @ -> a
224 -> 191  % \ -> t
208 -> 202  % } -> F
192 -> 203  % { -> T
106 -> 239  % | -> ·
Translation SwissGerman   % SwissGerman
124 -> 072  % @ -> º
074 -> 138  % [ -> -
224 -> 174  % \ -> +
090 -> 180  % ] -> ¯
161 -> 183  % ~ -> ¯
192 -> 188  % { -> S
106 -> 236  % | -> ÷
208 -> 252  % } -> n
Translation DenmarkEBCDIC   % Denmark EBCDIC
124 -> 183  % @ -> ¯
066 -> 252  % í -> n
065 -> 250  % á -> ·
224 -> 251  % \ -> v
095 -> 187  % ^ -> p
121 -> 100  % ` -> ¦
208 -> 235  % } -> )
192 -> 225  % { -> ƒ
079 -> 234  % ! -> (
161 -> 115  % ~ -> +
Translation FinlandEBCDIC   % Finland EBCDIC
091 -> 070  % $ -> Ñ
065 -> 183  % á -> ¯
066 -> 252  % í -> n
124 -> 204  % @ -> O
224 -> 186  % \ -> G
095 -> 187  % ^ -> p
192 -> 100  % { -> ¦
208 -> 235  % } -> )
121 -> 105  % ` -> ¦
```

```
079 -> 120  % ! -> -
161 -> 115  % ~ -> +
Translation FranceEBCDIC   % France EBCDIC
123 -> 067  % # -> ó
066 -> 072  % í -> º
161 -> 127  % ~ -> "
065 -> 081  % á -> ¬
124 -> 103  % @ -> +
224 -> 236  % \ -> ÷
208 -> 104  % } -> +
192 -> 105  % { -> ¦
079 -> 099  % ! -> ¦
Translation ItalyEBCDIC   % Italy EBCDIC
123 -> 067  % # -> ó
124 -> 072  % @ -> º
065 -> 081  % á -> ¬
192 -> 103  % { -> +
224 -> 236  % \ -> ÷
208 -> 104  % } -> +
066 -> 105  % í -> ¦
161 -> 089  % ~ -> ¦
079 -> 098  % ! -> ¦
121 -> 099  % ` -> ¦
Translation NorwayEBCDIC   % Norway EBCDIC
234 -> 079  % ( -> !
115 -> 161  % + -> ~
124 -> 183  % @ -> ¯
066 -> 252  % í -> n
065 -> 250  % á -> ·
224 -> 251  % \ -> v
095 -> 187  % ^ -> p
121 -> 100  % ` -> ¦
208 -> 235  % } -> )
192 -> 225  % { -> ƒ
079 -> 234  % ! -> (
161 -> 115  % ~ -> +
Translation SwissFrenchEBCDIC   % Swiss French EBCDIC
123 -> 067  % # -> ó
066 -> 072  % í -> º
161 -> 127  % ~ -> "
065 -> 081  % á -> ¬
124 -> 103  % @ -> +
224 -> 236  % \ -> ÷
208 -> 104  % } -> +
192 -> 105  % { -> ¦
079 -> 099  % ! -> ¦
Translation SwissGermanEBCDIC   % Swiss German EBCDIC
124 -> 072  % @ -> º
065 -> 183  % á -> ¯
224 -> 186  % \ -> G
066 -> 187  % í -> p
161 -> 071  % ~ -> ª
192 -> 100  % { -> ¦
079 -> 120  % ! -> -
208 -> 115  % } -> +
```

```
Translation SpainEBCDIC   % Spain EBCDIC
065 -> 137  % á -> i
123 -> 067  % # -> ó
124 -> 072  % @ -> º
192 -> 081  % { -> ¬
066 -> 111  % í -> ?
079 -> 220  % ! -> =
208 -> 236  % } -> ÷
224 -> 157  % \ -> +
Translation SwedenEBCDIC   % Sweden EBCDIC
091 -> 070  % $ -> Ñ
065 -> 183  % á -> ¯
066 -> 252  % í -> n
124 -> 204  % @ -> O
224 -> 186  % \ -> G
095 -> 187  % ^ -> p
192 -> 100  % { -> ¦
208 -> 235  % } -> )
121 -> 105  % ` ->
079 -> 120  % ! -> -
161 -> 115  % ~ -> +
Translation UnitedKingdomEBCDIC   % United Kingdom EBCDIC
123 -> 067  % # ->
Translation GermanEBCDIC   % German EBCDIC
124 -> 072  % @ ->
065 -> 183  % á -> ¯
224 -> 186  % \ -> G
066 -> 187  % í -> p
161 -> 071  % ~ -> ª
192 -> 100  % { -> ¦
079 -> 120  % ! ->
208 -> 115  % } -> +
%=============================================================================
%                       Transforms
%                       ==========
%
%       A TRANSFORM is a special formatting routine used for populating
%       VIRTUAL datasets or transforming an update in other ways.
%       Syntax:
%
%               TRANSFORM <transformname> IN "<patchfiletitle>"
%       A transform is a user-written procedure in a patch file that
%       is compiled into Support (similar to an INTERNAL FORMAT). A
%       transform has access to both the before- and after-images at
%       the same time the formatting routine.
%
%       The patch file should declare the transform procedure header
%       using the DBTransformHead define in DBInterface. Unlike a
%       format, the procedure return value is DBMTYPE rather than a
%       boolean. The parameters are UI, BI, AI, Format, and Writer,
%       which are respectively, the UpdateInfo, before-image,
%       after-image, formatting routine, and the Accessory-supplied
%       output routine.
%
%       These transforms are predefined in Support:
```

```
%               ChangedRecordsOnly
%                       Discards records if none of the data items
%                       allowed by the filter are modified.
%
%               ChangedItemsOnly
%                       Discards unmodified data items. The Accessory
%                       receives only the key items and data items that
%                       changed.
%
%       Example:
%
%*      TRANSFORM VIRTUALADDRESS
%*          IN "PATCH/DATABRIDGE/SAMPLE/SUPPORT/FORMATADDRESS"
%
%  Transforms go here ...
%=============================================================================
%                       Formats
%                       =======
%
% Option        Means                                                 Default
% --------      ---------------------------                           -------
% <type>        type of format                                        DISPLAY
% FORMAT        name of the formatting routine
% PREFIX        what to put at the beginning of each record           Nothing
% POSTFIX       what to put at the end of each record                 Nothing
% SEPARATOR     what to put between each data item                    Nothing
% DECIMALPT     decimal point character in numbers                    Nothing
% DATACHECK     generate code to check for nulls, invalid             TRUE
%               characters or illegal numbers and overflows
% PADDING       "fill" character for short records                    48"00"
% OVERFLOW      "fill" character for integer overflows                9999...
% NULL          "fill" character for NULL values       numbers: 0000...
%                                                         text: Spaces
% POSITIVE      format of a non-negative number                       VALUE
% NEGATIVE      format of a negative number                           VALUE
% UNSIGNED      format of an unsigned number                          VALUE
% TEXT          format of an alphanumeric item                        VALUE
% TRANSLATE     name of the translation table to use for text         None
% FLOAT         format of floating point REAL                 SCIENTIFIC (11)
%               Either DECIMAL (w, d) or SCIENTIFIC (w)
% TRUE          string value to use for true                          "1"
% FALSE         string value to use for false                         "0"
%
% <Type>        Format type
% -------       ----------------------------------
% DISPLAY       readable formatting
% BINARY        binary formatting
% RAW           unformatted
% VARYING       variable-length, e.g. no trailing spaces, readable
%               fields
%
% Code          Represents
% -----         ----------------------------------
% STRNUM        structure number (4 digits)
% STRNAME       structure name (17 characters)
```

```
% RECTYPE       record type (3 digits)
% CHANGECODE    change code ('A' add, 'D' delete, 'M' modify)
% MODFLAG       modifies flag ('1' delete/add was a modify, else '0')
% STACKNBR      stack number of updating program (4 digits)
% MODE          "tracking" mode, ('0' extract, '2' normal, etc.)
% FORMATLEVEL   dataset format update level (4 digits)
% AFN           audit file number (4 digits)
% ABSN          audit block serial number (10 digits)
% SEG           audit file segment number (7 digits)
% INX           audit block word index (5 digits)
% YYYY          year (4 digits)
% YY            year (2 digits)
% MM            month (2 digits)
% DD            day (2 digits)
% HH            hour (2 digits)
% MN            minute (2 digits)
% SS            second (2 digits)
%
% UID           unique ID of record as 12 decimal digits*
% HEXUID        unique ID of record as 12 hexadecimal digits*
% PUID          unique ID of parent record as 12 decimal digits*
% HEXPUID       unique ID of parent record as 12 hexadecimal digits*
% BIGUID        unique ID of record as 15 decimal digits*
% BIGPUID       unique ID of parent record as 15 decimal digits*
% RSN           (same as BIGUID)
%
%...    * Note: These UIDs may or may not be present depending on the
%               type of dataset. If they are present, they will auto-
%               matically have the 'separator' characters around them.
%
%       BIGUID or RSN is required for unique IDs that are RSNs to avoid
%       losing significant digits.
%
% CR            carriage return character
% LF            line feed character
% SPACE         space character
% TAB           horizontal tab character
%
% DATANAME      name of the data item
% VALUE         value of the data item in EBCDIC form.
%               Type            Size (in bytes) in output:
%               ----            -------------------------
%               ALPHA (n)       n
%               NUMBER (n)      n
%               NUMBER (Sn)     1 (sign) + n
%               NUMBER (n, m)   n + 1 (decimal point)
%               NUMBER (Sn, m)  1 (sign) + n + 1 (decimal point)
%               REAL            1 (sign) + w [from FLOAT SCIENTIFIC (w)
%                                             or FLOAT DECIMAL (w, d)]
%               REAL (n)        11
%               REAL (Sn)       1 (sign) + 11
%               REAL (Sn, m)    1 (sign) + 11 + 1 (decimal point)
%               RECORD TYPE     11
%               FIELD (n)       11
%               BOOLEAN         1    ('0' is false, '1' is true)
```

```
%
%=====================================================================
display format  KEYFORMAT
%               ---------
% formatting for key items. Required.
prefix         ==> SPACE
separator      SPACE
decimalpt      .
unsigned       DATANAME = VALUE
positive       DATANAME = +VALUE
negative       DATANAME = -VALUE
text           DATANAME = "VALUE"
boolean        DATANAME = VALUE
padding        SPACE
float          DECIMAL (12, 4)
display format  COMMAFORMAT
%               -----------
% prefix; comma between each data item
prefix         STRNUM , RECTYPE , CHANGECODE ,
postfix        UID PUID LF
separator      ,
decimalpt      .
positive       +VALUE
negative       -VALUE
text           "VALUE"
padding        SPACE
float          SCIENTIFIC (11)         % #.#####E-##
%float          DECIMAL (12, 2)         % #########.##
%translate      UnitedKingdom
%null           SPACE
%overflow       *
%true           "TRUE "
%false          "FALSE"
display format  RSNCOMMA
%               --------
% prefix; comma between each data item
prefix         STRNUM , RECTYPE , CHANGECODE ,
postfix        BIGUID BIGPUID LF
separator      ,
decimalpt      .
positive       +VALUE
negative       -VALUE
text           "VALUE"
padding        SPACE
float          SCIENTIFIC (11)         % #.#####E-##
display format  SNAPSHOTCOMMA
%               -------------
% COMMAFORMAT but no prefix; just the data
postfix        UID PUID LF
separator      ,
decimalpt      .
positive       +VALUE
negative       -VALUE
text           "VALUE"
padding        SPACE
```

```
%translate      UnitedKingdom
%float          DECIMAL (12, 2)
true           "1"
false          "0"
%* display format  COMMAFORMATQUOTEALL
%                  -------------------
% COMMAFORMAT with quotes around everything
%* prefix          "STRNUM" , "RECTYPE" , "CHANGECODE" ,
%* postfix         UID PUID LF
%* separator       ,
%* decimalpt       .
%* positive        "+VALUE"
%* negative        "-VALUE"
%* unsigned        "VALUE"
%* text            "VALUE"
%* boolean         "VALUE"
%* true            "T"
%* false           "F"
%* padding         SPACE
%* float           DECIMAL (12, 2)
display format  FIXEDFORMAT
%               -----------
% no delimiters; prefix has structure number, record type, change code
prefix         STRNUM RECTYPE CHANGECODE
postfix        UID PUID
decimalpt      .
positive       +VALUE
negative       -VALUE
padding        SPACE
%translate      UnitedKingdom
%float          DECIMAL (12, 2)
datacheck      TRUE
display format  SNAPSHOTFIXED
%                -------------
% FIXEDFORMAT but no prefix; just the data
postfix        UID PUID
decimalpt      .
positive       +VALUE
negative       -VALUE
padding        SPACE
%translate      UnitedKingdom
%float          DECIMAL (12, 2)
display format  NameFormat
%               ----------
% Insert the dataname in front of each data item value.
prefix         CHANGECODE SPACE STRNAME SPACE
separator      ,
unsigned       DATANAME = VALUE
positive       DATANAME = +VALUE
negative       DATANAME = -VALUE
text           DATANAME = "VALUE"
boolean        DATANAME = VALUE
padding        SPACE
binary format BINARYFORMAT  % required for DBServer
%             ------------
```

```
%* translate    UnitedKingdom    % translation for ALPHAs
format  AUDITLOC
%       --------
prefix          MM/DD/YYYY@HH:MN:SS[AFN,ABSN,INX] CHANGECODE
separator       ,
%
% A VARYING format has fields that vary in size depending on their
% contents. Alpha fields truncate trailing spaces. Numeric fields strip
% leading zeroes. The output file should be some type of STREAM file to
% allow for output records of different sizes.
%* varying format CSV
%               ---
%* prefix         STRNUM , RECTYPE , CHANGECODE ,
%* postfix        LF
%* separator      ,
%* decimalpt      .
%* positive       VALUE
%* negative       -VALUE
%* text           "VALUE"
%
%                      External formats
%                      ================
%
%       EXTERNAL FORMAT <formatname> IN "<librarytitle>"
%
% The <librarytitle> is the title of an ALGOL library codefile
% containing an EXPORTed formatting procedure called <formatname>.
% The user must write the library program. Example:
%
%       external format CUSTOMFORMAT
%               in "OBJECT/LOCAL/FORMATLIB"
%
%
%                      Internal formats
%                      ================
%
%       INTERNAL FORMAT <formatname> IN "<patchfiletitle>"
%
% The <patchfiletitle> is the title of an ALGOL source file
% containing a patch for Support that declares a formatting
% routine whose name is <formatname>.
%
%
%
%=============================================================================
%                      Filters
%                      =======
%
% Option        Means                                          Default
% --------      ----------------------------                   -------
% FILTER        name of filtering routine                      Nothing
% SELECT        specifies dataset name                         Nothing
% DEFAULT       default selection expression                   NONE
% $ INCLUDE     include ALGOL source file                      Nothing
%
```

```
% Syntax for "SELECT" option:
%
% SELECT <filteritems> FROM <dataset> <optional where> <optional using>
%
%       where <filteritems> is:
%               ALL
%
%.              *       (same as ALL)
%
%               <dataitem 1> <dataitem 2> ...
%
%               ALL EXCEPT <dataitem 1> <dataitem 2> ...
%
%               <vf 1>: <dataitem 1> <dataitem 2> ...
%               <vf 2>: <dataitem 3> <dataitem 4> ...
%               ELSE: * (or NONE or ALL)
%               ...     (for variable format datasets)
%
%       Dataitems can be GROUPs or elementary items. If the data item
%       OCCURS, all occurrences will be selected.
%
%       The <optional where> part has the following syntax:
%               WHERE <booleanexpression>
%               (if omitted, all records will be selected from the
%               dataset)
%
%       The <booleanexpression> can specify the type of update.
%       Syntax:
%
%               UPDATETYPE (<type>)
%
%       where <type> is CREATE, MODIFY, or DELETE.
%
%       Example:
%               select * from ORDERS
%                       where UpdateType (CREATE)
%                          or UpdateType (MODIFY)
%
%       The <optional using> part has the following syntax:
%               USING <(sub)set>
%
%       The <(sub)set> specifies the name of a set or subset that will
%       be used to extract records during a clone. If the subset has a
%       WHERE clause in the DASDL, GenFormat will automatically append
%       that expression to the WHERE clause in the SELECT statement.
%
%       Example:
%               select * from ACCOUNTS
%                       where ACCT-TYPE = 1
%                       using ACTIVEACCTS       % subset of ACCOUNTS
%
%
% Syntax for "DEFAULT" option:
%
% DEFAULT ALL   (select all records for unSELECTed datasets)
```

```
% DEFAULT ANY   (same as ALL)
% DEFAULT NONE  (do not select any records for unSELECTed datasets)
% DEFAULT ALL EXCEPT datasetlist   (exclude datasets in list)
% DEFAULT UPDATETYPE (<type>) [OR UPDATETYPE (<type>)]
%
% Relational
% Operator      Means
% ---------     ----------------------------------
%  =            Equal to
%  ^=           Not equal to
%  >            Greater than
%  >=           Greater than or equal to
%  <            Less than
%  <=           Less than or equal to
%  EQL          Equal to
%  NEQ          Not Equal to
%  GTR          Greater than
%  GEQ          Greater than or Equal
%  LSS          Less than
%  LEQ          Less than or Equal
%  IS           Bit-for-bit equal
%  ISNT         Not bit-for-bit equal
%
% "AND" and "OR" may be used to connect multiple expressions.
% "NOT" may be used to reverse the value of an expression.
filter  DISCARDALL    % discard every record
%         ----------
default NONE
% ------------------------------------------------------------------------
filter  EVERYTHING    % keep every record
%         ----------
default ALL
% ----- Examples ------
%* filter  ONLYBANK1
%       ---------
%* default NONE    % return records only for the datasets specified in
% the SELECT statements
%* select  * from BANK     where BANK-ID = 1;
%* select  * from BRANCH   where BANK-ID = 1;
%* select  * from CUSTOMER where BANK-ID = 1;
% ------------------------------------------------------------------------
%
%* filter  EXAMPLEFILTER
%       -------------
%* default ALL     % return all records for datasets that do not have
%*                 % a SELECT statement
%* select  * from BRANCH where BRANCH-ID = 2
%*                          and BRANCH-NAM = "SOUTH MANHATTAN";
%* select  * from TELLER where BRANCH-ID = 2 and TL-CSH-OUT > 99.99;
%* select  * from ACCOUNT where FALSE; % no records at all
%* select  * from L1 where DS-NAME  > "a" using L1-SS2;
% ------------------------------------------------------------------------
%
%* filter  CERTAINFIELDS
%         -------------
```

```
%* default NONE  % don't return any records for datasets that do
%*               % not have a SELECT statement
%* select  BRANCH-ID BRANCH-AD1
%*               from BRANCH
%*               where BRANCH-ID > 1
%*                 and BRANCH-NAM = "SOUTH MANHATTAN"
%*                 and BRANCH-AD1 NEQ " ";
%* select  FS-NAME from FUNNY-STUFF where FS-2;  % FS-2 is a BOOLEAN
% ACCOUNT is a variable-format dataset
%* select 1: AC-TYPE AC-NUMBER CUST-ID AC-BALANCE
%*           BRANCH-ID AC-HLD-AMT
%*        3: ac-number AC-TYPE   % lowercase data item OK
%*        2: ALL
%*        4: ALL EXCEPT AC-BALANCE
%*        0: AC-NUMBER AC-DT-CLSE
%*        else: none
%*     from ACCOUNT where BANK-ID = 1;
%* select MONTH, BANK-ID from TRIALBALANCES;
% ------------------------------------------------------------------------
%
%* filter WAREHOUSE
%      ---------
%     Discard any deletes, but keep creates and modifies
%* select * from HISTORY
%*          where PROC-DATE >= 000101
%*            and not UpdateType (Delete);
%* default all except TRIALBALANCES, FUNNY-STUFF, SHORT-VF
% ------------------------------------------------------------------------
%
%       Put additional filters here ...
%
% ------------------------------------------------------------------------
%
%               External filters
%               ================
%
%       EXTERNAL FILTER <filtername> IN "<librarytitle>"
%
% The <librarytitle> is the title of an ALGOL library codefile
% containing an EXPORTed filtering procedure called <filtername>.
% The user must write the library program. Example:
%
%% external filter CUSTOMFILTER in "OBJECT/LOCAL/FILTERLIB"
%
% ------------------------------------------------------------------------
%
%               Internal filters
%               ================
%
%       INTERNAL FILTER <filtername> IN "<patchfiletitle>"
%
% The <patchfiletitle> is the title of an ALGOL source file
% containing a patch for Support that declares a filtering
% routine whose name is <filtername>. Example:
%
```

```
%% internal filter CustomSelection
%%         in "PATCH/DATABRIDGE/SUPPORT/LOCALFILTERING"
%
% ------------------------------------------------------------------------
%
%               Error Manager
%               =============
%
%       ERROR MANAGER <errormanagername> IN "<patchfiletitle>"
%
% The <patchfiletitle> is the title of an ALGOL source file containing
% an error handling entry point called <errormanagername> that will
% replace the DBErrorManager entry point in Support. This entry point
% can analyze an error and log it, display it, etc., and its return
% value indicates whether the Accessory should continue processing or
% terminate.
%
% The entry point can be declared with the DBErrorManagerHead define
% declared in DBInterface. The patch file can contain declarations
% global to the error manager procedure that will persist until
% Support terminates. Example:
%
%*      Error Manager ErrorHandler
%*              in "PATCH/DATABRIDGE/SAMPLE/SUPPORT/ERRORHANDLER"
%
% ------------------------------------------------------------------------
%
%               Startup
%               =======
%
%       STARTUP IN "<patchfiletitle>"
%
% The <patchfiletitle> is the title of an ALGOL source file
% containing a patch for Support that contains user-written code that
% is to be executed when the Support library is initializing *before*
% it freezes. The contents of the patchfile is inserted into a
% procedure called CustomStartup. It may contain declarations as well
% as statements. Any variables declared in the patchfile will be
% discarded when CustomStartup exits unless they are declared OWN.
% Example:
%
%*      Startup in "PATCH/DATABRIDGE/SAMPLE/SUPPORT/STARTUP"
%
% ------------------------------------------------------------------------
%
```

```
%               Shutdown
%               ========
%
%       SHUTDOWN IN "<patchfiletitle>"
%
% The <patchfiletitle> is the title of an ALGOL source file
% containing a patch for Support that contains user-written code that
% is to be executed after the Support library thaws. The contents of
% the patchfile is inserted into a procedure called CustomShutdown. It
% may contain declarations as well as statements. Example:
%
%*      Shutdown in "PATCH/DATABRIDGE/SAMPLE/SUPPORT/SHUTDOWN"
%
% ================== End of GenFormat input ==========================
```

Creating a Format

The GenFormat parameter file can contain any number of formats. To create a format, type it in the GenFormat parameter using the syntax for declaring formats. See "FORMAT Syntax" on page 69.

Declaring Internal and External Formats

Instead of using GenFormat to define your formats, you can write custom internal and external formats that you specify in the GenFormat parameter file. For more information on user-written formats, refer to the *Databridge Programmer's Reference.*

Format types

There are two main types of formats: binary and display.

Binary formats, e.g. BINARYFORMAT and RAWFORMAT, don't convert the data types of the data items in the DMSII database in the output record. For example, NUMBER items consist of 4-bit digits in the database and retain exactly the same 4-bit values in the formatted output record. These formats are declared with either the keyword BINARY or RAW preceding the keyword FORMAT.

Display formats, e.g. COMMAFORMAT and FIXEDFORMAT, convert the database data items into EBCDIC characters such that the output record is easily readable by any text editor. These formats are declared with either the keyword DISPLAY or VARYING preceding the keyword FORMAT. DISPLAY is the default format type.

Binary Value Conversions

All DISPLAY formats generated by GenFormat routines convert binary values to 11-digit items in the output. For example, the data item `FIELD (nn)` is a binary value where *nn* is the number of bits. Bits do not map cleanly to digits, however, so the binary value the bits represent are converted to 11-digit items.

This binary value conversion occurs for all GenFormat DISPLAY formats. The default format (RAWFORMAT), BINARYFORMAT, and any formats you write will not have the binary format conversion.

DMSII uses `FIELD` in two ways. As discussed above, `FIELD (nn)` is an integer of *nn* bits. The other way DMSII uses `FIELD` is as a GROUP of one-bit BOOLEANs, as in the following DASDL example:

```
APPLE-FLAGS FIELD
(RED-GREEN;
SWEET-TART;
SOFT-HARD);
```

Although a FIELD item with no specified number of bits can be treated as an integer, it functions as a group for the BOOLEANs RED-GREEN, SWEET-TART, and SOFT-HARD. In this case, the GenFormat format routines ignore APPLE-FLAGS and return the value of the BOOLEANs within it.

Boolean Values

You can specify the representation of Boolean values in each DISPLAY format. The default is to represent TRUE as 1 and FALSE as 0, both as unsigned values. To change this representation:

BOOLEAN *leadingcharacters* VALUE *trailingcharacters*
TRUE "*string*"
FALSE "*string*"

Where	Is
leadingcharacters	Optional. Any character, such as single or double quotes, that you want to place in front of the value returned from a Boolean expression.
VALUE	Required. A literal that indicates the value returned when a Boolean expression is evaluated.
trailingcharacters	Optional. Any character, such as single or double quotes, that you want to place after the value returned from a Boolean expression.
TRUE or FALSE	Required. The Boolean values TRUE or FALSE.
"*string*"	Required. Any alphanumeric character or characters you want to use to represent the Boolean values TRUE or FALSE. You must enclose the string in quotation marks.

Example 1

In the following example, the Y represents TRUE and the N represents FALSE. If the Boolean evaluates to TRUE, Y is returned.

```
BOOLEAN VALUE
TRUE "Y"
FALSE "N"
```

Example 2

In the following example, lowercase true represents TRUE, and lowercase false represents FALSE. If this Boolean evaluates to true, "true " (in double quotation marks) is returned. The double quotation marks are returned because this example uses the optional leading and trailing characters for Boolean values.

NOTE: Insert one space after the word true so that the literals "true " and "false" have the same length (five characters). This assures that the formatted records have the same length. Otherwise, GenFormat displays a warning.

```
BOOLEAN "VALUE"
TRUE "true "
FALSE "false"
```

Default Formats

This section summarizes the formats available in the sample GenFormat parameter file.

Format	Description
AUDITLOC	AUDITLOC inserts a prefix consisting of the time-of-day, the audit location (AFN, ABSN, Inx), and a change code in front of the record values.
BINARYFORMAT	BINARYFORMAT copies the binary image of each data item in the record, except for the following: ◆ Data items that are not in the filter ◆ Text items when the TRANSLATE option is specified
COMMAFORMAT	COMMAFORMAT includes prefix, data, and postfix and is similar to FIXEDFORMAT except that it puts double quotes around alpha items and a comma between all items. This format is suitable for many PC and UNIX import programs.
COMMAFORMATQUOTEALL	COMMAFORMATQUOTEALL includes prefix, data, and postfix. Like COMMAFORMAT, each item is delimited by commas. Unlike COMMAFORMAT, all items (alpha or not) are placed in double quotes. **NOTE:** COMMAFORMATQUOTEALL is commented out by default. If you want to use it, you must uncomment it.
FIXEDFORMAT	FIXEDFORMAT includes prefix, data, and postfix. FIXEDFORMAT converts each data item to an EBCDIC representation of a fixed width. This would be similar to a COBOL program moving each field to a PIC ... DISPLAY.
KEYFORMAT	KEYFORMAT lists the data items (by name and value) for a primary key.
NAMEFORMAT	NAMEFORMAT inserts the name of the data item and an equal sign in front of the value of each data item.
RAWFORMAT	RAWFORMAT copies the DMSII record as a binary image without any formatting or data item filtering. It provides a datastream that is exactly the same as the data stored on the host. RAWFORMAT is an alias for the default DBFORMAT, which formats records as a binary image of the record as it would appear to a COBOL program.

Format	Description
RSNCOMMA	RSNCOMMA is similar to COMMAFORMAT but uses BIGUID and BIGPUID, the 15-digit versions of UID and PUID, which can accommodate RSN values. The sample GenFormat parameter file defines additional formats that are commented out with "%*" at the beginning of each line. You can quickly uncomment those formats by replacing the "%*" character sequence with spaces.
SNAPSHOTCOMMA	SNAPSHOTCOMMA includes data only, no prefix or postfix. Like COMMAFORMAT, each item is delimited by commas and alpha items have double quotes.
SNAPSHOTFIXED	SNAPSHOTFIXED is the same as FIXEDFORMAT except that only data is included. There is no prefix or postfix.

FORMAT Syntax

You can use generated formats or user-written formats. Literals for defining these formats are in the section that follows.

Generated FORMAT

To declare a generated format, use the following syntax:

```
[type] FORMAT formatname
formatoption1 .
.
.
formatoptionn
```

User-written FORMAT

To declare a user-written format, use the following syntax:

```
INTERNAL FORMAT formatname IN "patchfiletitle"
```

—or—

```
EXTERNAL FORMAT formatname IN "librarytitle"
```

Where	Is
type	Optional. type represents DISPLAY, BINARY, RAW, VARYING, EXTERNAL, or INTERNAL as explained in "Type" in the following table. If you enter a type, do not include the brackets.
formatname	Required. *formatname* is the name of the format you will enter in Databridge Accessory parameter files when you want to use this particular format. The name must start with a letter followed by letters, numbers, or underscores. The only limit on the length of the format name is the length of a record in the parameter file, which for SEQDATA files is 72 characters.
formatoption	Optional. *formatoption* is any of several methods of formatting the data. See "FORMAT Options" on page 70

Literals for Defining Formats

The following table lists the literals used with formats that can be defined.

Type	Description
`DISPLAY FORMAT formatname`	`DISPLAY` indicates that the fields are formatted in readable format. All numbers, Booleans, etc., are converted to EBCDIC text. If you don't include a type, GenFormat uses `DISPLAY` as the default. Valid format options: All (for example, `POSITIVE`, `UNSIGNED`, `TEXT`, etc.)
`VARYING FORMAT formatname`	While similar to `DISPLAY`, `VARYING` discards trailing spaces in alpha items and discards leading zeros in numeric items. `NULL` items are represented as empty strings. This format type is intended for writing comma-separated values to a stream file. Valid format options: Same options as `DISPLAY` except `NULL` and `OVERFLOW` options are ignored.
`BINARY FORMAT formatname`	`BINARY` indicates the fields are bit images from the original record. Valid format options: `TRANSLATE` for specifying a translation table for text items
`RAW FORMAT formatname`	`RAW` indicates the original record is output without editing or data item filtering. Valid format options: None
`INTERNAL FORMAT formatname IN "patchfiletitle"`	`INTERNAL` indicates that the (user-written) formatting routine is included as a patch file when the Support Library is compiled. Valid format options: None
`EXTERNAL FORMAT formatname IN "librarytitle"`	`EXTERNAL` indicates that the (user-written) formatting routine is an entry point in a library program. Valid format options: None

FORMAT Options

The following FORMAT items are optional.

Option	Default	Description
`PREFIX`	Nothing	Items inserted before the record. The possible values are listed in "PREFIX and POSTFIX Codes" on page 72.
`POSTFIX`	Nothing	Items inserted after the record. The possible values are listed in "PREFIX and POSTFIX Codes" on page 72.

Option	Default	Description
DATACHECK	TRUE	Determines whether the format should contain code for validating data, such as checking for nulls, illegal characters, and integer overflows. Syntax: `DATACHECK [ TRUE | FALSE ]` By setting DATACHECK to FALSE, the formatting routines will be smaller and execute faster (by about 5%). For example: `format NoCheck` `prefix STRNAME RECTYPE CHANGECODE` `postfix UID PUID` `DataCheck false` The Span and Snapshot Accessories can benefit from setting DATACHECK to FALSE especially if they use a DISPLAY FORMAT such as FIXEDFORMAT. (DBServer/Databridge Client will see very little improvement.) **NOTE:** The DATACHECK option is not valid for RAW formats.
SEPARATOR	Nothing	Identifies the separator to use between each item (column). The value specified here will separate each of the DATAITEMS selected from each DATASET. This value does not apply to PREFIX or POSTFIX.
DECIMALPT	Nothing	Identifies the decimal point character.
PADDING	NULL	The character used to fill short records.
POSITIVE	VALUE	The format of a non-negative number. For more information, see "DATANAME and VALUE" on page 74. If you want the name of the data item in the output records, enter the keyword DATANAME, as follows: `POSITIVE` *DATANAME* `= +`*VALUE*
NEGATIVE	VALUE	The format of a negative number. See "DATANAME and VALUE" on page 74. If you want the name of the data item in the output, enter the keyword DATANAME, as follows: `NEGATIVE DATANAME = -`*VALUE*
UNSIGNED	VALUE	The format of an unsigned number. See "DATANAME and VALUE" on page 74. If you want the name of the data item in the output, enter the keyword DATANAME, as follows: `UNSIGNED DATANAME =` *VALUE*
TEXT	VALUE	The format of an alphanumeric item. See "DATANAME and VALUE" on page 74. If you want the name of the data item in the output, enter the keyword DATANAME, as follows: `TEXT DATANAME = "`*VALUE*`"`

Option	Default	Description
TRANSLATE	None	The name of a translation table defined in the Translations portion of the GenFormat parameter file. Translation tables are explained in "Creating and Using Translation Tables" on page 85.
FLOAT	SCIENTIFIC (11)	The format of a real number. Can be specified as DECIMAL (w, d) or SCIENTIFIC (w). DECIMAL has a fixed number of decimal places, indicated by d. For example, DECIMAL (12, 2) means 12 total characters, two of which are to the right of the decimal point. **Example:** #########.## SCIENTIFIC uses exponential notation to represent a floating point number in a fixed width, indicated by w. For example, SCIENTIFIC (11) means 11 total characters in the following format: #.#####E-##
OVERFLOW	9	When you include the OVERFLOW option, an integer overflow replaces the field with the OVERFLOW character for the width of the field. If you do not specify an OVERFLOW character, the field will contain all nines and will be formatted normally.
NULL	Numbers: 0 Text: spaces	If specified and the field contains the DMSII defined NULL character(s), then the formatting routine will replace the field with the FORMAT-defined NULL character for the width of the field.

PREFIX and POSTFIX Codes

This topic includes a table of valid codes for the PREFIX and POSTFIX format options. You can use all of these codes in any combination.

Code	Description
STRNUM	Structure number of the data set, 4 digits
STRNAME	The 17-character data set name
RECTYPE	Type of record for variable-format data sets, 3 digits
CHANGECODE	A = add D = delete M = modify H = audit file header record L = link after-image N = link before-image S = state update X = documentation record

Code	Description
MODFLAG	1= if ADD was originally the after-image or DELETE was originally the before-image of a MODIFY 0 = otherwise
STACKNBR	Stack number of updating program, 4 digits
MODE	The state of the data set related to cloning, as follows: 0 = The data set is in the extract phase. 1 = The data set is in the fixup phase. 2 = The data set is in the update phase. 3 = The data set was reorganized. 4 = The data set was purged.
FORMATLEVEL	Data set format level, 4 digits
AFN	Audit file number, 4 digits
ABSN	Audit block serial number, 10 digits
SEG	Audit file segment number, 7 digits
INX	Audit block word index number, 5 digits
YYYY YY MM DD HH MN SS	Year, 4 digits Year, 2 digits Month, 2 digits Day, 2 digits Hour, 2 digits Minute, 2 digits Second, 2 digits Note: These time values represent an estimate of when an update occurred. The actual time of the update could be later than these values indicate.
UID	Unique ID (absolute address) of the record. The UID may not be present, depending on the type of data set. If the UID is present, it is a 12-digit number without leading or trailing characters.
BIGUID	For data sets having an RSN, the format will use the RSN as the BIGUID Unique ID (absolute address) of the record. The BIGUID may not be present, depending on the type of data set. If the BIGUID is present, it is a 15-digit number without leading or trailing characters. RSN is a synonym for BIGUID.
HEXUID	Unique ID (absolute address) of the record. The HEXUID may not be present, depending on the type of data set. If the HEXUID is present, it is a 12-digit hexadecimal number without leading or trailing characters.
PUID	Parent unique ID (parent absolute address) of the parent record. The PUID may not be present, depending on the type of data set. For example, a PUID is typically present for embedded data sets. If the PUID is present, it is a 12-digit number without leading or trailing characters.
BIGPUID	Parent unique ID (parent absolute address) of the parent record. The BIGPUID may not be present, depending on the type of data set. For example, a BIGPUID is typically present for embedded data sets. If the PUID is present, it is a 15-digit number without leading or trailing characters.

Code	Description
HEXPUID	Parent unique ID (parent absolute address) of the parent record. The HEXPUID may not be present, depending on the type of data set. For example, a HEXPUID is typically present for embedded data sets. If the HEXPUID is present, it is a 12-digit hexadecimal number without leading or trailing characters.
CR	Carriage return
LF	Line feed
SPACE	Space character
TAB	Horizontal tab character

DATANAME and VALUE

DATANAME DATANAME is the name of the data item and the associated subscript used in conjunction with VALUE so that output has the format of `data item=value`. For example, for a data item named EMPLOYEE-ID (5), the output would be EMPLOYEE-ID.

VALUE VALUE represents the EBCDIC representation of a data item. The following chart shows the number of bytes Databridge uses to format various types of data items.

Type	Size
ALPHA(*n*)	*n* characters, depending on the actual size of the data item
NUMBER(*n*)	*n* digits, depending on the actual size of the data item
NUMBER(S*n*)	1 character (sign) + *n* digits
NUMBER(*n*, *m*)	1 character (decimal point) + *n* digits, of which *m* are to the right of the assumed decimal point
NUMBER(S*n*, *m*)	1 character (sign) + 1 character (decimal point) + *n* digits, of which *m* are to the right of the assumed decimal point
REAL	1 character (sign) + *w* digits where *w* is from FLOAT SCIENTIFIC (*w*) or from FLOAT DECIMAL (*w*, *d*)
REAL(*n*)	11 digits
REAL(S*n*)	1 character (sign) + 11 digits
REAL(S*n*,*m*)	1 character (sign) + 1 character (decimal point) + 11 digits, of which *m* are to the right of the assumed decimal point
RECORDTYPE	11 digits

FIELD(*n*)	11 digits
BOOLEAN	1 character, as follows (by default): 0 = false 1 = true

FORMAT Example

The following example shows a FORMAT declaration:

```
format  COMMAFORMAT
%       -----------
% prefix; comma between each data item
This example uses the following formatting guidelines
PREFIX          STRNUM , RECTYPE , CHANGECODE ,
POSTFIX         UID PUID LF
SEPARATOR       ,
DECIMALPT       .
POSITIVE        +VALUE
NEGATIVE        -VALUE
TEXT            "VALUE"
PADDING         SPACE
FLOAT           SCIENTIFIC (11)         % #.#####E-##
```

The preceding example uses the following formatting principles:

- As designated by the entry for PREFIX, each record will start with the structure number, record type, and change code. Each field of the prefix will also be separated by commas.
- As designated by the entry for POSTFIX, each record will end with the absolute address (unique ID or UID), the parent unique ID (if present), and a line feed (LF).
- All of the data within the record will be separated by commas as designated by the entry for SEPARATOR.
- Positive numbers within the data will be formatted, for example, as +000120 (designated by the entry for POSITIVE).
- Negative numbers within the data will be formatted, for example, as -000120 (designated by the entry for NEGATIVE).
- The decimal point character is the period (.), as designated by the entry for DECIMALPT.
- All text within the record will be enclosed in quotation marks, as designated by the entry for TEXT.
- Any unused area at the end of a record will be filled with spaces, as designated by the entry for PADDING.
- All real numbers within the record will use the scientific format, as designated by the entry for FLOAT.

Creating a Filter

By default, GenFormat allows all supported DMSII data sets and records to be replicated. You can create filters to limit the records that can be replicated. To create a filter, you must compile a tailored support library.

The GenFormat parameter file can contain any number of filters. The only predefined filter in the GenFormat parameter file is DISCARDALL, which discards every record. To create a filter using GenFormat, type the filter into the GenFormat parameter file using the syntax for declaring filters. See "FILTER Syntax" on page 77.

Declaring User-Written Filters

Instead of using GenFormat to define your filters, you can also write your own ALGOL filter (refer to the *Databridge Programmer's Reference* for information about user-written filters), you must specify it when you declare the filter in the GenFormat parameter file.

There are three ways to declare user-written filters:

- INTERNAL FILTER
- EXTERNAL FILTER
- $ INCLUDE

When you declare an INTERNAL, EXTERNAL, or $ INCLUDE filter, you cannot use any other filter options with that filter.

INTERNAL FILTER

Use the INTERNAL FILTER statement to include your user-written filter as a patch file when the Support Library is compiled. In an INTERNAL FILTER, you can declare a procedure name, a procedure heading, and global data. Using INTERNAL FILTER rather than the $ INCLUDE type of filter is preferred because it allows you to declare data global to the filter procedure and the procedure heading is in the patch file.

Use the following syntax to declare an INTERNAL FILTER:

```
INTERNAL FILTER filtername IN "patchfiletitle"
```

where *patchfiletitle* is the title of an ALGOL source file containing a patch for the Support Library that declares a filter whose name is *filtername*. Note that if you have Accessories running under a different usercode from where the Databridge software is installed and you want to use the automatic recompilation of the Support Library feature, you must include the usercode and pack name where the patch file resides in *patchfiletitle*.

EXTERNAL FILTER

Use the EXTERNAL FILTER statement to indicate that your user-written filter is an entry point in a library.

Use the following syntax to declare an EXTERNAL FILTER:

```
EXTERNAL FILTER filtername IN "librarytitle"
```

where *filtername* is the name you have given to the filter and *librarytitle* is the file title of your compiled ALGOL library code.

$ INCLUDE

Use the $ INCLUDE statement to include your user-written filter as a patch file when the Support Library is compiled. The included file must contain ALGOL for the body of the filter. GenFormat will automatically declare the procedure heading and the outer BEGIN and END.

Use the following syntax to declare an $ INCLUDE:

```
FILTER filtername
$ INCLUDE "patchfiletitle"
```

where *patchfiletitle* is the title of an ALGOL source file containing a patch for the Support Library that defines a filter whose name is *filtername*. If you have Accessories running under a different usercode from where the Databridge software is installed and you want to use the automatic recompilation of the Support Library feature, you must include the usercode and pack name where the patch file resides in *patchfiletitle*.

FILTER Syntax

Filters use the following syntax:

```
FILTER filtername
DEFAULT defaultoption
SELECT statements
```

Where	Is
filtername	The name of the filter you will enter in Accessory parameter files when you want to use this particular filter. The name must start with a letter followed by letters, numbers, or underscores. There is no limit on the length of the filter name; however, it must fit on a single line.

Where	Is
defaultoption	An option that only affects data sets that are not included in a SELECT statement. Set *defaultoption* to one of the following: ◆ ALL, which indicates that Databridge should return all records for data sets that are not included in SELECT statements. ALL EXCEPT *dataset1*, *dataset2*, ..., which indicates that all datasets not having a SELECT statement are included in the filter except for *dataset1*, *dataset2*, etc. **Example:** `filter PUBLIC` `select * from ORDERS where ORD-TOTAL < 10000` `default all except SALARY, REVIEWS` **NOTE:** In the preceding example, SALARY and REVIEWS are not visible to the Accessory using the PUBLIC filter. ◆ ANY, which is a synonym for ALL. ◆ NONE, which indicates that Databridge should return no records except those you specify in SELECT statements. UPDATETYPE (*updatetype*) [OR UPDATETYPE (*updatetype*)] where *updatetype* is CREATE, MODIFY, or DELETE. **Example:** `DEFAULT UpdateType (Create) or UpdateType (Modify)`
statements	The series of statements that actually filter the data items and records for each specified data set. See "SELECT Statement Syntax" on page 78.

SELECT Statement Syntax

Enter the SELECT statements to specify the data sets and the records you want to be filtered. Use the following format:

SELECT *filteritems* FROM *datasetname* [WHERE *expression*] [USING *set_or_subset*]

Where	Is
SELECT	A required literal.

Where	Is
filteritems	One of the following: ◆ * or the literal ALL to select all the data items in the data set. Note that if the SELECT statement specifies * or ALL for the item list of a variable format data set and no rectypenumber lists, then each variable format will return all items belonging to that format part. If the SELECT statement specifies a data item list without a rectypenumber, then that list applies to every variable format part. ◆ One or more data items from a fixed-format data set, as follows: `dataitem[,] dataitem[,] dataitem` ◆ One or more data items from a variable-format data set, as follows: *`rectypenumber`*`: dataitem[,] dataitem[,] dataitem[,]` *`rectypenumber`*`: *` *`rectypenumber`*`: ALL` `[ ELSE: `*`elseoption`*` ]` **NOTE:** The *elseoption* can be NONE, * , or ALL. If the *elseoption* is NONE, unlisted record types will not be in the dataset enumeration and no data records of those types will be sent to the Accessory. If *elseoption* is * or ALL, unlisted record types will be in the dataset enumeration, and records that satisfy the WHERE clause will be sent to the Accessory. If the ELSE clause is not specified, it defaults to the filter's DEFAULT option value.
FROM	A literal.
datasetname	The name of the data set or remap from which you want to filter records.
WHERE	A literal.

Where	Is
expression	One of the following: ◆ A data item from the data set followed by a relational operator and a value, as in the following examples: `WHERE dataitem = n;` `WHERE BANK-ID = 1;` `WHERE BRANCH-ID = 2 AND BRANCH-NAME = "SOUTH MANHATTAN";` ◆ A BOOLEAN data item, as in the following examples: `WHERE EMP-SALARIED;` `WHERE EMP-GENDER OR EMP-ACTIVE;` ◆ A type of update—CREATE, MODIFY or DELETE. Specifying an update type is useful for data warehousing, where sites don't want to delete records. For example: `WHERE UPDATETYPE (Modify);` `WHERE DI-ORD-PLANT = 23 and not UpdateType (Delete);` ◆ An ALTERed data item. Specify the original attributes for the ALTERed data item. For example, if a data item was ALTERed from a NUMBER to an ALPHA, use the original numeric value in the WHERE clause, as follows: `WHERE dataitem = numericvalue` ◆ The DIV (integer division) or MOD (remainder) operators. This capability can be used to select representative sample records for testing purposes. For example: `WHERE CUST-ID MOD 5 = 1` **NOTE:** *expression* can also use the literal CONTAINS, as in the following syntax: `dataitem CONTAINS "string"` `WHERE alphadataitem CONTAINS "casesensitivestring"` The CONTAINS expression is true if the alpha data item contains the case-sensitive string enclosed in double quotation marks. You must include the quotation marks, as in the following example: `WHERE ADDR-LINE1 CONTAINS "BOX" OR ADDR-LINE1 CONTAINS "RURAL ROUTE"`
USING	A literal.

Where	Is
set_or_subset	The name of an existing set or subset that points to the data set. DBEngine uses this during an extract to locate records in the data set, which means records will be extracted in the order specified by the key. You can decrease the time it takes to do the extract by specifying a subset that has relatively few records. If *set_or_subset* is the name of an automatic subset, the WHERE clause defined in the DASDL for the subset will be appended to expression. Therefore, the Accessory will receive only the records that satisfy both the subset and the WHERE clauses. If *set_or_subset* is the name of a set or manual subset, there are no changes to expression. If *set_or_subset* is the name of a manual subset, you must ensure that the manual subset actually contains all of the records satisfying the WHERE clause. DBEngine will not extract any records unless they are in the manual subset, but it will retrieve updates during the tracking phase for all records satisfying the WHERE clause, whether or not they are in the manual subset. If *set_or_subset* does not allow duplicates, it becomes the primary set. When DBEngine enumerates the sets of a data set, it includes the specified subset.

Filter Examples

Example 1

The following examples show how to select all records that contain a data item that meets the WHERE criteria. The data items can be selected only from the specified data set.

```
SELECT ALL FROM BANK WHERE REGION-ID = 10;
SELECT * FROM BRANCH WHERE BRANCH-NAME = "Pittsburgh"
```

Example 2

The following two examples show how to select one or more data items from a data set.

This example shows one data item to be selected from the data set FINANCIAL-STATUS. Note that FS-2 is a Boolean.

```
SELECT FS-NAME FROM FINANCIAL-STATUS WHERE FS-2;
```

This example selects two data items (BRANCH-ID and BRANCH-AD1) for the data set named BRANCH.

```
SELECT BRANCH-ID BRANCH-AD1 FROM BRANCH
WHERE BRANCH-ID > 1
AND BRANCH-NAME = "SOUTH MANHATTAN"
AND BRANCH-AD1 NEQ " ";
```

Example 3

The following example shows how to select one or more data items from a variable-format data set.

```
SELECT    % variable-format dataset
1: AC-TYPE AC-NUMBER CUST-ID AC-BALANCE
AC-STAT1 BRANCH-ID AC-HLD-AMT
2: AC-TYPE AC-NUMBER AC-STAT2
3: AC-NUMBER, AC-STAT3, AC-TYPE
5: *
0: AC-NUMBER AC-DT-CLSE
FROM ACCOUNT WHERE BANK-ID = 1;
```

The following examples build on the previous example, using the ELSE: clause (assuming ACCOUNT has a record type 4):

```
filter  NoType4
%       -------
select 1: AC-TYPE AC-NUMBER AC-BALANCE AC-VF1-GROUP
BRANCH-ID AC-HLD-AMT
2: AC-TYPE AC-NUMBER AC-VF2-GROUP
3: AC-NUMBER AC-VF3-GROUP AC-TYPE
0: AC-NUMBER AC-DT-CLSE
% no type 4 in enumeration; no type 4 updates
else: none
from ACCOUNT where BANK-ID > 5000;
filter  WithType4
%       ---------
select 1: AC-TYPE AC-NUMBER AC-BALANCE AC-VF1-GROUP
BRANCH-ID AC-HLD-AMT
2: AC-TYPE AC-NUMBER AC-VF2-GROUP
3: AC-NUMBER AC-VF3-GROUP AC-TYPE
0: AC-NUMBER AC-DT-CLSE
% type 4 is in enumeration; type 4 updates
% must satisfy the WHERE clause
else: *
from ACCOUNT where BANK-ID > 5000;
filter  NoType4PerDefault
%       -----------------
default none
select 1: AC-TYPE AC-NUMBER AC-BALANCE AC-VF1-GROUP
BRANCH-ID AC-HLD-AMT
2: AC-TYPE AC-NUMBER AC-VF2-GROUP
3: AC-NUMBER AC-VF3-GROUP AC-TYPE
0: AC-NUMBER AC-DT-CLSE
% no type 4 in enumeration; no type 4 updates
from ACCOUNT where BANK-ID > 5000;
filter  Type4PerDefault
%       ---------------
default all
select 1: AC-TYPE AC-NUMBER AC-BALANCE AC-VF1-GROUP
BRANCH-ID AC-HLD-AMT
2: AC-TYPE AC-NUMBER AC-VF2-GROUP
3: AC-NUMBER AC-VF3-GROUP AC-TYPE
0: AC-NUMBER AC-DT-CLSE
% type 4 is in enumeration; type 4 updates
% must satisfy the WHERE clause
from ACCOUNT where BANK-ID > 5000;
```

Transforms

A transform is a special routine used for populating VIRTUAL data sets or transforming an update in other ways.

A transform is a user-written procedure that is used by the Support Library. A transform has access to both the before- and after-images at the same time.

The following transforms are predefined in the Support Library:

- ChangedRecordsOnly—Discards records if none of the data items allowed by the filter are modified.
- ChangedItemsOnly—Discards unmodified data items. The Accessory receives only the key items and data items that changed. Note that this transform is not supported by the Databridge Client.

Refer to the *Databridge Programmer's Reference* for instructions on creating transforms.

Error Manager

The Support Library contains a default error manager that displays all error messages. If desired, you can write a custom error manager that, for example, analyzes an error and logs it, displays it, and determines whether the Accessory should continue processing or terminate. To use a custom error manager, you must write the error manager and then declare it in the GenFormat parameter file. Refer to the *Databridge Programmer's Reference* for instructions on writing a custom error manager.

Use the following syntax to declare a custom written error manager in the GenFormat parameter file:

```
ERROR MANAGER errormanagername IN "patchfiletitle"
```

where *errormanagername* is the name you have given to the error manager and *patchfiletitle* is the file name of the ALGOL patch file containing the error manager.

ALTER and VIRTUAL Data Sets

Refer to the *Databridge Programmer's Reference* for instructions on creating ALTER or VIRTUAL data sets.

ALTER Data Set

Use the ALTER declaration when you want to reformat data items in a data set to different layouts (i.e., data item conversion). If you want to change date formats and are using Databridge Clients, however, it is often less expensive to use the date formats provided by the Databridge Client software. Common uses of the ALTER declaration include the following:

- Subdividing items.
- Merging adjacent items in the same parent group
- Formatting dates that cannot be done in the Client

VIRTUAL Data Set

VIRTUAL data sets allow you to create a structure that does not physically reside in the DMSII database, while appearing as a normal data set to the Accessories.

When to Use Primary Keys

DBEngine supports data sets whether or not they have unique key sets. Accessories, on the other hand, support data sets as follows:

- Span does not require that data sets have unique key sets.
- Snapshot requires data sets that don't have valid AAs, such as ORDERED and COMPACT, to have a unique key set. If the dataset does not have a unique key set, you can define a primary key via the GenFormat program.
- Databridge Clients require that data sets have unique key sets or valid AAs or RSNs. Use the GenFormat program to define primary keys for data sets, or set the key items manually in the client control tables.

Creating a Primary Key

Declare a primary key in the following circumstances:

- For VIRTUAL data sets that you want Databridge to update. Declaring the primary key allows Databridge to update the records in the VIRTUAL data set.
- For data sets where Databridge requires a unique key set, but the unique key set does not exist in the DMSII DASDL. Declaring the primary key satisfies the Databridge requirements, without any DASDL modifications.

To create a primary key, GenFormat allows you to specify the data items for a data set that form a primary key.

CAUTION: Databridge does not check the validity of the primary key you define. It is your responsibility to choose a primary key that is unique for all records in the data set. Failure to do so could result in duplicate or lost records.

Primary Key Syntax

Syntax: `PRIMARY KEY OF` *`dataset`* `IS (`*`keyitems`*`)`

Where	Is
PRIMARY	An optional word
KEY	A required word
OF	An optional word
dataset	The name of the VIRTUAL data set that you declared in the GenFormat parameter file -or- A data set in the DMSII database
IS	An optional word
keyitems	One or more data items that will make a unique key. Separate each data item with a comma. The parentheses are required.

Example:

The following example uses all of the key words and only one data item:

```
Primary Key of CUSTOMER is (CUST-ID)
```

The following example does not use the optional key words; it does, however, use a combination of data items to create a primary key.

```
KEY ORDER-DETAIL
(
ORD-NBR,
LINE-NBR
)
```

Creating and Using Translation Tables

The formatting routines you create can use translation tables to translate individual EBCDIC characters in text (ALPHA) data items. This is useful for converting between international character sets.

Syntax for Creating Translation Tables

Syntax:

TRANSLATION *tablename*

sourcevalue -> destinationvalue

Where	Is
tablename	The name of a table you create. The name must start with a letter followed by letters, numbers, or underscores. This is the name you will enter for the TRANSLATE option in the FORMAT declaration.
sourcevalue	The value that you want to translate. This value can be in any of the following forms: ◆ Decimal value of a character ◆ Hexadecimal value of a character, 0x*nn* where *n* is 0–9 or A–F ◆ Quoted character
->	The required characters between the source value and the destination value.
destinationvalue	The replacement value for the source. This value can be in any of the following forms: ◆ Decimal value of a character ◆ Hexadecimal value of a character, 0x*nn* where *n* is 0–9 or A–F ◆ Quoted character

Example:

```
Translation CRToLF % Convert Carriage Return to Line Feed
013 -> 037      % ? -> ?
```

—or—

```
0x0D -> 0x25
```

The percent sign (%) designates a comment.

Syntax for Using Translation Tables

The GenFormat parameter file supplies you with several translation tables. To use a translation table, you must add the translation table name to a FORMAT, as in the following example

```
FORMAT formatname
translate translationtablename
.
.
```

where *translationtablename* is the name of a translation table. The following translation tables are predefined in the DATA/GENFORMAT/SAMPLE/CONTROL file:

Translation Table	EBCDIC Translation Table
CRToLF	N/A
UnitedKingdom	UnitedKingdomEBCDIC
Denmark	DenmarkEBCDIC
Finland	FinlandEBCDIC
France	FranceEBCDIC
Germany	GermanEBCDIC
Italy	ItalyEBCDIC
Norway	NorwayEBCDIC
Spain	SpainEBCDIC
Sweden	SwedenEBCDIC
SwissFrench	SwissFrenchEBCDIC
SwissGerman	SwissGermanEBCDIC

5 Chapter 5: Server

This chapter describes the Databridge Server Accessory and how to use it.

How the DBServer Works

Databridge Server (DBServer) is installed automatically on the host when you install the Databridge software. On the Unisys MCP server, it interacts with DBEngine. On a remote system, DBServer interacts with the Databridge Client. DBServer supports communications over TCP/IP, and can support up to 100 simultaneous connections to Databridge Clients. DBServer is required for all Databridge Clients (including the DMSII Client and Databridge Twin). It is not required for Databridge host Accessories such as Span or Snapshot.

The following describes how DBServer works with other Databridge components in the replication and update process:

1. The Client calls DBServer to retrieve layout information for a DMSII database by calling DBEngine.
2. DBServer calls DBEngine to clone the database, and DBEngine accesses the DMSII database via DMSII access routines.
3. DBServer receives the records from DBEngine, calls the filter and format in the Support library, and then sends the records to the Client.
4. DBServer then calls DBEngine to retrieve updates that occurred during and following the extraction.
5. The DBEngine reads the audit trail and sends the information to DBServer, which calls the filter and format in the Support library.
6. DBServer sends the changed information from the audit file to the Client.

Databridge Client Access

DBServer must be running for a Databridge Client to connect to it. If it is not running, the Databridge Client tries to connect until it eventually times out.

Run DBServer

Follow these steps to edit the Databridge Server parameter file before you run Databridge Server.

To run DBServer

1. Edit DBServer parameter file (DATA/SERVER/CONTROL) and save the file when you're finished. See "DBServer Parameter File" on page 90.
2. Make sure that the DMSII database security attributes are set to give the Engine read access to the DMSII database and DMSII DESCRIPTION, CONTROL, and audit files. See the *Databridge Installation Guide* for information about configuring guard files.
3. From the usercode that contains the DBServer files, start DBServer as follows:

   ```
   START WFL/DATABRIDGE/SERVER
   ```

   After DBServer starts, it will sit idle in the mix until it receives a request from a Databridge Client. If DBServer fails to start, check the system messages on the host for the reason.
4. To check the status of a DBServer client process or to take DBServer out of the mix, see "DBServer Commands" on page 100.

DBServer Parameter File

The DBServer parameter file (DATA/SERVER/CONTROL) is a SEQDATA file type with 72 characters and a sequence number.

Typically, you would use only one DBServer parameter file for all DMSII databases that Databridge Clients are replicating from a particular host. The SOURCE declaration is what identifies each database. If you want to run Databridge Server in a test environment as well as a production environment, you could use multiple DBServer parameter files. (Do this by changing the name of DBServer control file from within DBServer WFL.) Each DBServer parameter file would then specify a different listening port and different databases and usercodes.

When a Databridge Client connects, Databridge Server automatically reloads the parameter file if it has been modified. You do not have to bring Databridge Server down to modify the parameter file unless you are changing network parameters, like the port number.

Note the syntax carefully:

- The literal SOURCE and the source name are followed by a colon (:).
- Commas and semicolons are interchangeable and optional. SOURCE declarations do not have to end with a semicolon.
- There is no termination character.
- There is no continuation character.
- The comment character is the percent sign (%). The comment character can appear anywhere on a line and anything after the comment character is ignored.
- To enter a value that is the same as a keyword (such as AUDIT) in the parameter file, enter that value in quotation marks.
- Boolean values can be expressed as follows:
  - TRUE, YES, or no value (empty) which defaults to TRUE
  - FALSE, NO

Global Options

You can use the following global options to customize the DBServer Parameter file.

AFTER COPY JOB

Syntax: `AFTER COPY JOB "WFLtitle"`

where *WFLtitle* is the name of the WFL job to run after DBEnterprise performs a file transfer. Since file transfers run under the usercode alias associated with the Windows user name, the title of the WFL job should include a usercode and family name. The job will run under the usercode specified in the file title. For example, the following will job will run under the ADMIN usercode:

```
after copy job "(ADMIN)WFL/XFER/MONITOR ON SYSTEM"
```

The WFL job must take two parameters. The first is the title of the MCP file that was copied. The second is a boolean that is true if the file was uploaded to the MCP environment and false if it was downloaded from the MCP environment.

AUTHORIZED CLIENT IP

Syntax: `AUTHORIZED CLIENT IP "ipaddress1" [, "ipaddress2" ...]`

The AUTHORIZED CLIENT IP option restricts client connections to certain IP addresses. This is a "global" option in that it applies to all sources. (See also the source option "CLIENT HOST" on page 95.)

Examples:

```
AUTHORIZED CLIENT IP "192.168.0.99"
AUTHORIZED CLIENT IP "10.10.1.45", "172.27.172.9", "10.17.1.2"
```

If the option is omitted, clients can connect from any IP address (but are still subject to any CLIENT HOST restrictions specified for the particular source).

Changes to this option require DBServer to be restarted to take effect.

AUTO CONNECT

Syntax: `AUTO CONNECT "hostname" PORT portnumber`

where *hostname* is the name of the host where the Databridge DMSII Client resides and *portnumber* is the port number that Databridge Server uses to connect to the Databridge DMSII Client. Note that you can list any number of AUTO CONNECT entries in the global section (before the first SOURCE declaration) of the DBServer parameter file.

NOTE: The Databridge DMSII Client is the only Databridge Client that supports the AUTO CONNECT feature. To use this feature, you must also configure the Databridge DMSII Client for AUTO CONNECT. Refer to your *Databridge DMSII Client Administrator's Guide*.

This option enables Databridge Server to notify a Databridge DMSII Client when additional audit file information becomes available. When notified, the Databridge DMSII Client will then establish a connection with Databridge Server to receive the additional audit file information.

The first time Databridge Server runs, it connects to the specified ports. If successful and the Databridge DMSII Client initiates contact with Databridge Server, Databridge Server will note the database the client uses and contact the Databridge DMSII Client when audit file information becomes available for that database.

PORT

When DBServer initializes, it ensures that no other program is already using the specified port. This prevents multiple copies of DBServer from using the same port.

`TCP/IP` `TCP/IP Port = number`

`[BUFFER size messagesize]`

where *number* is the port number on which DBServer will listen for incoming requests from Databridge Clients. The port number must match what the Databridge Clients use.

This value must be an integer between 1024 and 65535 that defines the TCP/IP port number. A range of port numbers is pre-allocated for TCP/IP and therefore cannot be used by DBServer. To avoid any conflicts, do not use any of the well-known sockets predefined for TCP/IP. Most of the well-known sockets are 500 and below. Most port numbers (sockets) over 1024 are available for use, but you should check with your TCP/IP administrator before setting the port number.

TIP: Many sites use the Databridge release number, such as 7000, as the port number, to isolate and simplify testing and integration of new releases while the earlier Databridge release continues to run.

For `BUFFER SIZE messagesize`, enter a port file record size (in bytes) that matches the message size in use by your network. In some situations, matching the message size to your network can improve throughput. In addition to setting the port file record size, this entry also sets the size of the buffer. Optimally, the buffer size should be a multiple of the TCP/IP segment size. By default, `BUFFER SIZE` is set to 8736 bytes.

PRINT STATISTICS

Syntax: `PRINT STATISTICS [ = ] [ TRUE | FALSE ]`

If specified before the first SOURCE, this is the default setting for each SOURCE. See the AX STATISTICS command in "DBServer Commands" on page 100.

WORKERS

Syntax: `WORKERS nn`

where *nn* is a number from 1–100.

NOTE: This option is not related to or affected by the DBEngine WORKERS setting.

This option enables you to set the total number of DBServer Worker stacks that can be active at one time. Each DBServer Worker stack enables one Databridge Client to retrieve information. So, if you want five Databridge Clients to be able to connect to DBServer simultaneously, set DBServer WORKERS option to 5.

When Enterprise Server is in “browse” mode (that is, it is initiated by users for viewing the various servers and sources), it may use several Workers. Generally, Enterprise Server will use one Worker for each source the user is actively browsing. Therefore, you may want to set the number of WORKERS to a large value, such as 20.

You can change this setting at run time by using the AX WORKERS command. See “DBServer Commands” on page 100.

Each Worker’s task name includes an ON clause identifying the remote workstation by either its IP address or host name. You can view this information by viewing active entries. If you are using guard files, you can enter the remote workstation’s ID in the title of an allowed Worker in your guard file. This enables you to allow specific workstations to access specific databases.

You can also determine the name of the remote workstation by using the *workertasknumber* AX STATUS command.

SOURCE Options

Use the following SOURCE options to customize individual sources.

SOURCE

Syntax: SOURCE *sourcename*

Databridge Clients connect to DBServer by using the specified port number. During the initial handshake, the client specifies which database to use by passing a source name to DBServer.

Enter a unique name for identifying the database, filter, etc., you are using. All sources must have a unique name. The actual source definition is provided by settings defined in the remainder of this section.

Your entry for SOURCE is the name the Databridge Client uses to request the layout information to define a data source.

AUDIT JOB

Syntax: AUDIT JOB "(*usercode*)*WFLname* ON *pack*"

The quotation marks are required.

Use this parameter to specify the name (and optional usercode and pack) of a WFL that you want DBServer to start after DBEngine has processed an audit file. DBServer passes the WFL an integer, which is the AFN (audit file number).

AUDIT JOB takes effect when both of the following circumstances are met:

- The AFN changes.
- There is a COMMIT.

You might use AUDIT JOB, for example, to initiate the Audit Remove WFL to remove an audit file from a secondary pack. For information, see “Audit Remove Utility” on page 188.

NOTE: You can combine the AUDIT ON and AUDIT JOB options as follows:

```
AUDIT ON packname JOB "WFLname"
```

AUDIT ON

Syntax: `AUDIT ["fileprefix"] ON media [OR media] [NO WAIT]`

where *media* is one of the following:

```
[ PRIMARY | SECONDARY ] alternatepack
[ PRIMARY | SECONDARY ] ORIGINAL FAMILY
[ PRIMARY | SECONDARY ] TAPE
```

The optional *fileprefix* parameter can specify a title prefix, enclosed in quotes. This enables processing of audit files that have been renamed. PRIMARY refers to the primary audit (for example, BANKDB/AUDIT1234) and SECONDARY refers to the secondary audit (for example, BANKDB/2AUDIT1234). If you don't specify PRIMARY or SECONDARY, then Databridge will look for both if the database is declared to have a duplicated audit.

Use one of the AUDIT ON parameters to specify where the audit files are stored. This parameter overrides the audit pack specified in the DMSII CONTROL file. This parameter does not specify where audit files are created; it specifies only where Databridge will look for audit files. During cloning, however, Databridge ignores the AUDIT ON parameter and reads from the active audit file on the original family where it is being written.

When the optional NO WAIT parameter appears, DBEngine will not wait when an audit file is missing. Instead, it will return DBM_AUD_UNAVAIL(7).

NOTE: You can combine the AUDIT ON and AUDIT JOB options as follows:

```
AUDIT ON packname JOB "WFLname"
```

Examples

In these examples, the database is called BANKDB and 1234 is the stored audit file. The DASDL has the following statement:

```
AUDIT TRAIL ATTRIBUTES (PACK = PAUDIT, DUPLICATED ON PACK = SAUDIT).
```

Modify the following examples to meet your specific needs:

```
AUDIT ON ORIGINAL FAMILY
% BANKDB/AUDIT1234 ON PAUDIT, or
% BANKDB/2AUDIT1234 ON SAUDIT
AUDIT ON SECONDARY SPARE
% BANKDB/2AUDIT1234 ON SPARE only
AUDIT ON PRIMARY ORIGINAL FAMILY
% BANKDB/AUDIT1234 ON PAUDIT
AUDIT ON SPARE OR ORIGINAL FAMILY
% BANKDB/AUDIT1234 ON SPARE, or
% BANKDB/2AUDIT1234 ON SPARE, or
% BANKDB/AUDIT1234 ON PAUDIT, or
% BANKDB/2AUDIT1234 ON SAUDIT
```

NOTE: If you want Databridge to look for the audit file on *alternatepack* and on the original family and on tape, you must use AUDIT ON *alternatepack* OR ORIGINAL FAMILY OR TAPE.

AUTHORIZED USERCODE

Syntax: AUTHORIZED USERCODE [=] usercode1 [OR usercode2 ...]

The AUTHORIZED USERCODE source option specifies which usercodes are allowed to access the source. For Windows clients, the user name of the person or process running the client must be a valid usercode on the MCP system or mapped to a usercode using the REMOTEUSER construct.

Usercodes should be in quotes if they could be confused with reserved words.

Example

```
source Bank:
   database = DESCRIPTION/BANKDB
   authorized usercode = BILLSMITH or "SUPPORT";
```

If an unauthorized user attempts to use a source, DBServer will return error 1003:

username at *clienthost* not authorized for source *sourcename*.

CLIENT HOST

Syntax: CLIENT HOST = "*host1*" [OR "*host2*" ...]

where *host* is either a valid client host name or IP address.

This option restricts Client access to DMSII databases. Only Databridge Clients running on the listed server(s) will be allowed to connect to this source. Otherwise, error 1033 is returned:

Invalid client host *host* for *source*.

CLONE

Syntax: CLONE [OFFLINE | ONLINE]

Use this parameter to specify whether Databridge should do an online extraction (the database is open for updates) or an offline extraction (the database is not open for updates). If you do not specify OFFLINE or ONLINE, DBServer will do an online clone. You might want to clone offline to avoid having to apply any fixup records to the extracts file. Note that if the database has INDEPENDENTTRANS set, DBEngine will use SECURE STRUCTURE to enforce an offline clone, which allows other programs to update the data sets not being cloned. Otherwise, an OPEN SINGLE UPDATE is performed to prevent other programs from updating the database at all during the clone.

If you specify OFFLINE, there will not be a fixup phase in the replication process since there will be no updates to the cloned data sets during the extract phase.

NOTE: When using Databridge Enterprise to clone a dataset, the CLONE OFFLINE option will be ignored if it is reading the dataset in Direct or Indirect mode. In this case Databridge Enterprise always performs an online clone.

DATABASE

Syntax: DATABASE [*logicaldatabase* OF] *description_title*

Enter the actual name of the DMSII database DESCRIPTION file or a logical database within a DESCRIPTION file. Include the usercode and the disk family on which the DESCRIPTION file resides if different from the usercode DBServer is running under.

Examples

```
DATABASE DESCRIPTION/BANKDB ON DISK
DATABASE LDB2 OF (ABC)DESCRIPTION/BANKDB
```

FILTER

Syntax: FILTER *filtername*

where *filtername* is the name of a GenFormat filter routine. To create your own filtering routine, see "Creating a Filter" on page 76.

This option specifies the name of a filtering procedure, which is an entry point in DBSupport. DBServer calls the filtering procedure for each data set record. Your entry for "SUPPORT" on page 100 determines which library DBServer calls.

KEY

Syntax: KEY "*password*"

Entering a key (password) is optional. You can enter a password with a maximum of 17 alphanumeric characters. If you do enter a password, Databridge Client users must enter this password in addition to the source name.

In DBServer, if the key is specified with quotes it will retain the case exactly. Without the quotes, the key will be uppercase.

Example

```
KEY = Secret     (Client must use - SECRET)
KEY = "Secret"   (Client must use - Secret)
```

CAUTION: The password is stored in the file without encoding or encryption. Therefore, make sure you limit access to the DBServer parameter file.

NOTIFY

Syntax: NOTIFY "*ipaddress*" PORT *number*

where *ipaddress* is the IP address of Enterprise Server and number is the port number on the Enterprise Server computer where the Databridge Director service will listen for notifications from DBServer.

This option enables DBServer to notify the Client via Enterprise Server when audit files become available. You can define multiple NOTIFY options and multiple SOURCEs can NOTIFY the same PORT.

NOTE: This feature works in conjunction with the Notify WFL (page 178) and the Copy Audit Utility (page 186).

PRINT STATISTICS

Syntax: `PRINT STATISTICS [ = ] [ TRUE | FALSE ]`

If part of a SOURCE declaration, this option determines whether a statistics report is printed for a Worker using that source. See the AX STATISTICS command in "DBServer Commands" on page 100.

PRIORITY

Syntax: `PRIORITY [=] number`

where *number* is between 0 and 99.

This option enables you to specify the priority a Worker should have when serving this source. When stacks compete for CPU resources, a stack with a higher priority will get the CPU before one with lower priority.

Example

```
SOURCE BANKDB:
...
PRIORITY = 65
...
```

READER

Syntax: `["readerprogram"] USING "directory"`

where *directory* is the directory containing the flat files you want to replicate.

This option is only used with Databridge FileXtract and does not appear in the sample DBServer parameter file. It does appear in DATA/SERVER/SAMPLE/FILEBRIDGE/CONTROL however, and you can insert all or part of that file in DATA/SERVER/CONTROL.

SOURCE LIKE

Syntax: `SOURCE sourcename LIKE sourcename: attribute_overrides;`

A SOURCE can use another SOURCE for default values. This is useful when you want to declare several SOURCEs with similar attributes. The following is a sample entry:

```
SOURCE KPMDB:
     DATABASE = DESCRIPTION/KPMDB ON DISK,
     USERCODE = HRD,
     FILTER =   KPROP,
     SUPPORT =  OBJECT/DATABRIDGE/SUPPORT/KPMDB;

SOURCE KPMDBONLYBANK1 LIKE KPMDB:
     FILTER =   ONLYBANK1;
```

In this example, KPMDBONLYBANK1 has all of the same options as KPMDB except that it uses a different filter.

If the target source is not found, DBServer displays the following error message:

`DBServer: Source not found (at line nnn): targetname`

STOP

Syntax: `STOP { BEFORE | AFTER } { timedate | "taskname" }`

Enter this command to stop Databridge from returning updates beyond a specified quiet point. The quiet point can be before or after a specified time, day, program, or a specific program on a specific date. For example, if you wanted to limit daily transactions to only those processed before 4:00 p.m., you would configure the STOP command to stop processing at the last quiet point before 4:00 p.m.

Time in the STOP command refers to the time the update occurred on the primary system, not the current time of day.

The "+/- days" are in relation to the DBServer Worker start date. For example, when the Worker starts, it calculates an audit file STOP date based on the current date plus or minus the "+/- days" parameter. The replication stops when it reaches an audit location with this calculated date.

The following table explains the STOP parameters:

STOP Options	Description
`STOP { BEFORE | AFTER } hh:mm [ AM | PM ]`	Replication stops at either the last quiet point before or the first quiet point after the designated time. Note that the time entry hh:mm can be either 24-hour time format (22:30 for 10:30 p.m.) or 12-hour format (10:30 p.m.) For example, the following command informs replication to stop at the last quiet point before 10:30 p.m. `STOP BEFORE 10:30 PM`
`STOP { BEFORE | AFTER } { hh:mm [AM | PM] } ON [+ | -] days`	Replication stops at either the last quiet point before or the first quiet point after the designated time and after the designated number of days. Note that the time entry hh:mm can be either 24-hour time format (22:30 for 10:30 p.m.) or 12-hour format (10:30 p.m.). Also note that a days entry of zero indicates the current day. For example, the following command causes replication to stop processing at the last quiet point that occurs tomorrow before 10:30 p.m. `STOP BEFORE 22:30 ON + 1`

STOP Options	Description
STOP { BEFORE | AFTER } { *hh:mm* [AM | PM] ON *mm/dd/yyyy* }	Replication stops at either the last quiet point before or the first quiet point after the designated time on the designated day. Note that the time entry hh:mm can be either 24-hour time format (22:30 for 10:30 P.M.) or 12-hour format (10:30 P.M.). For example, the following command causes replication to stop at the first quiet point after 1:30 p.m. on December 9, 2009. `STOP AFTER 13:30 ON 12/9/2009`
STOP { BEFORE | AFTER } { "*taskname*" }	Replication stops processing at either the last quiet point before the BOT or the first quiet point after the EOT of the designated task name. If you specify a usercode, Databridge looks for a task name match under the specified usercode. Otherwise, Databridge looks for task name match under any usercode. For example, the following command causes replication to stop at the first quiet point after the EOT of task name "OBJECT/SAVINGS/POSTING". `STOP AFTER "OBJECT/SAVINGS/POSTING"`
STOP { BEFORE | AFTER } { "*taskname*" | { BEFORE | AFTER } *hh:mm* [AM | PM] ON *mm/dd/yyyy* }	Depending on which occurs first, replication stops at either the last quiet point before the designated task started or the first quiet point after the designated task ended OR at either the last quiet point before or the first quiet point after the designated time on the designated date. For example, the following command causes replication to stop at the first quiet point after the EOT of task name "OBJECT/SAVINGS/POSTING" on December 9, 2009 or at the last quiet point before 10:30 p.m. on December 9, 2009, whichever occurs first: `STOP AFTER "OBJECT/SAVINGS/POSTING" OR BEFORE 10:30 PM ON 12/9/2009`

Keep in mind the following when using the STOP command:

- If you do not specify a usercode on the taskname, replication will stop at a task with any usercode in the task name.
- If a STOP BEFORE "*taskname*" command is specified, replication stops at the last quiet point before the task opened the database.
- If a STOP AFTER "*taskname*" command is specified, replication stops at the first quiet point after the task closed the database.
- If more than one task name is specified, only the last one specified is used. Similarly, if more than one time/date is specified, only the last time/date specified is used.

SUPPORT

Enter the filename of the support library, using the following table as a guide.

Library	Entry
Non-tailored support library	OBJECT/DATABRIDGE/SUPPORT
Tailored support library	OBJECT/DATABRIDGE/SUPPORT/*dbname*

If a Worker is unable to link to the DBSupport Library for any reason, the Worker will try to recompile it.

NOTE: If you make any changes to the SUPPORT entry, you must run the redefine command on the Client. For instructions, see the *Databridge Client Administrator's Guide*.

TRANSFORM

Syntax: TRANSFORM *transformname*

where *transformname* is the name of a GenFormat transform routine. To create a transform routine, see "Transforms" on page 83.

This option specifies the name of a transform routine, which is an entry point in the DBSupport Library. DBServer calls the transform routine in DBSupport for each data set record. Your entry for "SUPPORT" on page 100 determines which DBSupport library DBServer calls.

USERCODE

Syntax: USERCODE *worker_usercode*

This entry specifies the usercode (and any associated FAMILY substitution) under which the DBServer Worker stack will run. DBServer initiates the stack as a process when the Databridge Client connects to DBServer and then the Worker calls DBChangeUser to change the usercode of the stack prior to opening the DMSII database.

DBServer Commands

Use the following commands when running DBServer. The *servertasknumber* in these commands is the task number of the main DBServer (OBJECT/DATABRIDGE/SERVER) stack. A *workertasknumber* is the task number of one of the DBServer Workers (DBSERVER/WORKER/*n*). Each DBServer Worker is connected to exactly one Databridge client.

AX BUFFER — Displays or sets the TCP/IP buffer size. See comments on BUFFER SIZE.

servertasknumber AX BUFFER [*nn*]

The *nn* value is the size of the TCP/IP buffer new Workers will use.

AX HELP

To display the AX commands that you can use with DBServer, enter the following command:

servertasknumber AX HELP

Or

workertasknumber AX HELP

AX QUIT

Typically, most sites leave DBServer in the mix. That way, DBServer is available for the Databridge Clients. However, to quit DBServer, use the following command:

servertasknumber AX QUIT [*source*]

where the optional *source* is the particular SOURCE to be terminated, in which case all Workers using that source will be terminated. If *source* is not specified, DBServer terminates after all of the Workers have completed.

If needed, you can terminate individual Worker stacks without terminating the DBServer.

workertasknumber AX QUIT

The Worker terminates after the next commit.

AX SIZES

Displays various size-related statistics: network read and write, audit record, and transaction group. The number in parentheses is the count of samples of that category. The four numbers at the end of the displayed line are the minimum, maximum, average, and total.

workertasknumber AX SIZES

AX STATISTICS

Displays or sets the option enabling or disabling printing the statistics report when a Worker finishes.

workertasknumber AX STATISTICS [TRUE | FALSE]

This command will override the PRINT STATISTICS option in the parameter file for this Worker.

servertasknumber AX STATISTICS [TRUE | FALSE]

This command affects all sources. If printing is disabled, Workers will not print a statistics report. If printing is enabled, only sources having PRINT STATISTICS = TRUE in the parameter file will have statistics reports.

AX STATUS

You can use the AX STATUS command to display status information for either the main DBServer stack or a Worker stack.

Server stack

servertasknumber AX STATUS

Displays the transport type, port number, number of offered port subfiles, the Worker limit, the reason it is waiting, and the client host name and source name for each Worker. For example:

```
DBServer: Listening on TCP/IP port 5000
DBServer: Subports offered: 1
DBServer: Workers active: 3, Limit: 10
DBServer: Waiting for new connection or Worker event
DBServer:(3) DBHOST accessing BANKDB via TCP/IP port
5000/3
DBServer:(2) DEMOHOST accessing DEMODB via TCP/IP port
5000/2
DBServer:(1) DBHOST accessing CUSTDB via TCP/IP port
5000/1
```

The number in parentheses after "DBServer:" is the Worker number.

Worker stack

workertasknumber AX STATUS

Displays the Client host name, the source name, the transport type, the port number, the audit location, the number of records processed, any STOP condition, the size of message segments, and the size of the message buffer. For example:

```
DBServer:(2) DEMOHOST accessing DEMODB via TCP/IP port
5000/2
DBServer:(2) Protocol level: Client=15 Server=15
DBServer:(2) Port segment size: 4368, Buffer size: 8736
DBServer:(2) AFN=182, ABSN=5845, January 9, 2010 @
08:41:09
DBServer:(2) Records sent=50, skipped=17, Commits=3
DBServer:(2) Waiting for client request
```

AX TIMES

Displays various time-related statistics: Network read and write, DBRead wait, thread lock waits, and total processor time.

workertasknumber AX STATUS

The number in parentheses is the count of samples of that category. The four numbers at the end of the displayed line are the minimum, maximum, average, and total.

AX WORKERS

Displays or limits the current number of active Worker stacks.

To display the limit, enter:

servertasknumber AX WORKERS

To limit the number of active Worker stacks, enter:

servertasknumber AX WORKERS *nn*

where *nn* is a number from 1 to 100. This command will not terminate any active Workers. DBServer uses this number only when it is considering whether to offer additional subports for new client connections.

Troubleshooting DBServer Usercodes

Ensure that OBJECT/DATABRIDGE/CHANGEUSER has the TASKING attribute set. This is required for DBServer to change the usercode as specified in DBServer parameter file.

For information on how the usercode under which you run DBServer can affect the usercodes for the stack initiated for each Databridge Client, see “USERCODE” on page 100.

If a DBServer Worker is unable to connect to the proper database, check this USERCODE entry. This usercode must ensure that DBEngine has visibility to the DBEngine parameter file (`DATA/ENGINE/CONTROL`).

6 Chapter 6: Span

This chapter describes the Databridge Span Accessory and how to use it.

In this Chapter

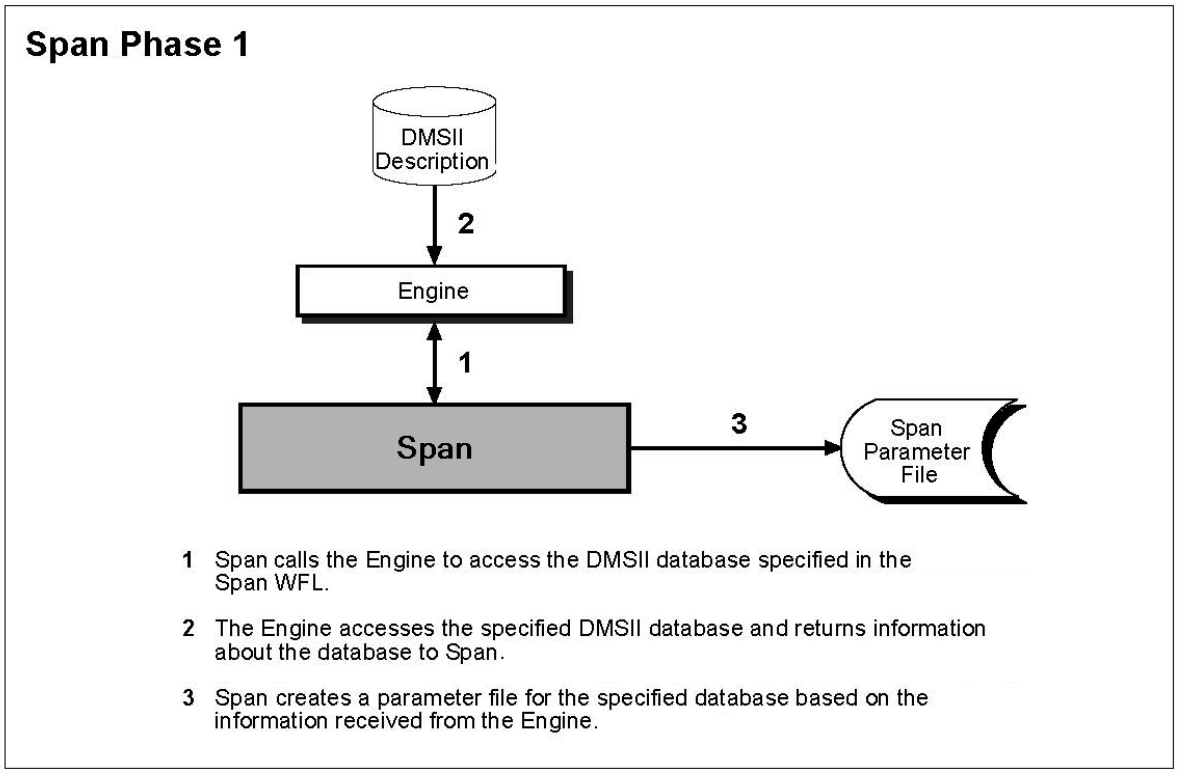
How Span Works

The Span Accessory provides host-based replication of a DMSII database. Use Span to create data files that contain copies of DMSII database records. This approach is useful when a network is not available for a Databridge server-client relationship or when a Databridge Client does not exist for a particular client system.

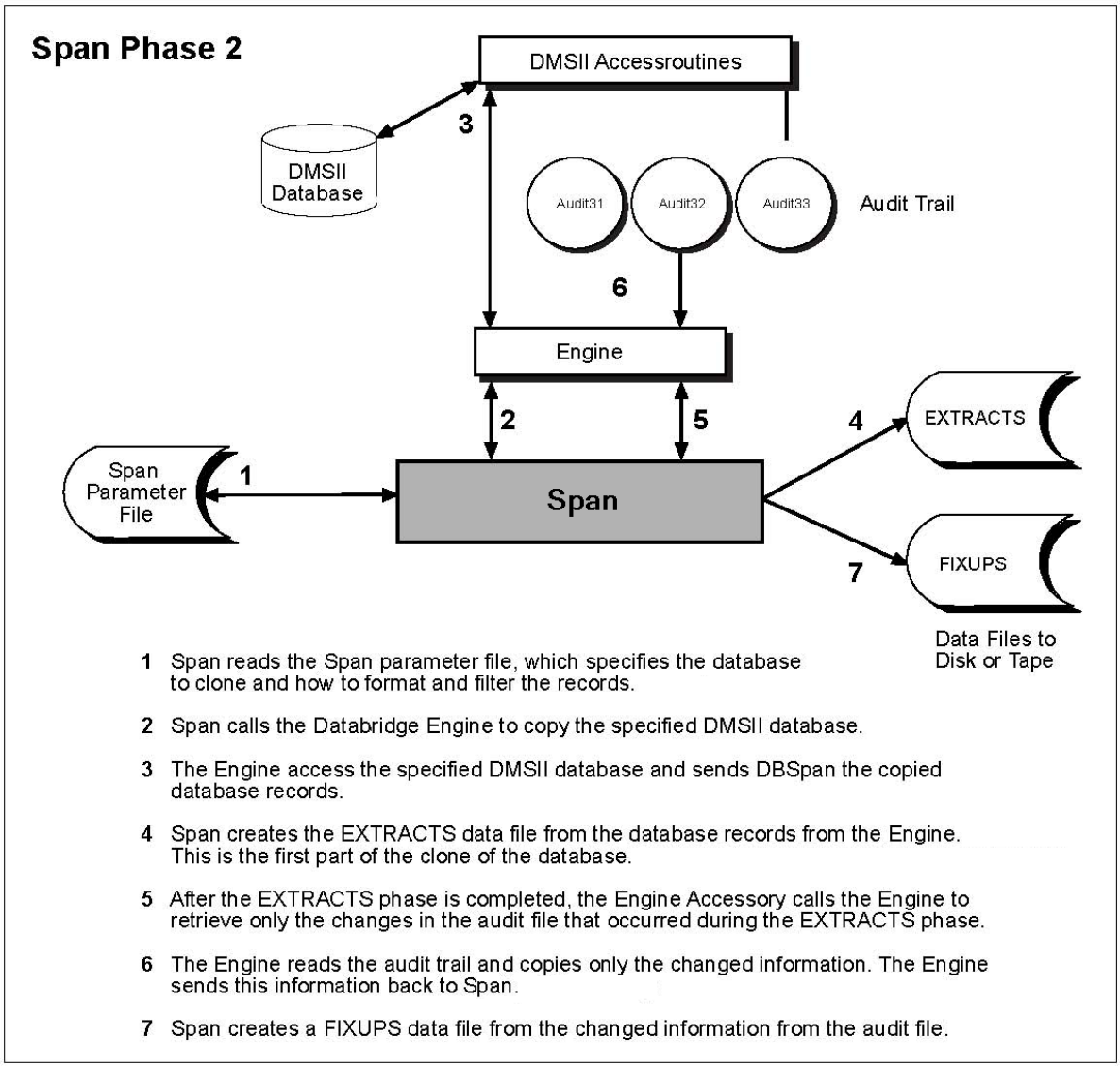
When Span requests updates for data sets that have not been cloned, the Databridge Engine (DBEngine) initiates a routine to extract all of the records for the uncloned data sets from the DMSII database. DBEngine returns these records to Span as if they were newly-created records. DBEngine also forwards any additional changes that occurred to these records during the extraction. (This extraction may include duplicate records or other anomalies.)

There are three phases when using Span:

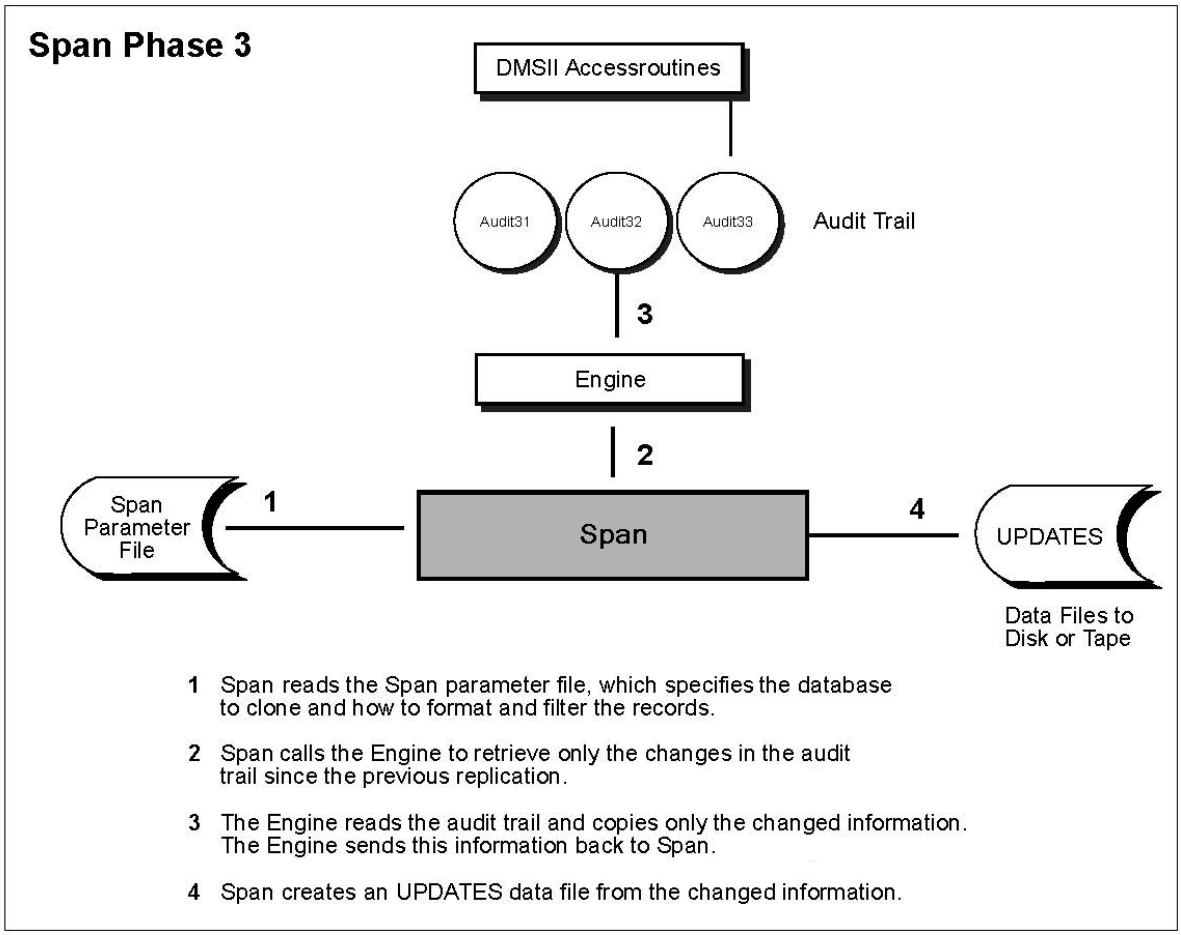
- Phase 1—Run Span to generate a Span parameter file for the database to be replicated.



- Phase 2—Run Span to replicate the database.



- Phase 3—Run Span to update the replicated database based on changes in the audit file since the initial replication or since the previous update.



Span and Snapshot Compared

The Span and Databridge Snapshot Accessories both extract the DMSII database to flat sequential disk files or tape. The following table includes a comparison of these Accessories to help you choose the most suitable replication method.

Span	Databridge Snapshot
Most useful when you have frequent, ongoing database changes that you need to track.	Most useful for one-time clones. Also useful for gathering periodic information (such as month-end information that can be queried from a database).
Cleanup and consolidation occur on the secondary database system (requires fewer mainframe resources than Snapshot).	Cleanup and consolidation occur on the mainframe.
Maintains audit location information about each replicated data set. This information is used to determine where to start retrieving changes that it finds in the audit trail. This quiet point location is stored in the DATA/SPAN/databasename/CONTROL file on the host.	Records in the data files created by Snapshot represent the same QPT (quiet point) in the audit trail.

Span	Databridge Snapshot
Provides updates to the secondary database (only database *changes* are gathered).	When you update the database, you must run Snapshot again to reclone it.
When Span requests updates for data sets that have not been cloned, DBEngine initiates a routine to extract all of the records for the uncloned data sets and returns these records to Span as newly-created records. In addition, DBEngine forwards any changes that occurred to these records during the extraction.	
Span's update data must be loaded to the secondary database (for example, via a user-written program).	Snapshot's output data files must be transported to the secondary database.

Span WFL

This WFL (WFL/DATABRIDGE/SPAN) starts Span. You can modify it for the:

- QUEUE—If, for example, you want Span to enter the system through job queue 10, you must modify the WFL job that compiles Databridge software to include the QUEUE (or CLASS) = 10 declaration.
- STARTTIME
- BDNAME—This is the default file naming convention for the printer file. The name of the printer file that is generated by default is DBBD/RUN/SPAN/*dbname*.

Span TASKVALUEs

WFL/DATABRIDGE/SPAN also includes support for reporting the success or failure of Span. To do so, Span sets its TASKVALUE (same as VALUE) to either a positive or negative number.

When Span encounters any errors from DBEngine (*DBMnnn*), it returns the *nnn* number as a positive TASKVALUE. These *DBMnnn* values are listed in the *Databridge Error and Message Guide*, included with the product documentation.

When Span detects an error specific to itself (for example, no such parameter file), it returns a negative number, as listed in the following table.

In WFL/DATABRIDGE/SPAN, the TASK variable is named SPAN and the value is returned as SPAN (VALUE). The following sample WFL code displays the SPAN (VALUE).

Example:

```
IF SPAN(TASKVALUE) < Ø THEN
DISPLAY "SPAN TASKVALUE = -" & STRING(SPAN(TASKVALUE), *);
ELSE
DISPLAY "SPAN TASKVALUE = +" & STRING(SPAN(TASKVALUE), *);
```

TASKVALUE	Description
TV_USER_TERMINATED = -1	An operator entered the AX QUIT command to terminate Span.

TASKVALUE	Description
TV_CONTROL_INUSE = -2	The Span parameter file is in use by another version of Span. Wait until that version is finished and try again.
TV_CONTROL_CREATED = -3	This value is returned when you run Span for the first time. It indicates that the Span parameter file was successfully created.
TV_HEADER_MISMATCH = -4	This value indicates that the HEADER option is set to TRUE. Span is set to output to a consolidated file, but one of the following occurred: ◆ The file is missing the header. ◆ The header is corrupted. ◆ The new starting location does not match the old ending location.
TV_WRITE_ERROR = -5	This value indicates that Span is attempting to write output, but an I/O error occurred, resulting in Span terminating.
TV_CF_SYNTAX = -6	This value indicates a syntax error in the Span parameter file. Make the correction in the parameter file and try again.
TV_RECOMPILE_FORMAT = -7	This value indicates that the formatting entry point in the Support Library does not match the database layout. This is usually the result of a data set reorganization. Recompile the Support Library and then rerun Span.
TV_DUP_ENTRY = -8	This values indicates duplicate entries in the Span parameter file. When there are two entries for the same data set, Span terminates with TASKVALUE = -8.
TV_AUDITFILE_LIMIT = -9	This value indicates that Span has processed the number of audit files specified by the STOP AFTER parameter.
TV_RECORD_LIMIT = -10	This value indicates that Span has reached the output file size limit specified by the STOP AFTER parameter.
TV_AUDITTIME_LIMIT = -11	This value indicates that Span has stopped processing because of the time specified by the STOP AFTER parameter.
TV_AUDITNAME_LIMIT = -12	This value indicates that Span has stopped processing because of the task name specified by the STOP parameter.

Run the Span Accessory

You must run Span twice to replicate a database, as follows:

- First, to create a parameter file for the database you specify
- Second, to actually perform the replication

Once the database is replicated, run Span to perform the gathering and propagating of updates from the original database.

For subsequent runs of Span, you can use the existing Span parameter file for the database you are cloning. In other words, once you run Span to create the parameter file, you do not need to create the parameter file again.

To run Span:

1 Run Span without a parameter file, but with the name of the database to be replicated.

```
START WFL/DATABRIDGE/SPAN ("databasename" [,"logicaldatabasename"])
```

Where	Is
"*databasename*"	The title of the DESCRIPTION file without the DESCRIPTION node. The database name can include a usercode and pack, which are used to locate the database DESCRIPTION file, as follows: "*(usercode)databasename ON packname*" The quotation marks are required. When you enter just a database name, Span creates a parameter file named as follows: (*usercode*)DATA/SPAN/*databasename*/CONTROL ON *familyname*
"*logicaldatabasename*"	An optional name of a logical database when you want to do one of the following: ◆ Clone the data sets and data items in the logical database instead of in the physical database. ◆ Create an alternate parameter file for the physical database, which is useful when you want to generate multiple support libraries for the same database but with different parameter files. When you enter a logical database name: ◆ Span uses DATA/SPAN/*dbname*/*ldbname*/CONTROL as the title of the parameter file, if it exists ◆ Otherwise, Span uses DATA/SPAN/*ldbname*/CONTROL as it did in previous releases, if it exists ◆ Otherwise, if neither file exists, Span creates one titled DATA/SPAN/*dbname*/*ldbname*/CONTROL The logical database name provides a means for having multiple parameter files for the same database. If *ldbname* is the name of a logical database within the *dbname* database, Span uses that logical database. If not, it uses *ldbname* to specify an alternate parameter file.

2 Use CANDE to edit the Span options in the parameter file that Span just created. To do so, use the parameter descriptions listed in "Span Parameter File" on page 113. These parameters determine where output is located, types of formatting and filtering routines to use, etc.

3 For each data set you want to replicate, replace the comment sign (%) that precedes the replication status information (structure number and state information) with a space. The replication status information looks similar to the following:

```
 00005 000  0024 0000000866  0000340  00010 20140611135255 2 00000 00000
```

If you prefer, you can also uncomment the actual data set name. However, uncommenting the data set name has no effect on replication. You must uncomment the replication status information for the data set to be replicated.

4 Save the edited the Span parameter file.

5 Run Span again, this time with the edited parameter file.

START WFL/DATABRIDGE/SPAN ("*databasename*" [,"*logicaldatabasename*"])

where *databasename* and *logicaldatabasename* indicate the database names you used in step 1.

NOTE: If the message "No data set entries in parameter file" appears, make sure that you uncommented the replication information for each data set. Although you can uncomment data set names, Span reads only the replication information to clone a data set. For an example, see "Sample Span Parameter File" on page 130.

---

6 While Span is running, you can view the number of records it has replicated by using the AX STATUS command (page 142). Span creates the output data files for the data sets that are to be replicated as listed in the parameter file. If Span is writing the output data files to disk, skip to step 8.

If Span is writing to tape, dismount the tapes as soon as Span closes the output data file destined for that tape.

For example, Span closes an output data file when both of the following conditions occur:

- The mode (Extract, Fixup, or Update) changes
- The destination for the new mode is different than that for the old mode

7 When Span is finished, move the output files to the secondary database.

8 Back up the Span parameter file. This is essential if you need to reprocess from this point later. For example, if you do a rollback, you will use these parameter files. Keep in mind that the audit locations for synchronizing data set updates are written to the parameter file each time you run Span.

9 For the next step, you may want to do one or more of the following:

- Replicate a new database. Repeat steps 1-8 in this procedure.
- Determine how often to run Span for gathering updates. See "Automating Span Processing" on page 142 and "Span Tracking" on page 143.
- Recover from a reorganization or rollback. See Chapter 10, DMSII Reorganizations and Rollbacks (page 199).

Span Parameter File

The first time you run Span on a particular database, it creates a parameter file for that database. This parameter file contains audit synchronization information (STATEINFO) for every data set in the database. You can then edit the parameter file Span created and designate the data sets you want to replicate. When you run Span the second time, it uses the parameter file and replicates the data sets you designated.

Span uses the parameter file to keep track of which data has been replicated. The parameter file also contains options that affect how Span processes the records. The default title for the Span parameter file is as follows:

DATA/SPAN/*databasename*/CONTROL

For the format of the Span parameter file, note the following:

- For all TRUE entries, you can use the synonym YES; likewise, for all FALSE entries, you can use the synonym NO.
- You can enter the parameters in the parameter file in any order.
- You can enter multiple parameters on a single line.
- You can split parameters across multiple lines except for data set replication status lines.
- There is no termination character.
- There is no continuation character.
- The comment character is the percent sign (%). The comment character can appear anywhere on a line and anything after the comment character is ignored.
- If you have a file name or family name that is the same as a parameter file keyword, enclose the file name or family name in quotation marks. For example, if you have a family named SUPPORT (which is also the name of an option in the Span parameter file) and you want to enter that for the AUDIT ON option, enclose SUPPORT in quotation marks, as follows:

  ```
  AUDIT ON "SUPPORT"
  ```

  This means that the audit files are on a family called SUPPORT.
- All parameters are optional except for the list of data set replication statuses.
- If a data set name has a hyphen, you must enclose the data set name in quotation marks.

  The remainder of this section describes the syntax and semantics of each parameter. However, before you edit or create a Span parameter file, read “Run the Span Accessory” on page 110.

  The first part of the parameter file contains all of the Span options listed in this section. Edit these as necessary.

Determining Output

The first three options in the Span parameter file determine where Span writes its output. There are three types of Span output, and you can specify the same or different locations for each output type, as follows:

- EXTRACTS—These output files contain the records that Span gathers from the DMSII database during the clone.

- FIXUPS—These output files contain the records that are recorded in the audit trail during the extraction phase of cloning. Fixup records are any update records that are written to the audit file while cloning takes place. If the database was not updated during the extraction phase, there will be no FIXUPS files.
- UPDATES—These output files contain the update records that Span retrieved from the audit files since the end of the fixup phase.

To direct the output files to the same location, specify the location for the UPDATES option and leave the other two options (EXTRACTS and FIXUPS) blank. In this case, all output is written to the location specified by UPDATES.

Note, however, that the *filename* and *devicename* parts of the EXTRACTS, FIXUPS, and UPDATES options default separately, as in the following examples:

Example 1

In this example, all UPDATES, FIXUPS, and EXTRACTS files are named with the file title *nodes/filename* and are written to DISK.

```
EXTRACTS
FIXUPS
UPDATES      nodes/filename ON DISK
```

Example 2

In this example, the UPDATES and FIXUPS files are named with the file title *nodes/filename* and are written to DISK. The EXTRACTS files are named with the file title *nodes/filename* but are written to TAPE.

```
EXTRACTS      ON TAPE
FIXUPS
UPDATES       nodes/filename ON DISK
```

Example 3

In this example, the UPDATES files are named with the file title *nodes/filename* and are written to DISK. The FIXUPS files are named A/B/= and are written to DISK. The EXTRACTS files are named with the file title *nodes/filename* but are written to TAPE.

```
EXTRACTS      nodes/filename ON TAPE
FIXUPS        A/B/=
UPDATES       nodes/filename ON DISK
```

Example 4

In this example, the UPDATES files are named with the files title *nodes/filename* and are written to DISK. The FIXUPS files are named *nodes/filename* and are written to PACK. The EXTRACTS files are named with the file title *nodes/filename* and are written to TAPE.

```
EXTRACTS      nodes/filename ON TAPE
FIXUPS        nodes/filename ON PACK
UPDATES       nodes/filename ON DISK
```

File Title and Device Names

The file title and device name designations default separately. If you specify EXTRACTS ON TAPE, Span uses the file title designation of the UPDATES. Even if UPDATES are designated to be written to DISK, the EXTRACTS will be written to TAPE.

Using this same example, if you specify FIXUPS A/B/= with no device name, the FIXUPS files will be located in the place designated by UPDATES, in this case, DISK. The FIXUPS files will be named A/B/*datasetname*.

NOTE: Normal family substitution applies to the device name DISK. For example, if your family substitution statement is FAMILY DISK = DATAPACK OTHERWISE HLDISK, Span will write the output files to DATAPACK rather than the disk called DISK.

AUDIT JOB

Syntax: `AUDIT JOB "(`*`usercode`*`)`*`WFLname`* `ON` *`pack`*`"`

The quotation marks are required.

Use this parameter to specify the name (and optional usercode and pack) of a WFL that you want Span to start after it has processed an audit file. Span passes the WFL an integer, which is the AFN (audit file number).

AUDIT JOB takes effect when both of the following circumstances are met:

- The AFN changes.
- There is a COMMIT.

You might use AUDIT JOB, for example, to initiate the Audit Remove WFL to remove an audit file from a secondary pack. For more information, see "Audit Remove Utility" on page 188. You can combine the AUDIT ON and AUDIT JOB options as in the following syntax:

`AUDIT ON` *`packname`* `JOB "`*`WFLname`*`"`

AUDIT ON

Syntax: `AUDIT ["`*`fileprefix`*`"] ON` *`media`* `[OR` *`media`*`][NO WAIT]`

where *media* is one of the following:

```
[ PRIMARY | SECONDARY ] alternatepack
[ PRIMARY | SECONDARY ] ORIGINAL FAMILY
```

The optional *fileprefix* parameter can specify a title prefix, enclosed in quotes. This enables processing of audit files that have been renamed.

When the optional NO WAIT parameter appears, DBEngine will not wait when an audit file is missing. Instead, it will return DBM_AUD_UNAVAIL(7).

PRIMARY refers to the primary audit (for example, BANKDB/AUDIT1234) and SECONDARY refers to the secondary audit (for example, BANKDB/2AUDIT1234). If you don't specify PRIMARY or SECONDARY, then Databridge will look for both if the database is declared to have duplicated audit.

Use one of the AUDIT ON parameters to specify where the audit files are stored. This parameter overrides the audit pack specified in the DMSII CONTROL file. You can combine the AUDIT ON and AUDIT JOB options as in the following syntax:

```
AUDIT ON packname JOB "WFLname"
```

NOTE: The AUDIT ON parameter does not specify where audit files are created; it only specifies where Databridge will look for audit files. During cloning, however, Databridge ignores the AUDIT ON parameter and reads from the active audit file on the original family where it is being written.

The following are examples of AUDIT ON parameters. In these examples, the database is called BANKDB and 1234 is the stored audit file. The DASDL has the following statement:

```
AUDIT TRAIL ATTRIBUTES (PACK = PAUDIT, DUPLICATED ON PACK = SAUDIT).
Modify the following examples to meet your specific needs:

AUDIT ON PRIMARY ORIGINAL FAMILY
% BANKDB/AUDIT1234 ON PAUDIT
AUDIT ON SPARE OR ORIGINAL FAMILY
% BANKDB/AUDIT1234 ON SPARE, or
% BANKDB/2AUDIT1234 ON SPARE, or
% BANKDB/AUDIT1234 ON PAUDIT, or
% BANKDB/2AUDIT1234 ON SAUDIT
```

CLONE

```
Syntax: CLONE [ OFFLINE | ONLINE ]
```

Use this parameter to specify whether Databridge should do an online extraction (the database is open for updates) or an offline extraction (the database is not open for updates).

If you do not specify OFFLINE or ONLINE, Span will do an online clone. You might want to clone offline to avoid having to apply any fixup records to the extracts file. If you specify OFFLINE, there will not be a fixup phase in the replication process since there will be no updates to the cloned data sets during the extract phase.

If the database has INDEPENDENTTRANS set, DBEngine will use SECURE STRUCTURE to enforce an offline clone. This allows other programs to update the data sets not being cloned. Otherwise, an offline dump is simulated.

DEFAULT MODIFIES

The DEFAULT MODIFIES option controls what Span writes to its updates files for all data sets except those for which you override DEFAULT MODIFIES, as explained in the next section.

The DEFAULT MODIFIES options are as follows:

```
DEFAULT MODIFIES   AFTER IMAGES ONLY
DEFAULT MODIFIES   BEFORE AND AFTER IMAGES
```

Use only one of these two options at a time. If both are uncommented, the last one in the list takes precedence.

You can override this value for individual data sets. See “Individual Data Set Options” on page 129.

Selecting DEFAULT MODIFIES

Span can write the modified record only (AFTER IMAGE), or it can write the record as it looked before the modification (BEFORE IMAGE) and how it looked after the modification (AFTER IMAGE).

Typically, your decision on whether or not to save AFTER IMAGES ONLY or save BEFORE AND AFTER IMAGES is based on what you will do with the output file. Use the following table as a guide for which DEFAULT MODIFIES option to select.

DEFAULT MODIFIES	Description
AFTER IMAGES ONLY	Span writes only the resulting change to the Span output file. In other words, Span writes the record the way it looks after the update. Selecting AFTER IMAGES ONLY is useful if your database has keys that are unique and that won't change (for example, a customer number). This way, a client database can always find records via the key and therefore process any changed records.
BEFORE AND AFTER IMAGES	In this case, Span writes both the before image and the after image to the Span output file. This is the same as converting any modification of an existing record to a DELETE/CREATE pair (DELETE contains the before image; CREATE contains the after image). Selecting BEFORE AND AFTER IMAGES is useful when the DMSII database allows key fields to be updated. It ensures that even if the key of a record is changed, the client database can process the change correctly. For example, when the output file contains before and after images, the client software can search for the before image and find a match before it makes the update. The disadvantage of selecting BEFORE AND AFTER IMAGES is that the number of records for modifies doubles and could result in additional disk space requirements in the output files and additional processing time to write and read the records.

DEFAULT RECORDS PER BLOCK/AREA

DEFAULT RECORDS PER BLOCK	DEFAULT RECORDS PER BLOCK *nnn* Default value: 100 records per block. Use this option to indicate the number of records per block for the output files.
DEFAULT RECORDS PER AREA	DEFAULT RECORDS PER AREA *nnnn* Default value: 1000 records per area Use this option to indicate the number of records per area for the output files.

These values can be overridden for individual data sets. See "Individual Data Set Options" on page 129 for more information.

EMBEDDED EXTRACTS

Syntax: EMBEDDED EXTRACTS [TRUE | FALSE]

Use this parameter to specify that embedded data sets should be extracted even if INDEPENDENTTRANS is not set, or the parent data sets do not have valid AA values (such as COMPACT or ORDERED data sets).

NOTE: This parameter does not allow you to track embedded data sets.

ERRORSFATAL

Syntax: ERRORSFATAL [TRUE | FALSE]

Use this option to specify whether or not you want to terminate Span on the first error encountered, as described in the following table.

Value	Description
FALSE or NO	Default. Span continues processing when it encounters an error. If Span encounters a fatal error, however, it terminates immediately.
TRUE or YES	Span treats any error as a fatal error. In other words, Span reports the error message and then terminates processing immediately.

NOTE: Span sets its TASKVALUE to the last DBM error code. TASKVALUEs less than zero are errors or warnings defined by Span. By checking for the DBM error code in TASKVALUE in WFL/DATABRIDGE/SPAN, you can more easily determine the success or failure of a Span run.

EXTRACTS

Syntax: EXTRACTS [*directoryname* | *filename* [ON *devicename*]]

Leave this option blank if you want to use your entry for UPDATES as the default location. If you leave this option blank, EXTRACTS defaults to the destination for UPDATES. Otherwise, this location specifies where Span will write records extracted from the DMSII data sets during replication.

Directory To specify a directory location, enter the following:

```
EXTRACTS directoryname/=
```

Span creates an output file for each data set in the directory specified by *directoryname*. The title of each output data file is *directoryname/datasetname*. For example, the following entry:

```
EXTRACTS DATA/MYDB/= ON DATAPACK
```

causes the EXTRACTS output file for the CUSTOMER data set to be titled as follows:

```
DATA/MYDB/CUSTOMER ON DATAPACK
```

If the data set has variable formats, Span appends the record type number as the last node of the file title. For example, the following entry:

```
EXTRACTS = ON TAPE
```

ensures that each data set is written to a separate tape titled data set name. If the data set has variable formats, the tapes are labeled as follows:

```
datasetname/recordtype
```

Usercode You can also specify a usercode as part of the directory name or file title. If so, you must do one of the following:

- Run Span under that usercode
- Run Span under a privileged usercode

A usercode is not required for files output to TAPE.

File Title Instead of Directory Name

If you specify *filename* rather than *directoryname*, Span writes all extracted records to that one file, which is called a "consolidated file" on page 210. By default, it is a STREAM file containing variable-length records. This option is intended primarily for sites replicating the data onto a PC or UNIX database, where data files are stream-oriented and a line-feed character delimits each record. The formatting routine in the Support Library should append the data set number and line-feed character to each record as needed.

NOTE: To specify fixed-length records, uncomment the file equation for EXTRACTS in the Span WFL job and adjust the file attributes as needed.

For example, the following entry:

```
EXTRACTS      DATA/SPAN/COMBINED ON PACK
```

ensures that all extracted records are written to the following:

```
DATA/SPAN/COMBINED ON PACK
```

If you omit ON *packname*, the output defaults to the pack name for UPDATES. If you specify ON *packname*, FAMILY substitution rules apply.

FILTER

Syntax: `FILTER` *`filtername`*

where *filtername* is the name of a GenFormat filter routine that you have created. To create your own filtering routine, see “Creating a Filter” on page 76.

This option specifies the name of a filtering procedure, which is an entry point in the Support Library. Span calls the filtering procedure in the Support Library for each data set record. Your entry for “SUPPORT” on page 126 determines the library that Span calls.

FIXUPS

Syntax: `FIXUPS [` *`directoryname`* `|` *`filename`* `[ON` *`devicename`*`] ]`

This option specifies where Span writes the fixup records generated after the extraction phase of replication. Fixup records are any update records that are written to the audit file while extraction takes place.

NOTE: If you do an offline clone, there is no fixup phase.

---

directoryname or *filename* are treated the same as they are for the “EXTRACTS” on page 118.

Leave this option blank if you want to use your entry for UPDATES as the default location. If you leave this option blank, FIXUPS defaults to the destination for UPDATES.

FORMAT

Syntax: `FORMAT` *`formatname`*

This option specifies the name of the formatting procedure, which is the entry point in the Support Library. Span calls the format in the Support Library for each data set record. Your entry for “SUPPORT” on page 126 determines which library Span calls.

Valid values include the following:

formatname	**Description**
userwrittenformat	This is the name of a format routine that you have created. To create your own formatting routine, see Creating a Format. (page 66)
FIXEDFORMAT	This is a format in the default Support Library. This format converts a data set record into fixed width, readable (EBCDIC) fields. This is similar to a COBOL program moving each field to a PIC … DISPLAY. When you use FIXEDFORMAT (or a similar format), you might want to set the DATACHECK format option to FALSE for improved performance. See "FORMAT Options" on page 70.
COMMAFORMAT	This is a format in the default Support Library. This format delimits all records in the data set by commas.
RAWFORMAT	This is a format in the default Support Library. This is a binary format that DMSII uses to store data on the host. BINARYFORMAT This is a format in the default Support Library. This is a binary format that applies the filter and any ALTERs.

HEADER

`Syntax: HEADER [ TRUE | FALSE ]`

The HEADER option applies only to consolidated Span output files. A consolidated Span output file results when you specify a file title for the EXTRACTS, FIXUPS, and UPDATES options. In this case, the file specified by the file title contains all of the Span output. Compare this to specifying these options with *directory/=*, which creates several Span output files, each with the name of the data set in its title.

Use this option to specify whether or not you want Span to place a header at the beginning of a consolidated output file. The header contains the HOSTNAME as well as the complete starting and ending audit location in readable format. The HEADER option applies to normal updates, which occur when MODE = 2. The normal updates refer to updating a previously replicated database.

Using HEADER can facilitate secondary database processing by clarifying the order in which to process files. It also provides some continuity checking to ensure that files aren't lost or processed twice.

Value	Description
FALSE or NO	Default. No header is placed at the beginning of the consolidated output file.
TRUE or YES	Span writes a header record containing the complete audit locations corresponding to the start and end of the data file. Span inserts the following symbols in the header record: @ to designate the beginning of header data > to designate the starting audit location < to designate the ending audit location If Span discovers that the data file already exists, it ensures that the old ending audit location matches the current starting location. If it does not match, Span renames the file by adding a node (…/BAD) and opens a new file.

NONSTOP

Syntax: NONSTOP *nonstopoptions*

If you want to use the NONSTOP option, you must add it to the Span parameter file. By default, it does not appear in the file.

Ordinarily, Span stops processing when it receives a purge or reorganization notification from DBEngine, or an output file reaches its maximum file size. You can override this, however, with the NONSTOP option.

CAUTION: Span does not handle the problem of a change in record size due to a reorganization; therefore, use these options with care. If you anticipate record size changes, turn the NONSTOP REORG option off.

You can specify any combination of the following *nonstopoptions* after the literal NONSTOP.

nonstopoptions	Description
REORG	Span reselects the structure with values indicating that the problem has been resolved and continues processing even though DBEngine notifies it of a DMSII reorganization. Span will stop, however, in the case of a purge if NONSTOP PURGE has not been specified.
PURGE	Span reselects the structure with values indicating that the problem has been resolved and continues processing even though DBEngine notifies it of a DMSII purge. Span will stop, however, in the case of a reorganization if NONSTOP REORG has not been specified.

nonstopoptions	**Description**
FILE LIMIT	**CAUTION:** NONSTOP FILE LIMIT applies only when you are using Span to update a replicated database. If you use NONSTOP FILE LIMIT for the initial database clone, the data extraction results will be unpredictable. Span closes the output file and creates a new one when the current output file reaches the host file size limit of 15000 areas. When the output file reaches 15000 areas and new records still need to be added, Span stops processing unless you set the NONSTOP FILE LIMIT option. When NONSTOP FILE LIMIT is set, Span closes the output file and renames it as follows: *outputfilename/ending_ABSN* where *outputfilename* is determined by the entries explained in "File Title and Device Names" on page 115. Span then opens a new output file under the original name. If the full output file contains a partial transaction at the end of the file, Span will move the partial transaction to the new output file before closing the original file. Therefore, all output files will contain only complete transactions.

READER

Syntax: READER "*librarytitle*" using "*param*"

where *librarytitle* is the title of a Databridge FileXtract Reader library and *param* is the directory where the flat files you want to replicate are located. This parameter enables you to specify a Databridge FileXtract Reader library and the directory of the flat files you want to replicate, as in the following example:

```
READER "BANKFILE" USING "DATA/LOAD/SAMPLE/DATABASE"
```

REPORT

Syntax: REPORT *reportoptions*

The REPORT option controls the following:

- The types of information that Span writes to its report file. The types of information are controlled by the ERRORS, WARNINGS, and INFO options. Your selections are identified by the label "Logging" in the Span report file.
- The type of data errors that Span writes to its report file when the formatting routines detect a data error in a data set record. The types of data errors are controlled by the INVALID or NULL NUMBERS and TEXT options. Your selections are identified by the label "DataErrors" in the Span report file.

CAUTION: The formatting routines apply the data error options when you are using a formatting routine generated by GenFormat such as BINARYFORMAT, COMMAFORMAT or FIXEDFORMAT. RAWFORMAT routines do not use the data error options. User-written formatting routines do not use the data error options.

You can specify any combination of the following report options after the literal REPORT.

reportoptions	**Description**
ERRORS	Span writes all error messages to the report. ERRORS include any errors detected by the formatting routines generated by the GenFormat program.
WARNINGS	Span writes all warning messages to the report.
INFO	Span writes informational messages such as data file titles, number of records replicated, etc., to the report.
INVALID NUMBERS	Span checks for the following error conditions: ◆ Invalid signs ◆ Overflows ◆ Undigits
INVALID TEXT	Span writes all invalid text to the report. Invalid text is any text that has unreadable characters in an ALPHA data item.
NULL NUMBERS	**NOTE:** This option is provided for sites that want to ensure that none of their data items have a NULL value. For these sites, a NULL value would be invalid data. Other sites use the NULL value to indicate undefined data; in this case, the NULL value is not invalid data. Span writes all null numbers to the report. A null number is the value defined in the DASDL to be NULL for that numeric (NUMBER or REAL) data item. If the DASDL does not explicitly specify a NULL value for a data item, the NULL value is all bits turned on.
ALL	This is the equivalent of ERRORS, WARNINGS, and INFO. This is also the default, and it is in effect even when you comment out the REPORT option. The only way to turn off the REPORT option is to enter NONE, explained next.
NONE	Span still creates a printer backup file, but it contains only the date Span was initialized, the version, and the Span parameter filename.

The internal name of the Span report file is MSGFILE and it defaults to a printer backup file named as follows:

(*usercode*)DBBD/RUN/SPAN/*databasename*/*jobnumber*/*tasknumber*/000MSGFILE

For a sample report file, see "Span Report Files" on page 140.

You can file equate MSGFILE to some other device in WFL/DATABRIDGE/SPAN. The location of the printer backup file depends on the configuration at your site.

SOURCE

Syntax: SOURCE sourcename AT hostname VIA protocol PORT number

where *sourcename* is the name of a SOURCE in DBServer parameter file on the remote host, *hostname* is either the name of the remote host or if using TCP/IP, its TCP/IP address, *protocol* is either TCPIP, HLCN, or BNA, and *number* is the port number DBServer is using on the remote system.

Use the SOURCE parameter to specify that the audit files are on a remote host. This allows Span to run on a host that is less busy than the production host. DBEngine will communicate with DBServer on the remote host to retrieve portions of the audit trail.

The following is an example of a SOURCE parameter:

```
SOURCE BANKDB AT PROD VIA TCPIP PORT 3000
```

STOP

```
Syntax: STOP [ BEFORE | AFTER ] option
```

Use the STOP parameter to limit how much Span will replicate by specifying when it should stop processing audit information.

NOTE: Time in the STOP command refers to the time the update occurred on the primary system, not the current time of day. The "+/- days" are in relation to the Span Accessory start date. For example, when Span starts, it calculates an Audit file STOP date based on the current date plus or minus the "+/- days" parameter. Span stops when it reaches an Audit file with this calculated date.

The following table explains the STOP parameters:

STOP Parameter	Description
`STOP AFTER` *n* `AUDIT FILES`	Span will stop processing at the first quiet point after *n* audit files have been processed. Use this option to set the maximum number of audit files Span will read during a Span run. For example, the following command informs Span to stop processing after seven audit files: `STOP AFTER 7 AUDIT FILES`
`STOP AFTER` *n* `RECORDS`	Span will stop processing at the first quiet point after *n* records have been processed. Use this option to limit the number of records Span will write to its update files during a run. For example, the following command informs Span to stop processing at the first quiet point after 1000 records: `STOP AFTER 1000 RECORDS`
`STOP { BEFORE | AFTER }` *hh:mm* `[AM | PM] [ON +` *days*`]`	Span will stop processing at either the last quiet point before or the first quiet point after the time (*hh:mm*) and number of days (0 indicates current) specified. Use this option to set the earliest or latest time and date at which Span will stop processing. For example, the following command informs Span to stop processing at the last quiet point before 10:30 p.m.: `STOP BEFORE 10:30 PM`

STOP Parameter	Description
STOP { BEFORE | AFTER } "*taskname*"	Span will stop processing at either the last quiet point before the designated task started or the first quiet point after the designated task is completed. If you specify a usercode in taskname, Span looks for a task name match under the specified usercode. Otherwise, Span looks for task name match under any usercode. Use the STOP AFTER option to coordinate termination of Span with that of another, unrelated task. When Span finds a record in the audit trail that is a result of the specified task doing a CLOSE on the database, it stops at the next QPT. Use the STOP BEFORE option to ensure that Span update data represents the data available before the specified task started. When Span finds a record in the audit trail that is a result of the specified task doing an OPEN UPDATE on the database, it rolls back to the previous QPT, which results in stopping before the specified task. For example, the following command informs Span to stop processing at the first quiet point after the EOT of a task named OBJECT/SAVINGS/POSTING: `STOP AFTER "OBJECT/SAVINGS/POSTING"`
STOP { BEFORE | AFTER } "*taskname*" OR { BEFORE | AFTER } *hh:mm* [AM | PM] ON *MM/DD/YYYY*	Depending on which occurs first, Span will stop processing at either the last quiet point before the designated task started or the first quiet point after the designated task finished or at either the last quiet point before or the first quiet point after the designated time on the designated date. For example, the following command informs Span to stop processing at the first quiet point after the EOT of task name OBJECT/SAVINGS/POSTING on December 9, 2012 or to stop processing at the last quiet point before 10:30 p.m. on December 9, 2012, whichever occurs first: `STOP AFTER "OBJECT/SAVINGS/POSTING" OR BEFORE 10:30 PM ON 12/9/2012`

SUPPORT

Syntax: SUPPORT *title*

If you are using a tailored support library, type the name of the tailored support library for the database you are replicating. The names you enter for FORMAT and FILTER must be compiled in the tailored support library that you specify for SUPPORT. If you have not created a tailored support library yet, see Understanding DBGenFormat (page 46).

Otherwise, if you don't enter a SUPPORT title, it is automatically entered the first time you run Span. Span looks for Support libraries in this order:

OBJECT/DATABRIDGE/SUPPORT/*dbname*/*ldbname*
OBJECT/DATABRIDGE/SUPPORT/*dbname*
OBJECT/DATABRIDGE/SUPPORT

TAPE

If you specify ON TAPE, Span names the tapes depending on the format (directory/= or filename) that you enter. Tape names can be only two nodes long, so Span resolves them as follows:

directory/=

If the format is *directory/=*, Span names each tape as follows:

last_node_of_the_directory/datasetname

For example, if you enter EXTRACTS A/B/C/= ON TAPE, the resulting tapes would be named as follows:

C/*datasetname*

If the data set has variable formats, the tape is named for each variable format (except 0) as follows:

datasetname/variable_format_number

filename

If the format is *filename*, Span names each tape with the last two nodes of the file title, as follows:

second_to_last_node/last_node

For example, if you enter EXTRACTS D/E/F ON TAPE, the resulting, single tape would be named as follows:

E/F

TITLE

Syntax: `TITLE [ TIMESTAMP | AFN | ABSN ]`

This optional parameter followed by at least one field causes additional nodes to appear in the output file titles. If *TIMESTAMP* is used, the current date and time are appended to output file titles, in *YYYYMMDD/HHMMSS* format. If *AFN* or *ABSN* are used, the audit file number or audit block serial number is appended to the output file titles. Ordering is fixed when multiple *TITLE* fields are specified. For example, if all fields are specified, titles are appended with *TIMESTAMP* first, followed by *AFN* and then *ABSN*.

NOTE: The TITLE option is ignored for files created by clones.

TRANGROUP

Syntax: `TRANGROUP [ QPT | CHECKPOINT | NONE ]`

Using this parameter enables you to specify a different size for the transaction groups instead of using the normal transaction groups (Checkpoint) that is specified in the DBEngine parameter file (see "Checkpoint Frequency" on page 30 for more information).

Your entry for TRANGROUP can be one of the following:

Option	Description
QPT	This option causes Span to perform a commit at every quiet point.
CHECKPOINT	DBEngine will determine the size of transaction groups based on the CHECKPOINT option in the DBEngine parameter file. This is the default.
NONE	This option makes Span modeless and transactions ungrouped. DBEngine will send updates to Span as part of one long group and will never tell Span to abort any updates. Also, it will ignore any reorgs or purges it encounters. This feature is useful for simply reporting the records in the audit trail without regard for transaction groups or audit discontinuities. **NOTE:** This option, NONE, is intended for reporting purposes only; it is not intended for generating data to be loaded in another database or for production processing.

TRANSFORM

Syntax: TRANSFORM *transformname*

where *transformname* is the name of a GenFormat transform routine that you have created. To create a transform routine, see "Transforms" on page 83.

This option specifies the name of the transform routine, which is the entry point in the Support Library. Span calls the transform in the Support Library for each data set record. Your entry for "SUPPORT" on page 126 determines which library Span calls.

UPDATES

Syntax: UPDATES [*directoryname* | *filename* [ON *devicename*]]

This option specifies where Span writes the normal update records retrieved during tracking.

directoryname or *filename* are treated the same as they are for the "EXTRACTS" on page 118 option. The default name for UPDATES is as follows:

DATA/SPAN/*databasename/dataset*

If the UPDATES output files already exist, Span appends the new records to the existing files.

NOTE: Each time Span closes a disk UPDATES file, the USERINFO file attribute is set to the ending ABSN.

Tape Example

If you enter UPDATES ON TAPE, Span takes the default filename, which for UPDATES is DATA/SPAN/*databasename*/=.

However, since tape names can be only two nodes, the tape names would actually be *databasename/datasetname*. If a data set requires more than one tape (or cartridge), the tape name stays the same, but the MCP increments the reel number.

CAUTION: If you specify a directory name (resulting in one file per structure) ON TAPE, you must have one tape drive per structure, and they must all be opened at once.

Individual Data Set Options

Individual data set options for RECORDS PER ... and BYTES PER RECORD apply only when each data set is written to its own file (that is, you are not using a consolidated output file). The individual data set options override the settings for the following:

- DEFAULT MODIFIES
- DEFAULT RECORDS PER BLOCK
- DEFAULT RECORDS PER AREA

Individual data set options are listed in the Span parameter file in the following format:

```
% Dataset "datasetname"
%        Records per Area  nnnn
%        Records per Block  nnn
%        Bytes per Record     ?
%        After Images Only
%        Before and After Images
00004 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000
```

Overriding DEFAULT MODIFIES

Once you set an option for DEFAULT MODIFIES, you can override it for individual data sets by removing the comment character (%) in front of your selection.

The following example is from the BANKDB database.

```
% Dataset "BANKDB"
%        Records per Area  1000
%        Records per Block  100
%        Bytes per Record     ?
%        After Images Only
Before and After Images
00001 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000
```

Overriding DEFAULT RECORDS PER AREA

Syntax: RECORDS PER AREA *nnnn*

Default value: 1000 records per area.

This option determines the size of each area (in records) that will be written to the Span output files.

Overriding DEFAULT RECORDS PER BLOCK

Syntax: RECORDS PER BLOCK *nnn*

Default value: 100 records per block.

This option determines the size of each block (in records) that will be written to the Span output file.

Overriding DEFAULT RECORD SIZE

Syntax: BYTES PER RECORD

There is no default value for BYTES PER RECORD because the actual value depends on the format routine you select.

This option determines the size of each record (in bytes) that will be written to the Span output files. For Span to calculate the bytes per record automatically, leave the comment character (%) in front of this option.

If you do not specify a BYTES PER RECORD value, the record size is dependent on the format you select. For example, a COMMAFORMATQUOTEALL format will require more space than a FIXEDFORMAT.

If you do decide to set bytes per record, keep the following in mind:

If the bytes per record are set too large, you get all the data, but the drawback is the extra space required.

If the bytes per record are set too small, the data are truncated.

The size limit is the host record size limit.

Sample Span Parameter File

The following is a sample Span parameter file for a DMSII database titled BANKDB.

The Span parameter file is divided in two parts: the first part lists the Span parameters; the second part lists all of the data sets and remaps in the database being replicated.

```
%     Databridge Span parameter file for BANKDB
% Created Monday, November 30, 2015 @ 10:39:52 by Version 6.6.0.001

     % location of output data files
 Extracts      ON SPANDISK

 Fixups        ON SPANDISK

 Updates     (DB66)DATA/SPAN/BANKDB/= ON SPANDISK

%Title       TIMESTAMP     % put date and time nodes in titles
%Title        ABSN         % put starting ABSN in titles
%Title        AFN          % put starting AFN in titles

 Support     (DB66)OBJECT/DATABRIDGE/SUPPORT/BANKDB ON USER

%Transform    <transformname> % transform routine in DBSupport
 Format      BINARYFORMAT % formatting routine in DBSupport
 Filter      DBFILTER    % filtering routine in DBSupport
 Prefiltered  false  % false    => do not use filtered audits
                      % ONLY     => use only filtered audits
                     % PREFERRED => filtered audits preferred
%Reader      "librarytitle" using "param"  % FILEBridge
 TranGroup    Checkpoint % use Engine's Checkpoint value (default)
%TranGroup     QPT      % commit at every quiet point
```

```
%Source <sourcename> at <hostname> via TCPIP port 3000  % remote host
%Audit      "(<usercode>)<directory>"  % audit file name prefix
%Audit       on <packname>     % location of audit files
%Audit    on ORIGINAL FAMILY % audit files on original packs
%Audit       no wait         % don't wait on NO FILE for audit
%Audit      Job "<WFL job>" % job to zip when done with audit file

 Report      ERRORS WARNINGS INFO  % message reporting options
         %  INVALID Numbers Text  % report invalid data
         %  NULL Numbers Text    % report NULL data
 ErrorsFatal  true   % true  => if any error occurs, stop program
 Header      false  % true  => audit location in consolidated file
%Extract Embedded    % true => extract embedded w/o INDEPENDENTTRANS
%Clone       OFFLINE % OFFLINE => exclusive use of database
                % ONLINE => allow updates and database dump

%Stop At First QPT After n AUDIT FILES % max audit files processed
%Stop At First QPT After n RECORDS    % max records written
%Stop At First QPT After hh:mm on + 0  % lower bound time and date
%Stop At Last QPT Before hh:mm on + 0  % upper bound time and date
%Stop At First QPT After "taskname"   % after EOT
%          or Before hh:mm on mm/dd/yyyy % but by this time & date
%Stop At Last QPT Before "taskname"   % before BOT

 Default Records per Block 100  % default number of records per block
 Default Records per Area 1000  % default number of records per area
 Default Modifies After Images Only      % new record value only
%Default Modifies Before and After Images  % old record value first

%Data  Rec  File  Aud Block  Segment  Word  Date  Time  Mo- Format-Lvl
%-set Type  Nbr  SerialNbr  Number  Index YYYYMMDDhhmmss de DMS Client
%---- --- ---- --------- ------- ----- -------------- - ---- ----
 % Dataset "BANKDB"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
 00001 000  0003 0000000376  0000000  00000 00000000000000 0 00000 00000

 % Dataset "DBTWINCONTROL"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00002 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000

 % Dataset "RG"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00004 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000
```

```
 % Dataset "MASTER-DS"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00005 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "ORDERED-DS"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00006 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "EMB-STANDARD"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00008 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "RSDATA"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00011 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "BANK"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00013 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "BRANCH"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00016 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "MERGE-BRANCH"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
```

```
%00017 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "CUSTOMER"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00019 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "EMPLOYEES"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00020 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "DEPENDENTS"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00021 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "R-CUST"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00025 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "CUSTONELINE"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00026 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "CUSTNAMEONLY"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00027 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "TELLER"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
```

```
 %      Before and After Images   % modified records
%00030 000  0000 0000000000  0000000  00000 000000000000000 0 01751 01751

 % Dataset "ACCOUNT"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00032 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "ACCOUNT" (1)
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00032 001  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "ACCOUNT" (2)
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00032 002  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "ACCOUNT" (3)
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00032 003  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "ACCOUNT" (4)
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00032 004  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "ACCOUNT" (5)
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00032 005  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "R-ACCT"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
```

```
 %      After Images Only       % modified records
 %     Before and After Images   % modified records
%00035 000 0000 0000000000 0000000 00000 000000000000000 0 00000 00000

 % Dataset "R-ACCT" (3)
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %      After Images Only       % modified records
 %     Before and After Images   % modified records
%00035 003 0000 0000000000 0000000 00000 000000000000000 0 00000 00000

 % Dataset "R-ACCT" (1)
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %      After Images Only       % modified records
 %     Before and After Images   % modified records
%00035 001 0000 0000000000 0000000 00000 000000000000000 0 00000 00000

 % Dataset "R-ACCT" (2)
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %      After Images Only       % modified records
 %     Before and After Images   % modified records
%00035 002 0000 0000000000 0000000 00000 000000000000000 0 00000 00000

 % Dataset "R-ACCT" (4)
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %      After Images Only       % modified records
 %     Before and After Images   % modified records
%00035 004 0000 0000000000 0000000 00000 000000000000000 0 00000 00000

 % Dataset "R-ACCT" (5)
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %      After Images Only       % modified records
 %     Before and After Images   % modified records
%00035 005 0000 0000000000 0000000 00000 000000000000000 0 00000 00000

 % Dataset "SERIOUS-STUFF"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %      After Images Only       % modified records
 %     Before and After Images   % modified records
%00036 000 0000 0000000000 0000000 00000 000000000000000 0 00000 00000

 % Dataset "HISTORY"
 %      Records per Area  1000
 %      Records per Block  100
```

```
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00037 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "TRIALBALANCES"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00039 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "ADDRESSES"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00041 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "DIS-ORDERED"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00043 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "EMB-ORDERED"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00044 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "L1"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00047 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "L2"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00048 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "L3"
 %      Records per Area  1000
```

```
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00049 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000

 % Dataset "R-L2"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00051 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000

 % Dataset "L1-WITH-EMB"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00055 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000

 % Dataset "R-L1"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00056 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000

 % Dataset "L1-NO-EMBEDDED"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00057 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000

 % Dataset "SHORT-VF"
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00058 000  0000 0000000000  0000000  00000 00000000000000 0 00000 00000

 % Dataset "SHORT-VF" (2)
 %       Records per Area  1000
 %       Records per Block  100
 %       Bytes per Record    ?
 %      After Images Only        % modified records
 %      Before and After Images   % modified records
%00058 002  0000 0000000000  0000000  00000 00000000000000 0 00000 00000

 % Dataset "SHORT-VF" (5)
```

```
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00058 005  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "STATEMENT"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00060 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "NAMES"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00061 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000

 % Dataset "R-BRANCH"
 %      Records per Area  1000
 %      Records per Block  100
 %      Bytes per Record    ?
 %     After Images Only        % modified records
 %     Before and After Images   % modified records
%00063 000  0000 0000000000  0000000  00000 000000000000000 0 00000 00000
```

Replication Status

The replication status information follows the data set parameters (records per area, record per block, etc.) and is organized as follows:

The following table briefly describes the replication status information. Do not make changes to the replication status information unless you are instructed to do so.

Column	Description
Data Set Structure Number	The DMSII structure number of the data set or remap.
RecType	The record type for variable format data sets.
AFN ABSN SEG INX	The information in these columns represent the physical point in the audit file for the data set.

Column	Description
Date Time	The point in time of the data set in the audit file. **NOTE:** The first time you run Span, there is no audit file time and date. Therefore, the date and time column contains 00000000000000, which is the host zero date. After the data set is replicated, the date and time stamp indicate the position in the audit file.
Mode	Indicates the data set status, as follows: Mode = 0 The data set needs to be extracted. Mode = 1 The data set is in the fix up phase. Mode = 2 The data set is in the update phase. Mode = 3 The data set was reorganized. Mode = 4 The data set was purged.
DMS Format Level	This is the data set format level. Span puts the format level in this column during the clone and during DMSII reorganizations. **NOTE:** Zero is a valid format level. The format level will change when the DMSII data set is reorganized.
Client Format Level	This is the format level in the client database.

Do not make changes to the replication status information in the Span parameter file except in the following circumstances:

- Modify Mode when you do a purge or reorganization. A mode of 3 indicates a data set reorganization. A mode of 4 indicates data set purge (DMUTILITY initialize).
- After the reorganization on the client side, change the mode to 2 and then update the Client column (Format Level) to match the DMSII column (also under Format Level). The DMSII and Client columns are as follows:
  - The DMSII column is the DMSII update level (when the data set was added to or modified in the database definition). Each DASDL compile increments the level by one. For example, if you reorganized a data set the tenth time you update the DASDL, that data set has the number 0010 in the DMSII column. Note, however, that if the data set was ALTERed using GenFormat, the format level is recomputed and will have no obvious relationship with the DASDL compile number.
  - The Client column represents the update level of the data set in the Client database. Using the example of a data set reorganized on the tenth update, you would update the Client column to 0010 (the same as the DMSII level) after you reorganized that data set in the Client database.

Purges

When you purge a data set in the DMSII database, the value for the mode changes to 4. A mode of 4 indicates that the data set was purged; however, there is no DMS format level change. Instead, when you get a mode of 4, you must purge the corresponding table (data set) in the client database and then change the mode to 2.

Reorganizations

When you change the DMSII database layout (for example, adding a data item), two things happen, as follows:

- The DMSII format level corresponding to the changed data set is increased to the current database update level. The client format level, however, does not change. Instead, update the client database with the change and then increase the client format level to match the DMSII level.

  NOTE: The change to the data set may require you to do a reorganization on the client database.

- The mode changes to 3. Once you add the data item, for example, to the client database and increase the client format level, change the mode back to 2.

Span Report Files

Following are examples of Span output based on the Span parameter file described previously. The Span report files are named as follows:

DBBD/RUN/SPAN/*databasename/jobnumber/tasknumber/nnn*MSGFILE ON *packname*

First Printer Backup File

The following printer backup file is created when you run Span for the first time for a database.

```
Databridge SPAN Initializing month dd, yyyy @ hh:mm:ss on host hostname.
Version 6.6.x.x compiled date @ hh:mm:ss
Creating new parameter file DATA/SPAN/BANKDB/CONTROL
Databridge SPAN Terminating date @ hh:mm:ss
```

Report File

A printer backup file similar to the following is created whenever you run Span to replicate or track a database. After the list of option settings, the Span report file shows the mode and data set name followed by the output file name:

- To the left of the arrow is a description of the mode (for example, extracts or updates) and type of record.
- To the right of the arrow is the actual name of the file Span used to write these records. A + (plus sign) preceding the file name indicates that the file already existed and Span appended the new records to it.

The message "`Databridge Engine: >> [0007] Audit file 3 is unavailable; Current (active) audit file <<`" indicates why Span stopped processing. In this case, Span processed all of the available audit files and then stopped when it encountered the audit file the Accessroutines was still using.

The last part shows how many records Span wrote and skipped (due to the filter) for each data set.

```
Databridge SPAN Initializing month dd, yyyy @ hh:mm:ss on host hostname
Version 6.6.x.x compiled date @ hh:mm:ss
Reading parameter file (DB66)DATA/SPAN/BANKDB/CONTROL ON HOST
Option settings for database BANKDB:
Extracts  = (DB66)DATA/SPAN/BANKDB/= on DISK
Fixups    = (DB66)DATA/SPAN/BANKDB/= on DISK
Updates   = (DB66)DATA/SPAN/BANKDB/= on DISK
Support   = (DB66)OBJECT/DATABRIDGE/SUPPORT/BANKDB ON HOST
Transform = DBTRANSFORM
Format    = COMMAFORMAT
Filter    = DBFILTER
Errors    = Fatal
Audits    = Normal
TranGroup = Checkpoint
Limit     = <no limits>
Logging   = Errors Warnings Informational
DataErrors= <ignore>
Clone     = Online
Defaults:
Block   = 100 records
Area    = 1000 records
Modifies= After-images only
Extracts BANK                ===> (DB66)DATA/SPAN/BANKDB/BANK ON HOST
Extracts BRANCH              ===> (DB66)DATA/SPAN/BANKDB/BRANCH ON HOST
>> [0007] Active audit file 2 is unavailable <<
Replicated     Skipped
----------     -------
BANK                             7           0
BRANCH                          28           0
Total records:                  35           0
Thruput: 9 records/sec. CPU (excluding Extract Workers)
2 records/sec. elapsed
Final audit location: date @ time, AFN = 2, ABSN = 56
Databridge SPAN Terminating  date @ time
```

Span Commands

AX STATUS

To display status information about Span, enter the following:

mixnumber AX STATUS

where *mixnumber* is the mix number of Span. Once you enter this AX STATUS command, enter one of these:

- SMSG (from MARC)
- MSG (from the ODT)

Span responses will appear in the system messages.

The following is an example of Span in the process of extracting:

```
#9936 Databridge Engine: Extracting for CUSTOMER.
#9936 DBSpan: 908 records replicated (0 skipped).
#9936 DBSpan: Audit location: AFN=0 ABSN=0.
#9936 DBSpan: Audit time= January 9, 2010 @ 10:18:05.
```

The number of records skipped refers to records skipped because of a filter condition.

AX QUIT

If for any reason you don't want Span to complete, enter the following command:

mixnumber AX QUIT

where *mixnumber* is the mix number of OBJECT/DATABRIDGE/SPAN. The AX QUIT command causes Span to terminate at the next quiet point.

After you enter the AX QUIT command, Span displays the message Terminating at next quiet point. The number of records replicated and the number of records skipped is written to the Span report file (MSGFILE).

For example, if the Span mix number is 1234 and you want Span to terminate at the next quiet point, enter the following:

```
1234AX QUIT
```

AX HELP

To display the available Span AX commands, enter the following: *mixnumber* AX HELP

where *mixnumber* is the mix number of Span.

Automating Span Processing

This section explains automating Span after an audit file closes. For information on controlling when audit files close, see "AuditTimer Utility" on page 179.

When you use Span to replicate a database, you must determine how often and when you want it to gather updates to the DMSII database. You could run Span each time an audit file closes, or you could run it once per day.

Run Span at Audit File Close

The most convenient option is to run Span with the DMSII copy audit WFL that the Accessroutines initiate when an audit file closes. (The advantage of this method is that the audit file is readily available. The disadvantage is that you don't have much control over when it runs.) To run Span each time an audit file closes, you must configure and run the Copy Audit program. For instructions, see "Copy Audit Utility" on page 186.

Once Per Day Alternative

Another option is to run Span once per day (as part of batch processing) and have it process all audit files closed during that day. The advantage is that you have complete control over when Span runs. The disadvantage is that the audit files Span needs may have been already copied to tape and removed from disk. In this case, you would have to mount the audit tapes as Span requests them.

For instance, if you prefer to run Span only once a day during the nightly batch run, include the Span WFL that runs Span in the nightly batch job stream.

Transaction Groups Across Audit Files

Regardless when you choose to run Span, a transaction group might be started in one audit file and finished in the next. Since the end of an audit file is not necessarily a quiet point—transaction groups often start in one audit file and finish in another—Span may need a previous audit file so that it can find the (partial) transaction group that didn't finish until the next audit file.

For example, suppose Span processes audit file 104. At the end of audit file 104, it sees the start of a transaction group that is carried over into audit file 105. Since Span only returns completed transaction groups, it will not return this transaction group until audit 105 is available. At that time it will read the end of audit file 104 and continue on reading audit file 105.

Because of this, Databridge requires at least two audit files to remain on disk to accommodate transactions that start in one audit file but end in another. To ensure that Databridge has the audit files it needs, use the Copy Audit program (`WFL/DATABRIDGE/COPYAUDIT`). You can use the "Copy Audit Utility" on page 186 to keep a specified number of closed audit files on disk.

Span Tracking

Tracking is the process of extracting only completed changes from a DMSII audit file and then updating the secondary database with those changes. To accomplish the tracking of changes from the host DMSII database to a secondary database, Span uses the audit files created by DMSII. The audit files contain information about which records have been modified, deleted, or added.

You can determine how "up-to-date" your secondary database is by how often you close the DMSII audit files and run Span. For example, you can close the audit file every hour, every day, or even every week, depending on how crucial current information is. If you do not explicitly close the audit file, the Accessroutines close it when the audit file reaches the size specified in the DASDL. If you use the DBEngine parameter setting READ ACTIVE AUDIT = TRUE, Span can process the entire audit trail up to the last quiet point. Since a new audit file is opened as soon as an old one is closed, you can take the closed audit file and immediately begin the updating on the secondary database. Of prime importance is that the secondary database know its exact position in the audit trail.

For information on automating this process, see "Automating Span Processing" on page 142. For information on controlling when audit files close, see Databridge Utilities (page 177).

When updating the secondary database, any of the following issues can arise:

- Audit position
- Replicating additional data sets
- DMSII rollback or rebuilding
- DMSII reorganizations

Audit Position	Each secondary database must record the current audit position for each data set. (Databridge provides this information at the end of each transaction group.) If the secondary database is rolled back or reloaded from a dump, the audit position of the secondary database can be rolled back to an audit position prior to the database rollback or reload. Tracking then starts at the rolled back audit position so that the data records on the secondary database are in sync with the actual data records.
Replicating Additional Data Sets	If you want to replicate data sets that you did not include in previous runs of Span, you must uncomment the appropriate entry in the Span parameter file so that Span replicates the added structures as well.
DMSII Rollback or Rebuilding	Before you perform a rollback or reload, see Normal Recovery from a Rollback (page 202).
DMSII Reorganizations	Before you perform a reorganization, see DMSII Reorganizations (page 199).

Troubleshooting Span

This topic includes guidelines for troubleshooting any problems you may encounter with Span.

1 ***Read the Span report (MSGFILE).***

The report includes the following:

- The name of the parameters file that Span read
- The option settings that Span used
- The output file destinations
- The number of records replicated and the number of records skipped
- The reason Span stopped processing

See "REPORT" on page 123 and "Span Report Files" on page 140.

2 ***If only some data sets are updated, check the Mode column for each data set in the Span parameters file.***

- If the mode is 3, the DMSII data set was reorganized. You must change the mode to 2 and change the client format level to the DMS format level after you make the corresponding change in the client database.
- If the mode is 4, the data set was initialized using DMUTILITY. You must change the mode to 2 after you purge the corresponding rows in the client database.

Change the mode before you run Span again. For more details, see Complete a DMSII Reorganization (page 199).

3 ***If Span renames files by adding the node …/BAD, check your entry for the HEADER option in the Span parameters file.***

Most likely the HEADER option is TRUE, but you have existing files that do not have the correct header. This could be because the HEADER option was previously FALSE when those files were created.

4 ***If Span waits on a NO FILE for an audit file, it can't find the audit file (on disk or tape) it needs to continue processing. Do one of the following:***

- Mount the tape containing that audit file and let Span continue.
- DS the DATABRIDGE/MISSING_AUDIT_FILE task, which tells it that you cannot provide the audit file now. In this case Span stops processing immediately. The next time Span runs, it requests the same audit file.

If replication fails during the fixup phase, the fixup phase can be restarted. The Span receives COMMITs at the end of transaction groups during the fixup phase. You can restart the replication from the last COMMIT.

5 ***If Span finds duplicate data file names on the host, remove the data files as you move them to the client system for processing or do the following:***

5a Change the data file titles

```
CHANGE SPAN/DATA/databasename/datasetname TO READYTOMOVE/SPAN/DATA/
databasename/datasetname
```

5b Transfer the READYTOMOVE files to a client system or another location.

5c Remove the READYTOMOVE files.

6 ***Process Span anomalies.***

Since DBEngine does not have the opportunity to consolidate the changes that occurred during the extraction with the extracted records themselves, there is the possibility that during the fixup phase, Span will receive duplicate creates, deletes, or modifies for deleted records. Databridge does not flag these anomalies; therefore, you should be aware that these anomalies can occur.

6a When a duplicate record is encountered on a new record insert, replace the old record in the table with the new record to be created. For example, for duplicate creates, convert the second create to a modify.

6b Ignore missing records when a delete function is requested. For example, when the delete function fails, ignore the second delete.

6c When a modify function does not find a record in the client table, discard the modify.

If the client cannot handle the anomalies, you can use Snapshot to create a clean snapshot of the data sets.

7 Chapter 7: Snapshot

This chapter explains what the Databridge Snapshot Accessory is and how to use it.

In this Chapter

How Snapshot Works

The Databridge Snapshot Accessory takes an image of specified data sets in a DMSII database and places the images in host data files (to DISK or TAPE), as follows:

Note the following when using Snapshot:

- Each run of Snapshot lets you clone an unlimited number of data sets in a single DMSII database.
- Snapshot does not support links and will discard any link values it receives.
- To update the cloned database, you must reclone entire data sets. Alternatively, you can create a fixed-up extraction of the database using Snapshot and then use Span to do the updates. See "Chapter 6: Span" on page 105.

Snapshot WFL

You can modify the WFL/DATABRIDGE/SNAPSHOT file for the following:

- STARTTIME
- QUEUE—If, for example, you want Snapshot to enter the system through job queue 10, you must modify the WFL to include the QUEUE (or CLASS) = 10 declaration.
- BDNAME—This is the default file naming convention for the printer file. The name of the printer file that is generated by default is DBBD/RUN/SNAPSHOT/*dbname*.

Run the Snapshot Accessory

You must run Snapshot two times to clone a database, as follows:

- First, to create a parameter file for the database you specify
- Second, to actually perform the cloning

For subsequent runs of Snapshot, however, you can use the existing Snapshot parameter file for the database you are cloning. In other words, once you run Snapshot to create the parameter file, you do not need to create the parameter file again.

To replicate a DMSII database using Snapshot

1 From CANDE, start Snapshot by entering one of the following commands. You must include the quotation marks.

```
START WFL/DATABRIDGE/SNAPSHOT ("databasename" ["logicaldatabasename"])
```

Where	Is
"*databasename*"	The title of the DESCRIPTION file without the DESCRIPTION node. Quotation marks are required. The database name can include a usercode and pack, which are used to locate the database DESCRIPTION file, as follows: `"(usercode)databasename ON packname"` For example, `START WFL/DATABRIDGE/SNAPSHOT ("(PROD)INVENTORY ON DEVPACK")` creates a parameter file called `DATA/SNAPSHOT/INVENTORY/CONTROL` When you enter just a database name, Snapshot creates a parameter file named as follows: `(usercode)DATA/SNAPSHOT/databasename/CONTROL ON familyname`

Where	Is
"*logicaldatabasename*"	An optional name of a logical database when you want to do one of the following: ◆ Clone the data sets and data items in the logical database instead of in the original database. ◆ Create an alternate parameter file for the physical database, which is useful when you want to generate multiple support libraries for the same database but with different parameter files. When you enter a logical database name: ◆ Snapshot uses DATA/SNAPSHOT/*dbname*/*ldbname*/CONTROL as the title of the parameter file, if it exists ◆ Otherwise, Snapshot uses DATA/SNAPSHOT/*ldbname*/CONTROL as it did in previous releases, if it exists ◆ Otherwise, if neither file exists, Snapshot creates one titled DATA/SNAPSHOT/*dbname*/*ldbname*/CONTROL The logical database name provides a means for having multiple parameter files for the same database. If ldbname is the name of a logical database within the dbname database, Snapshot uses that logical database. If not, it uses *ldbname* to specify an alternate parameter file.

2 Use CANDE to edit the Snapshot parameter file to specify which data sets to clone and where to locate output files.

- Modify the first part (Snapshot options) using the descriptions of the options listed in "Snapshot Parameter File" on page 150.
- Modify the second part (each data set in the database) to reflect only the data sets you want to replicate. To do so, delete the comment sign (%) that precedes the name of the data set you want to clone.

3 Run Snapshot again using the same command you used in step 1:

This time, Snapshot uses the Snapshot parameter file, clones the DMSII data, and outputs it according to the options you set.

4 While Snapshot is working, you can display the number of records extracted and skipped as well as the current audit location and audit time by issuing an AX STATUS command. See "Snapshot Commands" on page 150 for instructions.

NOTE: If the following message appears, "`Databridge Engine: >> [0027] datasetname does not have a set with a unique key <<`" and there are components of the data set that can form a unique key, you can create a primary key. See "Creating a Primary Key" on page 84 for more information. Once you define the primary key, Snapshot will be able to clone the data set.

---

5 Once the Snapshot Accessory is finished, transport (download, copy to tapes, etc.) the DATA/SNAPSHOT/*databasename*/*datasetname* files or DATA/SNAPSHOT/*logicaldatabasename*/*datasetname* files to the client system and load the data into the target database.

Snapshot Commands

AX STATUS	To display the number of records extracted and skipped as well as the current audit location and audit time, enter the following: *mixnumber* AX STATUS where *mixnumber* is the mix number of the Snapshot Accessory. The following is an example of when Snapshot is in the process of cloning: 8989 20:24 DBSnapshot: 84437 records extracted (0 filtered out) The number of records filtered out are the result of a filter condition, if one exists.
AX QUIT	If for some reason you don't want Snapshot to complete, enter the following command: *mixnumber* AX QUIT where *mixnumber* is the mix number of OBJECT/DATABRIDGE/SNAPSHOT. The AX QUIT command causes Snapshot to terminate at the next quiet point. After you enter the AX QUIT command, Snapshot displays the message Terminating at next quiet point.
AX HELP	To display the available Snapshot AX commands, enter the following: *mixnumber* AX HELP where *mixnumber* is the mix number of Snapshot.

Snapshot Parameter File

Snapshot reads a parameter file to determine which data sets to clone, which formatting routine to use, etc.

Note the following for the format of the Snapshot parameter file:

- The options in the parameter file can be in any order.
- You can list multiple parameters on a single line.
- You can split parameters across multiple lines.
- There is no termination character.
- There is no continuation character.
- The comment character is the percent sign (%). The comment character can appear anywhere on a line and anything after the comment character is ignored.
- If you have a file name or family name that is the same as a parameter file keyword, enclose the file name or family name in quotation marks. For example, if you have a family named SUPPORT (which is also the name of an option in the Snapshot parameter file) and you want to enter that for the OUTPUT option, enclose SUPPORT in quotation marks, as follows:

  OUTPUT CUSTDATA/REPORTS ON "SUPPORT"

  This means that the default destination is a family called SUPPORT.
- All parameters are optional except for the list of data sets.

The remainder of this section describes the syntax and semantics of each parameter. Parameters are listed in the same order as they are listed in the parameter file.

OUTPUT

Syntax: OUTPUT *directoryname*
OUTPUT *directoryname* ON *familyname*
OUTPUT ON TAPE

Use the following table as a guide when you want to use the OUTPUT option.

Option	Description
OUTPUT *directoryname*	Write all the files to the specified directory. The default pack is DISK. Optionally, you can add a slash(/) or a slash with an equals sign (/=).
OUTPUT *directoryname* ON *familyname*	Write all files to the specified directory on the specified pack. The FAMILY substitution statement applies to the pack you designate.
OUTPUT ON TAPE	Write all output to tape.
OUTPUT NUL	Snapshot will not write any records as output, but it will perform the data validation checks. Use this option to generate a report of records having data errors but not actually write the data files containing the records.

SPAN ENTRIES

Syntax: SPAN ENTRIES *filename*

where *filename* is the name of the Span parameter patch file.

This parameter writes all audit location information that is compatible with Span to a single file, rather than creating separate patch files for each cloned data set. You can insert this patch file (by default, DATA/SNAPSHOT/*databasename*/SPANCONTROL) into the Span parameter file and use Span to continue replicating those data sets.

If SPAN ENTRIES is not specified, Snapshot writes the audit location information to individual patch files, one for each data set. You can insert one or more of these patch files into the Span parameter file for continued replication. This technique may be more suitable for situations where you run Snapshot for different groups of data sets because the audit location information from one group will not erase the information for another group.

TRANSFORM

Syntax: TRANSFORM *transformname*

where *transformname* is the name of a GenFormat transform routine that you have created. To create a transform routine, see “Transforms” on page 83.

This option specifies the name of a transform routine, which is an entry point in the Support Library. The Snapshot Accessory calls the transform in the Support Library for each data set record. Your entry for “SUPPORT” determines which tailored support library Snapshot calls.

READER

Syntax: READER ["*readerlibrary*"] USING "*readerparam*"

where *readerlibrary* is the title of a Databridge FileXtract Reader library and *readerparam* is the directory where the flat files you want to replicate are located.

This parameter enables you to specify a FileXtract Reader library and the directory of the flat files you want to replicate, as in the following example:

```
READER "OBJECT/FILEBRIDGE/READER/BANKFILE" USING "DATA/LOAD/SAMPLE/
DATABASE"
```

FORMAT

Syntax: FORMAT *formatname*

where *formatname* is the name of a GenFormat formatting routine that you have created. To create a format, see “Creating a Format” on page 66.

This option specifies the name of a formatting procedure, which is an entry point in the Support Library. Snapshot calls the format in the Support Library for each data set record. Your entry for “SUPPORT” determines which library Snapshot calls.

If you use FIXEDFORMAT (or a similar format), you might want to set the DATACHECK format option to FALSE for improved performance. See “FORMAT Options” on page 70

FILTER

Syntax: FILTER *filtername*

where *filtername* is the name of a GenFormat filtering routine that you have created. To create a filtering routine, see “Creating a Filter” on page 76.

This option specifies the name of a filtering procedure, which is an entry point in a tailored support library. Snapshot calls the filter in the tailored support library for each data set record. Your entry for SUPPORT determines which tailored support library Snapshot calls.

SUPPORT

Syntax: SUPPORT *title*

If you are using a tailored support library, type the name of the tailored support library for the database you are replicating. The transform name you enter for TRANSFORM, the format name you enter for FORMAT and the filter name you enter for FILTER must be compiled in the tailored support library you specify for SUPPORT. If you have not created a tailored support library yet, see Understanding DBGenFormat (page 46).

Otherwise, if you don’t enter a SUPPORT title, it is automatically entered the first time you run Snapshot. Snapshot looks for Support Libraries in this order:

```
OBJECT/DATABRIDGE/SUPPORT/dbname/ldbname
OBJECT/DATABRIDGE/SUPPORT/dbname
OBJECT/DATABRIDGE/SUPPORT
```

If a tailored support library for the database exists when Snapshot creates the parameter file, it will set the SUPPORT option to the title of that tailored library.

If the DMSII database update level changes, Snapshot will detect the update level mismatch and attempt to recompile the support library.

SORT

Syntax: `SORT DISKFACTOR n`
`SORT MEMORY nnnnnn`
`SORT DISKFACTOR n MEMORY nnnnnn`

Use the following table as a guide when you want to use the SORT parameter.

Option	Description
`SORT DISKFACTOR n`	where DISKFACTOR *n* is used to calculate the amount of temporary disk space for SORTing, as follows: `sortdiskspace:= diskfactor * filesize` In most cases, a DISKFACTOR of 2 is sufficient. Example: `SORT DISKFACTOR 2`
`SORT MEMORY nnnnnn`	where MEMORY *nnnnnn* is the amount of main memory in A Series words that SORT can use. Example: `SORT MEMORY 85000`

You can put both DISKFACTOR and MEMORY on the same line, as in the following example:

```
SORT DISKFACTOR 3 MEMORY 200000
```

STOP AFTER

Syntax: `STOP AFTER time`
`STOP AFTER time ON stopdate`
`STOP AFTER time ON + days`

Where	Is	
time	The audit file time in one of the following formats:	
	hh:mm	Hours and minutes in 24-hour format. For example, 13:15 for 1:15 PM
	hh:mm AM	Hours and minutes in clock time. For example, 10:00 AM
	hh:mm PM	Hours and minutes in clock time. For example, 10:00 PM
ON *stopdate*	A date in the following format:	
	mm/dd/yyyy	Month, day, and four-digit year. For example, STOP AFTER 8:15 AM ON 12/9/2012
ON + *days*	Any integer greater than zero, which represents the number of days after Snapshot started running. For example, `STOP AFTER 9:30 AM ON +1` means that Snapshot should try to make its data files match the way the DMSII database will look at 9:30 AM on the day after Snapshot started running.	

The STOP AFTER parameter provides an "effective time" for Snapshot to stop requesting updates. In other words, Snapshot requests updates until it encounters the QPT (quiet point) after the time you specify. By default, STOP AFTER does not appear in the parameter file. If you want to use it, you must type it in the Snapshot parameter file.

NOTE: This parameter only works with online clones. Since an offline clone prevents any updating, there can be no audit to read to bring the database to a particular time. If you configure the STOP AFTER parameter and specify an offline clone, Snapshot ignores the STOP AFTER parameter and reports the stop point as the following:

```
Stop Point  = (ignored for offline clone)
```

Time in the STOP command refers to the time the update occurred on the primary system, not the current time of day.

The "+ days" is in relation to the Snapshot start date. For example, when Snapshot starts, it calculates an audit location STOP date based on the current date plus the "+ days" parameter. Snapshot stops when it reaches an audit location from the primary system with this calculated date.

Example:

If you enter the following:

```
STOP AFTER 11:30 PM
```

Snapshot will request updates until it sees the QPT after 11:30 PM, even if Snapshot starts at 10:05 PM and completes the extract at 10:45 PM. The data in the Snapshot data files will match the database as of 11:30 PM.

REPORT

Syntax: REPORT *reportoptions*

When the formatting routines detect a data error, Snapshot writes a description of the error to a file. The REPORT option controls the types of data errors that Snapshot writes to its report file during its run. Your selections are identified by the label “Data errors” in the Snapshot report file. For more information about the Snapshot report file, see “Snapshot Report File” on page 160.

IMPORTANT: The formatting routines apply the data error options when you are using a TEXT or BINARY formatting routine generated by GenFormat such as COMMAFORMAT or FIXEDFORMAT.

RAW format routines do not use the data error options.

User-written formatting routines do not use the data error options.

You can specify any combination of the following *reportoptions* after the literal REPORT.

reportoptions	**Description**
INVALID NUMBERS	Snapshot checks for the following error conditions: ◆ Invalid signs ◆ Overflows ◆ Undigits
INVALID TEXT	Snapshot writes all invalid text to the report. Invalid text is any text that has unreadable characters in an ALPHA data item.
NULL NUMBERS	**NOTE:** This option is provided for sites that want to ensure that none of their data items have a NULL value. For these sites, a NULL value would be invalid data. Other sites use the NULL value to indicate undefined data; in this case, the NULL value is *not* invalid data. Snapshot writes all null numbers to the report. A null number is the value defined in the DASDL to be NULL for that numeric (NUMBER or REAL) data item. If the DASDL does not explicitly specify a NULL value for a data item, the NULL value is all bits turned on.
NULL TEXT	**NOTE:** This option is provided for sites that want to ensure that none of their data items have a NULL value. For these sites, a NULL value would be invalid data. Other sites use the NULL value to indicate undefined data; in this case, the NULL value is *not* invalid data. Snapshot writes all null text to the report. Null text is the value defined in the DASDL to be NULL for that ALPHA data item. If the DASDL does not explicitly specify a NULL value for a data item, the NULL value is all bits turned on.

The internal name of the Snapshot report file is MSGFILE and it defaults to a printer backup file named as follows:

(*usercode*)DBBD/RUN/SNAPSHOT/*databasename/job number/tasknumber*/000MSGFILE

You can file equate MSGFILE to some other device in WFL/DATABRIDGE/SNAPSHOT. The location of the printer backup file depends on the configuration at your site.

WORKERS

Syntax: `WORKERS` *`number`*

where *number* is a number from 1 to 10. The default is 1.

The WORKERS parameter causes Snapshot to request a certain number of Extract Workers to perform the data set extracts during a clone. This may increase the speed of the cloning process. When there is more than one Worker running, Snapshot receives records interspersed from multiple data sets.

Note that the actual number of Workers running is the smallest of the following values:

- Snapshot Workers limit
- DBEngine Extract Workers limit
- Number of data sets to extract

EMBEDDED EXTRACTS

Syntax: `EXTRACT EMBEDDED [ [ = ] TRUE | FALSE ]`

Enabling Extract Embedded allows Snapshot to extract embedded data sets even if INDEPENDENTRANS is not set or the parent data sets do not have valid AA values (such as COMPACT or ORDERED data sets).

NOTE: The Embedded Extracts parameter only extracts embedded data sets. It does not perform any type of fixup to the data sets.

CLONE

Syntax: `CLONE [ OFFLINE | ONLINE ]`

Use this parameter to specify whether Databridge should do an online extraction (the database is open for updates) or an offline extraction (the database is not open for updates, or least programs cannot update the data sets to be cloned). If you do not specify OFFLINE or ONLINE, Snapshot will do an online clone. You might want to clone offline to avoid having Snapshot sort and apply any fixup records to the extract files. Note that if the database has INDEPENDENTTRANS set, DBEngine will use SECURE STRUCTURE to enforce an offline clone, which allows other programs to update the data sets not being cloned. Otherwise, the database is opened in exclusive mode.

NOTE: If you specify OFFLINE, there will not be a fixup phase in the replication process since there will be no updates to any of the extracted data sets.

ALL

To clone every supported data set in the database, type ALL in the Snapshot parameter file. By default, ALL does not appear in the parameter file.

Data Set List

To clone a portion of the database, specify a list of data sets. In this case, you can override the destination set by OUTPUT on a data set-by-data set basis. Use the following format for each item in the data set list:

datasetname [(*recordtype*)] [ON *devicename*]

Where	Is
datasetname	Required. The name of the DMSII data set.
recordtype	Optional. The record type number of the variable format data set (if applicable). You must enclose the record type in (parentheses), as in the following example: ACCOUNT (2)
devicename	Optional. The output device for this data set, as in the following example: CUSTOMER ON DATAPACK **NOTE:** The value for *devicename* overrides the default for the OUTPUT parameter for this data set only.

See "Sample Snapshot Parameter File" on page 157 for an example of the data set list.

All data sets are cloned to the location you specify via the OUTPUT parameter unless the ON *devicename* option is specified for the data set.

Sample Snapshot Parameter File

Following is a sample Snapshot parameter file. This parameter file specifies a list of data sets instead of the ALL option.

```
          % DataBridge Snapshot Parameter file for BANKDB
% Created Friday, June 13, 2014 @ 11:42:25 by Version 6.6.0.001

%
%Syntax:
%
%  [OUTPUT directoryname [ON familyname]]
%       where 'directoryname' is a DB directory name that
%       will contain the output files, for example, DATA/MYDB.
%        (A trailing / or /= is optional.)
%       The 'familyname' is the pack where the files will be
%       created. FAMILY substitution does apply.
%
%        To write to tapes, use OUTPUT ON TAPE.
%
%  [SPAN ENTRIES title]
%       where 'title' is the title of the file that will
%      contain SPAN-compatible audit location entries. You can
%       insert this file in DATA/SPAN/<database>/CONTROL.
%
```

```
%  [TRANSFORM entrypoint]
%       where 'entrypoint' is a DBTransform-type procedure in
%        the Support Library that will be called for each
%         dataset record.
%
%  [READER ["readerlibrary"] [USING "readerparam"]]
%       where 'readerlibrary' is the title of a FILEBridge
%       Reader Library and 'readerparam' is its parameter.
%
%  [FORMAT entrypoint]
%       where 'entrypoint' is a DBFORMAT-type procedure in the
%       Support Library that will be called to format each
%         dataset record.
%
%  [FILTER entrypoint]
%       where 'entrypoint' is a DBFILTER-type procedure in the
%       Support Library that will be called to filter each
%         dataset record.
%
%  [SUPPORT title]
%       where 'title' is the title of the Support Library
%        containing the FORMAT routine.
%
%  [SORT DISKFACTOR factor]
%       where 'factor' determines the DB size for SORT
%       temporary files according to the following formula:
%           sorttempsize = factor * inputfilesize.
%         Factor is usually 2.
%
%  [SORT MEMORY words]
%       where 'words' is the amount of SORT memory.
%
%  [STOP AFTER time [ON stopdate]]
%       where 'time' is hh:mm, hh:mm AM, or hh:mm PM
%       and 'stopdate' is mm/dd/yy, mm/dd/yyyy, or +days.
%        Example: 10:30 PM ON +1
%                 23:15 ON 12/31/96
%
%  [REPORT {INVALID or NULL} {NUMBERS and/or TEXT}]
%        reports invalid and/or null data items.
%        Example: REPORT INVALID NUMBERS
%                 REPORT NULL NUMBERS TEXT
%
%  [WORKERS [=] limit ]
%       where 'limit' is the maximum number of Workers to run
%
%  [EXTRACT EMBEDDED [ [=] { TRUE or FALSE} ]
%       If INDEPENDENTTRANS is reset and EXTRACT EMBEDDED is
%       true, then extract (but don't fix up) embedded dataset
%         records.
%
%  [CLONE [ [=] { OFFLINE or ONLINE } ]
%       An OFFLINE CLONE requires exclusive use of the database
%       but eliminates the need for any fixup records. ONLINE
%       clones permit concurrent updates and OFFLINE dumps.
```

```
%
%  {ALL or datasetlist}
%       where 'datasetlist' is one or more records of the form:
%
%  dataset [ (rectype) ] [ON devicename]
%       where 'dataset' is the name of the DMSII dataset,
%
%       'rectype' is the record type number of the variable
%        format dataset,
%
%       'devicename' is the output device for this dataset,
%        either a familyname or TAPE.
%
  % Database: BANKDB

   % destination of Snapshot data files .../<dataset>
    Output DATA/SNAPSHOT/BANKDB/= ON DB

    Span Entries DATA/SNAPSHOT/BANKDB/SPANCONTROL ON DB

    Support (DB66)OBJECT/DATABRIDGE/SUPPORT ON USER

    Transform DBTransform    % transform in Support Lib
    Format   SNAPSHOTCOMMA  % formatting routine in Support Lib
    Filter   DBFILTER      % filtering routine in Support Lib
     Sort     DiskFactor 2 Memory 100000
     % Report  Invalid numbers text
     % Report  Null   numbers text
     % Extract Embedded
     % Clone   Offline  % prevents updates
     % Workers 4

% remove the '%' in front of the datasets you want

  % BANKDB
  % RSDATA
    BANK
    BRANCH
  % MERGE-BRANCH
  % R-BRANCH
    CUSTOMER
    EMPLOYEES
  % DEPENDENTS
  % R-CUST
  % CUSTONELINE
  % CUSTNAMEONLY
    TELLER
    ACCOUNT
    ACCOUNT (1)
    ACCOUNT (2)
    ACCOUNT (3)
  % ACCOUNT (4)
  % ACCOUNT (5)
  % R-ACCT
  % R-ACCT (3)
```

```
% HISTORY
% TRIALBALANCES
% ADDRESSES
% DIS-ORDERED
% EMB-ORDERED
% L1
% L2
% L3
% R-L2
% L1-WITH-EMB
% L1-NO-EMBEDDED
% SHORT-VF
% SHORT-VF (2)
% SHORT-VF (5)
% DBTWINCONTROL
% RG
% SERIOUS-STUFF
% R-L1
```

Snapshot Report File

Running Snapshot creates the report file. You can control the types of data errors reported in the report file by your selections for the REPORT parameter. For more information, see “REPORT” on page 155.

The following is an example of a Snapshot report file a sample DMSII database called BANKDB. In this report, no data errors were specified.

Report File with No Data Errors

Following is an example of a Snapshot report file for a sample DMSII database called BANKDB. In this report, no data errors were specified.

```
Databridge Snapshot Initializing date @ hh:mm:ss on host hostname
Version 6.6.x.x compiled date @ hh:mm:ss
Reading parameter file DATA/SNAPSHOT/BANKDB/CONTROL
Support     = (DB66)OBJECT/DATABRIDGE/SUPPORT ON HOST
Filter      = DBFILTER
Format      = SNAPSHOTCOMMA
Sort Memory = 100000 words
Sort Disk   = 2 * file size
Output      = DATA/SNAPSHOT/BANKDB/= ON DISK
Span File   = DATA/SNAPSHOT/BANKDB/SPANCONTROL ON DISK
Stop point  = (unspecified)
Data errors = <ignore>
Embedded Extracts = False
BANKDB       ===> DATA/SNAPSHOT/BANKDB/BANKDB (1 records)
BANK         ===> DATA/SNAPSHOT/BANKDB/BANK (7 records)
BRANCH       ===> DATA/SNAPSHOT/BANKDB/BRANCH (7 records)
MERGE-BRANCH ===> DATA/SNAPSHOT/BANKDB/MERGE-BRANCH (7 records)
CUSTOMER     ===> DATA/SNAPSHOT/BANKDB/CUSTOMER (44 records)
R-CUST       ===> DATA/SNAPSHOT/BANKDB/R-CUST (44 records)
CUSTONELINE  ===> DATA/SNAPSHOT/BANKDB/CUSTONELINE (44 records)
TELLER       ===> DATA/SNAPSHOT/BANKDB/TELLER (9 records)
ACCOUNT      ===> DATA/SNAPSHOT/BANKDB/ACCOUNT/2 (88 records)
Effective time of snapshot: date @ hh:mm:ss
```

Report File with Data Errors

Following is an example of a Snapshot report file for the same sample DMSII database (BANKDB). In this report, data errors were specified.

```
Databridge Snapshot Initializing date @ hh:mm:ss on host hostname
Version 6.6.x.x compiled date @ hh:mm:ss
Reading parameter file DATA/SNAPSHOT/BANKDB/CONTROL
Support     = (UCC)OBJECT/DATABRIDGE/SUPPORT ON HOST
Filter      = DBFILTER
Format      = SNAPSHOTCOMMA
Sort Memory = 100000 words
Sort Disk   = 2 * file size
Output      = DATA/SNAPSHOT/BANKDB/= ON DISK
Span File   = DATA/SNAPSHOT/BANKDB/SPANCONTROL ON DISK
Stop point  = (unspecified)
Data errors = Bad number Bad text NULL number NULL text
Embedded Extracts = False
#0002  record at segment 000000E, word 0000 (UID = 000000000420)
Unreadable RDS-ID = 4"0000000000000000000"
#0002  record at segment 000000E, word 0003 (UID = 000000000423)
Unreadable RDS-ID = 4"0000000000000000000"
#0007  record at segment 0000046, word 00AF (UID = 000000002275)
XYZCO
Null FAX = 4"FFFFFFFFFFFFFFFFFFFFFFFFFFFF"
#0007  record at segment 0000046, word 00F5 (UID = 000000002345)
BXBBB
```

```
Null PHONE = 4"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
#0007  record at segment 0000046, word 00F5 (UID = 000000002345)
BXBBB
Null FAX = 4"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF"
BANKDB        ===> DATA/SNAPSHOT/BANKDB/BANKDB (1 records)
BANK          ===> DATA/SNAPSHOT/BANKDB/BANK (7 records)
BRANCH        ===> DATA/SNAPSHOT/BANKDB/BRANCH (7 records)
MERGE-BRANCH  ===> DATA/SNAPSHOT/BANKDB/MERGE-BRANCH (7 records)
CUSTOMER      ===> DATA/SNAPSHOT/BANKDB/CUSTOMER (44 records)
R-CUST        ===> DATA/SNAPSHOT/BANKDB/R-CUST (44 records)
CUSTONELINE   ===> DATA/SNAPSHOT/BANKDB/CUSTONELINE (44 records)
TELLER        ===> DATA/SNAPSHOT/BANKDB/TELLER (9 records)
ACCOUNT       ===> DATA/SNAPSHOT/BANKDB/ACCOUNT/2 (88 records)
Effective time of snapshot: date @ hh:mm:ss
```

Snapshot File Outputs

Snapshot creates two output files for each data set you select to clone—one data file and one parameter patch file (unless you enable the SPAN ENTRIES parameter, in which case all data set information is written to one patch file. See “SPAN ENTRIES” on page 151 for more information.).

Data Files

Snapshot creates one data file for each data set, with default titles as follows:

DATA/SNAPSHOT/*databasename/datasetname*

— or —

DATA/SNAPSHOT/*logicaldatabasename/datasetname*

These files contain all of the records in the database as of the time of the “snapshot.” The actual data in the files is formatted according to your entry for FORMAT (for example, COMMAFORMAT, FIXEDFORMAT, or RAWFORMAT).

If the data set has variable formats, each record format type has its own file. These file names have an additional node, as follows:

DATA/SNAPSHOT/*databasename/datasetname/recordtypenumber*

— or —

DATA/SNAPSHOT/*logicaldatabasename/datasetname/recordtypenumber*

In addition, the NOTE file attribute of each output data file contains the ending audit location.

Snapshot writes state information in a separate file for each data set, as explained next.

CONTROL File

All parameter patch (CONTROL) files are titled the same as the data files, but with the CONTROL node appended to the file title, as in the following example:

DATA/SNAPSHOT/*databasename/datasetname*/CONTROL

— or —

DATA/SNAPSHOT/*logicaldatabasename*/*datasetname*/CONTROL

This control file contains STATEINFO data that reflects the audit location, format level, etc., of the cloned data. The STATEINFO data is passed to DBEngine.

The STATEINFO data is in the same format as the STATEINFO that appears in the Databridge Span parameter file for each data set. The layout of the STATEINFO data is described in the file SYMBOL/DATABRIDGE/INTERFACE.

An advantage to having the STATEINFO data in a separate file is when you use Snapshot to clone a database and then later decide you want to update the cloned database instead of creating it again. In this case, insert the contents of the individual *datafilename*/CONTROL of the data sets you want to track into the Databridge Span parameter file. For each data set for which control information is placed in the Databridge Span parameter file, include a comment that identifies the data set.

NOTE: This STATEINFO data should be also inserted in the secondary database parameters information. This is important for the secondary database to keep track of the location in the audit file of the last update.

---

8 Chapter 8: Lister

This chapter explains how you can use the Databridge Lister Accessory to create reports about your DMSII database. For information about additional reports that you can generate, see "DBInfo Utility" on page 189.

In this Chapter

How Lister Works

Lister reads a DMSII DESCRIPTION file and a parameter file to produce a report describing the data definition of a DMSII database (database layout). This report also lists the structure numbers of the data sets you want to replicate. These structure numbers appear in the Span parameter file.

The Lister report lists all the data sets in the DMSII database. Note that if you are using a tailored support library, data sets excluded by a filter will not appear on the report.

In addition to the data set structure number, the report shows information for formatting routines, such as offsets and sizes, and the name of the Support Library used to generate the report. By default, the report is written to a printer file, but you can file equate the output device by modifying the WFL. For example, to write the report to a data file, equate the FILE LINE as follows:

```
FILE LINE (KIND=DISK, TITLE=datafilename,
PROTECTION=SAVE)
```

where *datafilename* is the name of the data file to which the report will be written.

Run the Lister Accessory

Use the following procedure for running Lister to produce a database layout report. You can run Lister at any time.

To run Lister

1. Using CANDE or another editor, open the sample Lister parameter file (`DATA/LISTER/SAMPLE/CONTROL`).
2. Save the sample parameter file as follows:

   `DATA/LISTER/databasename/CONTROL`

where *databasename* is the name of the database or logical database from which you plan to create the report.

3 Edit the new Lister parameter file (DATA/LISTER/*databasename*/CONTROL), and set report options. See "Lister Parameter File" on page 167 for an explanation of each option.

4 Start Lister using the following syntax.

NOTE: For a description of the Lister WFL runtime options, see "Lister WFL" on page 166.

```
START WFL/DATABRIDGE/LISTER ("databasename")
```

— or —

```
START WFL/DATABRIDGE/LISTER ("databasename", "logicaldatabasename")
```

The *logicaldatabasename* parameter provides a means for having multiple parameter files for the same database. If *logicaldatabasename* is the name of a logical database within the *databasename* database, Lister uses that logical database. Otherwise, it simply uses the *logicaldatabasename* as a way to specify an alternate parameter file (DATA/LISTER/*databasename*/*logicaldatabasename*/CONTROL). Lister then creates a report, which it directs to a printer file.

5 Print or view the report generated by Lister. By default it is named as follows:

```
DBBD/RUN/LISTER/databasename
```

— or —

```
DBBD/RUN/LISTER/logicaldatabasename
```

For an example and explanation of this report, see "Lister Report" on page 171.

Example:

The following command:

```
START WFL/DATABRIDGE/LISTER ("(PROD)INVENTORY ON DEVPACK")
```

indicates that Lister should use the following file:

```
(PROD)DESCRIPTION/INVENTORY ON DEVPACK
```

The Lister printer file will be named as follows:

```
DBBD/RUN/LISTER/INVENTORY/=
```

Lister WFL

You can modify the Lister WFL for the following:

- STARTTIME
- QUEUE—If you want this job to enter the system through job queue 10, you must modify the WFL to include the QUEUE (or CLASS) = 10 declaration.
- BDNAME—This is the default file naming convention for the printer file.

Lister Parameter File

Lister reads the Lister CONTROL parameter file to determine the amount and type of information on the Lister report.

Note the following for the format of the Lister parameter file:

- The parameters in the parameter file can be in any order.
- You can list multiple parameters on a single line.
- You can split parameters across multiple lines.
- There is no termination character.
- There is no continuation character.
- The comment character is the percent sign (%). The comment character can appear anywhere on a line and anything after the comment character is ignored.

The remainder of this section describes the syntax and semantics of each parameter. Enter the parameters in the Lister parameter file, following the commented section that describes each parameter.

NAMES ONLY

Syntax: `NAMES ONLY [ = ] [ TRUE | FALSE ]`

Default: FALSE

NAMES ONLY lists only the names of the DMSII database data sets. Note, however, that you can use this option in combination with any other Lister option except MAX RECORDS. (The MAX RECORDS option is ignored if the NAMES ONLY option is set to TRUE.) For example, if you set NAMES ONLY to TRUE and set SETS to TRUE, you will create a report that contains a list of the Database DMSII data sets names and the sets associated with each data set.

PARENT

Syntax: `PARENT [ = ] [ TRUE | FALSE ]`

Default: TRUE

PARENT shows the name and number of the parent data set if the data set is embedded.

MAX RECORDS

Syntax: `MAX RECORDS [ = ] [ TRUE | FALSE ]`

Default: FALSE

MAX RECORDS shows the upper bound estimate of the number of records in a data set.

NOTE: The MAX RECORDS option is ignored if the NAMES ONLY option is set to TRUE.

SETS

Syntax: SETS [=] [TRUE | FALSE]

Default: TRUE

SETS lists the sets associated with each data set.

SUBSETS

Syntax: SUBSETS [=] [TRUE | FALSE]

Default: FALSE

SUBSETS lists the subsets associated with each data set.

KEYS

Syntax: KEYS [=] [TRUE | FALSE]

Default: TRUE

KEYS lists the data items that compose the keys of each set and also list any KEY DATA items declared in the set. The KEY DATA items will appear below the keys list, introduced with the literal “Data:”.

NOTE: The KEYS option is ignored if the SETS option is set to FALSE.

DATAITEMS

Syntax: DATAITEMS [=] [TRUE | FALSE]

Default: TRUE

DATAITEMS lists the data items contained in each data set.

NULL

Syntax: NULL [=] [TRUE | FALSE]

Default: FALSE

NULL shows the NULL value for each data item.

COMMENTS

Syntax: COMMENTS [=] [TRUE | FALSE]

Default: FALSE

If the COMMENTS option is TRUE, Lister will report DASDL- specified comments for structures and data items on the next line with a leading %.

FILTER

Syntax: FILTER *filtername*

where *filtername* is the name of any filter generated by GenFormat.

This option specifies the name of a filtering procedure, which is an entry point in a tailored Support Library. Lister calls the tailored Support Library to filter each data set specified in the filter. Data sets excluded by a filter will not appear on the Lister report.

Your entry for SUPPORT, explained next, determines which library Lister calls.

SUPPORT

Syntax: SUPPORT "*libraryname*"

Type the name of the tailored Support Library for the database for which you want the Lister report. The filter name you enter for FILTER must be compiled in the tailored Support Library you specify for SUPPORT. If you have not created a tailored Support Library yet, see "Chapter 4: Support Library" on page 43.

If you are using the default Support Library (OBJECT/DATABRIDGE/SUPPORT), leave this field as is.

DATASET

Syntax: DATASET *datasetname*

Default: List every data set

DATASET names the data set for which you want to run Lister. For Lister to report on more than one data set, enter DATASET *datasetname* as often as you want. For Lister to report on all data sets, make sure that DATASET is commented out.

Sample Lister Parameter File

```
%--------------------------------------------------------------------
%
% (C) Copyright 2019 Micro Focus or one of its affiliates.
%
% Module: DATA/LISTER/SAMPLE/CONTROL
%
% Project: Databridge
%
% Description: Databridge Lister Sample Parameter File
%
% (C) Copyright 2019 Micro Focus or one of its affiliates.
%
%--------------------------------------------------------------------
%
% Syntax: (Options may appear in any order.)
%
%       NAMES ONLY [=] [TRUE/FALSE]
%              structure names only--no structure numbers, levels,etc.
```

```
%               This option sets the SETS, KEYS, DATAITEMS, and
%               PARENT options to FALSE.
%               default: FALSE
%
%       PARENT [=] [TRUE/FALSE]
%               show structure number and name of parent dataset
%               for embedded datasets.
%               default: TRUE
%
%       MAX RECORDS [=] [TRUE/FALSE]
%               show estimated maximum number of records in datasets;
%               default: FALSE
%
%       SETS [=] [TRUE/FALSE]
%               list sets associated with datasets;
%               default: TRUE
%
%       SUBSETS [=] [TRUE/FALSE]
%               list subsets associated with datasets;
%               default: FALSE
%
%       KEYS [=] [TRUE/FALSE]
%               list key items and KEY DATA items of sets and subsets;
%               ignored if SETS = FALSE;
%               default: TRUE
%
%       DATAITEMS [=] [TRUE/FALSE]
%               list data items for each dataset; ignored if DATASETS =
%               FALSE;
%               default: TRUE
%
%       NULL [=] [TRUE/FALSE]
%               show NULL value for each data item; ignored if
%               DATAITEMS = FALSE;
%               default: FALSE
%
%       COMMENTS [=] [TRUE/FALSE]
%               show DASDL comments for structures and data items,
%               if any;
%               default: FALSE
%
%       FILTER filtername
%               apply the filter to the datasets and data items returned
%
%       SUPPORT "libraryname"
%               location of the filter
%
%
%       DATASET [=] datasetname
%               list this dataset; not every dataset
%               this option may be repeated for all desired datasets
```

```
 Names Only  = false
 Parent      = true
 Max Records = false
 Comments    = false

 Sets        = true     Subsets = false    Keys = true

 Dataitems   = true     Null = false

 Filter        DBFILTER
 Support       "OBJECT/DATABRIDGE/SUPPORT" % add .../dbname if filtering

% Dataset   = CUSTOMER
% Dataset   = ACCOUNT
```

Lister Report

Following is a brief explanation of the main parts of the Lister report. Note that the report uses standard DMSII terms such as set, key, data type, offset, etc.

Dataset — Each data set is identified by its structure number and its name. In addition, the following information may appear:

Column	Description
Format level	The DMSII update (format) level.
RecType	The record type value for a variable format data set. Note that RecType does not appear for fixed format data sets.
RecSz	The size of the record in words. Words is a common way of expressing size on a ClearPath NX, LX, or A Series host. You can convert words into bytes as follows: 1 word = 6 bytes
MaxRecs	This is an upper bound record count estimate; it is not meant to be an accurate count of the records. Note that the MaxRecs count appears only when you set MAX RECORDS = TRUE.

Set — Sets are listed underneath their parent data set name and before its data item list, if included. Each set is identified by the set structure number and set name. Lister also reports if the set allows duplicates.

Subset — Subsets are listed after any sets and before the data item list, if included. Each subset is identified by the subset structure number and subset name. Lister also reports if the subset allows duplicates.

Key — Keys are listed underneath their set or subset. Descending keys will be labeled as “Descending.”

DataItems — The next several lines list the data items in the data set. The columns are explained in the following table.

Column Title	Description
Num	Data item number.

Name	Data item name. An asterisk (*) in front of the name indicates that it is a required item, i.e. it cannot be null.
Type	Type of data item, which can be one of the following: Number, Alpha, Binary, Boolean, Field, Group, Image, RecType, Link, and Kanji (for WIDE (n)).
Offset	Offset to the data item in digits (half-bytes). Offsets indicate the location of a data item in a record formatted by BINARYFORMAT. If the offset is a relative offset, rather than an absolute offset, a “+” precedes the number. Items in a group having an OCCURS clause have relative offsets. If the offset is a raw offset as opposed to a formatted offset, a "@" precedes the number.
Size	The physical size of the data item in 4-bit digits (half-bytes). Compare this to DecSz, which is a user declared size.
DecSz	The user-declared size of the item in the length of the field. The units of length vary according to the field type, as follows: NUMBER type is 4-bit digits (same as half-byte) ALPHA type is bytes GROUP is 0 Compare this to the Size column (explained above). The Size column shows the physical size of the data item in 4-bit digits.
Decs	If the data item is numeric, the Decs column shows the number of digits to the right of the assumed decimal point. If the numeric has no decimals, the Decs field is blank. If the data item is Boolean, the Decs column shows the number of the bit within the four bit digit that contains the value of the Boolean. If the Decs field is blank, the bit number is zero.
Sign	If the data item is numeric, an S indicates it has a sign; if the numeric has no sign, the field is blank.
Occurs	The data item OCCURS value. For example, a data item called MONTH could occur 12 times.
Subs	The number of subscripts the data item requires for occurring items or groups. For example, an occurring data item within an occurring GROUP would have a SUBS = 2.
Target	The structure number referenced by a link item. The example Lister report does not show the Target column because Links were not enabled for the database.

Sample Lister Report

Following shows a partial sample report where the parameter file contained the settings shown in the “Lister Report” on page 171.

```
Databridge Lister Report for BANKDB using (GPC)DESCRIPTION/BANKDB ON USER
     Created Friday, March 18, 2016 @ 14:39:55
     Version 6.6.0.001 compiled Saturday, October 31, 2015 @ 13:12:17

Reading parameter file (DB66)DATA/LISTER/BANKDB/CONTROL ON USER

List Options
------------
Names only  = False
Max records = False
Comments    = False
Parent      = True
Sets        = True
Subsets     = False
Keys        = True
DataItems   = True
Null        = False
Links       = False
Filter      = DBFILTER
Support     = OBJECT/DATABRIDGE/SUPPORT

#0001 Dataset: BANKDB Format level 0 ItemCount 37 RecSz 10
Num  Name             Type    Offset  Size DecSz Decs Sign Occurs Subs
   3 GLB-REAL          Float        0    12     0        S
   4 GLB-ALPHA         Alpha       12    30    15
   5 GLB-TS            Float       42    12     0        S
   6 GLB-BANK-NAME     Alpha       54    60    30
   7 GLB-FIELD         Field      114     3    10
   8 GLB-BOOLEAN       Boolean    117     1     1
   9 GLB-FLAGS         Field      118     1     2
  10 GLB-F1            Boolean    118     1     0    1
  11 GLB-F2            Boolean    118     1     0

#0002 Dataset: DBTWINCONTROL Format level 0 ItemCount 13 RecSz 13
          Primary Key: DBTWIN-STRNUM
                       DBTWIN-RECTYPE
#0003     Set: DBTWINSET
          Key: DBTWIN-STRNUM
               DBTWIN-RECTYPE
Num  Name             Type    Offset  Size DecSz Decs Sign Occurs Subs
   1*DBTWIN-STRNUM     Binary       0    12    11        S
   2*DBTWIN-RECTYPE    Binary      12    12    11        S
   3 DBTWIN-AFN        Binary      24    12    11        S
   4 DBTWIN-ABSN       Binary      36    12    11        S
   5 DBTWIN-SEG        Binary      48    12    11        S
   6 DBTWIN-INX        Binary      60    12    11        S
   7 DBTWIN-TIME       Float       72    12     0        S
   8 DBTWIN-MODE       Binary      84    12    11        S
   9 DBTWIN-FORMAT-LVL Binary      96    12    11        S
  10 DBTWIN-TABLE-LVL  Binary     108    12    11        S
  11 DBTWIN-ITEM-COUNT Binary     120    12    11        S
  12 DBTWIN-OPTIONS    Binary     132    12    11        S
  13 DBTWIN-HOST-INFO  Binary     144    12    11        S

#0004 Dataset: RG Format level 0 ItemCount 4 RecSz 10
```

```
      Remap of BANKDB
Num  Name              Type     Offset  Size DecSz Decs Sign Occurs Subs
   1 RG-REAL           Float         0    12     0        S
   2 RG-ALPHA          Alpha        12    30    15
   3 RG-TS             Float        42    12     0        S
   4 RG-BANK-NAME      Alpha        54    60    30

#0005 Dataset: MASTER-DS Format level 0 ItemCount 3 RecSz 1
          Primary Key: MST-ITEM
#0010     Set: MASTER-SET
          Key: MST-ITEM
Num  Name              Type     Offset  Size DecSz Decs Sign Occurs Subs
   1*MST-ITEM          Alpha         0    12     6

#0006 Dataset: ORDERED-DS Format level 0 ItemCount 2 RecSz 2
#0005  Parent: MASTER-DS
          Primary Key: ORD-ITEM
#0007     Set: ORD-ACCESS
          Key: ORD-ITEM
Num  Name              Type     Offset  Size DecSz Decs Sign Occurs Subs
   1*ORD-ITEM          Number        0    12    12
   2 ORD-NAME          Alpha        12     6     3
```

9 Chapter 9: License Support

This chapter explains what the Databridge License Support library is and how to use it. The Databridge License Support library (DBLicenseSupport) ensures compliance with the licensed features of Databridge.

In this Chapter

Installing License Support

The Install WFL will register DBLicenseSupport as a system library with the function name DBLICENSESUPPORT. To install a new version of DBLicenseSupport use the following command:

```
SL DBLICENSESUPPORT = (usercode)OBJECT/DATABRIDGE/LICENSESUPPORT ON
family:ONEONLY
```

where *usercode* is the usercode where Databridge is installed and family is the pack family. You will also need to do a `jobnumber AX QUIT` where *jobnumber* is the mix number of the old DBLicenseSupport. The previous copy of DBLicenseSupport will not terminate until all Databridge programs terminate that are linked to it.

If you DS DBLicenseSupport, it will DS all Databridge programs.

License Key File

DATA/DATABRIDGE/LICENSE contains one or more license keys indicating the purchased features and allowed number of hosts for each feature. This file must be under the same usercode and pack family as DBLicenseSupport. This file cannot be edited directly and can only contain the license key (s).

Use the DBLicenseManager program to add or inquire on license keys. Run it using the following CANDE command.

```
R DATABRIDGE/LICENSEMANAGER
```

License Expiration

Licenses expire at the end of a month. There is no grace period. During the month a license will expire, DBLicenseSupport will display a warning message that the key will expire.

Example 9-1 *(2) [0923]>>> Warning: Databridge license expires at month end <<<*

If a product is accessed after its key has expired, DBLicenseSupport will display an error message and return an error code to the Databridge Accessory.

Example 9-2 *#40784 DBLicenseSupport: ERROR: License for Enterprise expired 2/2015*

If you attempt to run a Databridge product from more host computers than are licensed, DBLicenseSupport will display an error message listing the release number and names of the hosts already using the license and return an error code to the Databridge Accessory. For example, if you have a one-host DBEnterprise license and you attempt to run release 6.6 DBEnterprise on host Saturn but DBEnterprise has already been run host Earth, you will see the following message:

Example 9-3 *#40523 DBLicenseSupport: ERROR: Enterprise max installs: 1 [Earth 6.6] ==> no license for Saturn 6.6*

To resolve a license problem, secure a new license key from Micro Focus and add it to the license file using the DBLicenseManager program (see "License Key File" on page 175). The next connection to Databridge will automatically use the new key. You can run DBLicenseManager at any time to see the licensed products and what hosts are using them.

License Support AX Commands

You can use the following AX commands with DBLicenseSupport.

The *jobnumber* in the AX command is the mix number of DBLicenseSupport.

To find the mix number, use the ODT or MARC command `LIBS NAME =LICENSESUPPORT=`.

`AX QUIT` DBLicenseSupport will unfreeze. When the last linked Accessory, e.g. DBServer, terminates, DBLicenseSupport will also terminate.

`AX STATUS` DBLicenseSupport will display the licenses in use by each host and when they expire. For example, the response to a `1791 AX STATUS` command might be:

```
1791 07:40 DBLicenseSupport: --- Databridge Licenses in use ---
1791 07:40 DBLicenseSupport: 6.6 File Transfer on EARTH expires 12/2017
1791 07:40 DBLicenseSupport: 6.6 Enterprise on EARTH expires 12/2017
1791 07:40 DBLicenseSupport: 6.6 DMSII Client on MARS (DB66) expires
12/2017
1791 07:40 DBLicenseSupport: 6.6 Twin on MARS (DBTWIN66) expires 12/
2017
1791 07:40 DBLicenseSupport: 6.6 Client on EARTH expires 12/2017
1791 07:40 DBLicenseSupport: 6.6 Host on MARS (DBTWIN66) expires 12/
2017
```

10 Chapter 10: Utilities

This chapter explains how to use the WFL (Work Flow Language) jobs and additional utilities included with Databridge.

In this Chapter

BCNotify

Use BCNotify to remotely launch scripts on a machine running the Databridge Client service. BCNotify has two modes of operation, each of which has its own WFL:

Mode	WFL
Standard mode	WFL/DATABRIDGE/SAMPLE/BCNOTIFY
DBNotify compatible mode	WFL/DATABRIDGE/SAMPLE/DBNOTIFY

Standard mode accepts a single string parameter. This string parameter is a list of <keyword>=<value> pairs that are used to pass items such as the DMSII database (for obtaining the current AFN), the remote command to execute, parameters to pass to the command, and the remote data source.

DBNotify compatible mode also accepts a string parameter. The DBNotify WFL is provided only for back compatibility with the deprecated DBNotify program (OBJECT/DATABRIDGE/DBNOTIFY) and WFL (WFL/DATABRIDGE/DBNOTIFY), which are no longer supported. The BCNotify program is executed in a manner nearly identical to DBNotify.

For information about using BCNotify to automate client operations, see "Automating Client Operations with the Service" in the *Databridge Client Administrator's Guide*.

BCNotify WFL Parameters

The following table lists parameters to use with the BCNotify WFL.

This parameter	Does this
CMD -or- COMMAND	Identifies the script or executable on the remote system. This is a required parameter.
PARAMETERS -or- ARGS	Passes the command-line arguments to the command above. This is an optional parameter.
DATASOURCE -or- DSOURCE	Identifies the data source of the remote system. DATASOURCE is a required parameter and must be a valid data source.
DATABASE -or- DB	Identifies the DMSII database. This parameter is optional and is currently used to obtain the current audit file.

Example

```
START WFL/DATABRIDGE/BCNOTIFY ("CMD=STARTPROCESS  ARGS=P1 P2 P3
DATASOURCE=BANKDB")
ST WFL/DATABRIDGE/BCNOTIFY("DB=(DBUSER)TESTDB ON DBASE CMD=PROCESS
DATASOURCE=TDB")
```

Notify (WFL)

The Notify WFL causes DBServer to notify Enterprise Server that audit files are available for processing. You must run the Notify WFL each time you want to initiate the Databridge Client from the MCP host system. You can automate this process by using the Copy Audit utility to have DBServer automatically notify Enterprise Server each time an audit file becomes available. See “Copy Audit Utility” on page 186.

To run the Notify WFL

NOTE: For instructions to steps 1-2, see the *Databridge Enterprise Administrator’s Guide*.

1. Configure Enterprise Server on the Enterprise Server computer (ClearPath PC or Windows server) to listen at a specific port.
2. Create a batch file to run the Databridge Client when Enterprise Server is notified by DBServer.
3. Configure the NOTIFY option in the DBServer parameter file with the Enterprise Server computer’s IP address and port number where Enterprise Server (specifically, the Databridge Director service) is listening. See “NOTIFY” on page 96 for instructions.
4. Run the Notify WFL as follows:

   START WFL/DATABRIDGE/NOTIFY ("*databasename*")

   where *databasename* is the name of the database you are replicating.

Save with Update Level (WFL)

Use this procedure to ensure that the Engine finds the correct software when processing audit files that switch from one update level to another. The Save with Update Level WFL (WFL/DATABRIDGE/BACKUPTAILORED) provides backup copies of both the DESCRIPTION and DMSUPPORT files, helping you plan ahead for record format or file format database reorganizations. You can run this WFL any time before or after a DASDL update.

To run the Save with Update Level WFL

- Start WFL/DATABRIDGE/BACKUPTAILORED using the following syntax:

  START WFL/DATABRIDGE/BACKUPTAILORED ("*databasename*")

  where *databasename* is the name of the database, with or without a leading user code and family name. (This is the same as all of the other WFLs.)

  For example:

  ```
  START WFL/DATABRIDGE/BACKUPTAILORED ("CUSTOMERDB")
  START WFL/DATABRIDGE/BACKUPTAILORED ("(PROD)CUSTOMERDB ON USERPACK")
  ```

  If the database update level of both files is 17, the files would be copied using the following filenames:

  ```
  DESCRIPTION/CUSTOMERDB/17
  DMSUPPORT/CUSTOMERDB/17
  ```

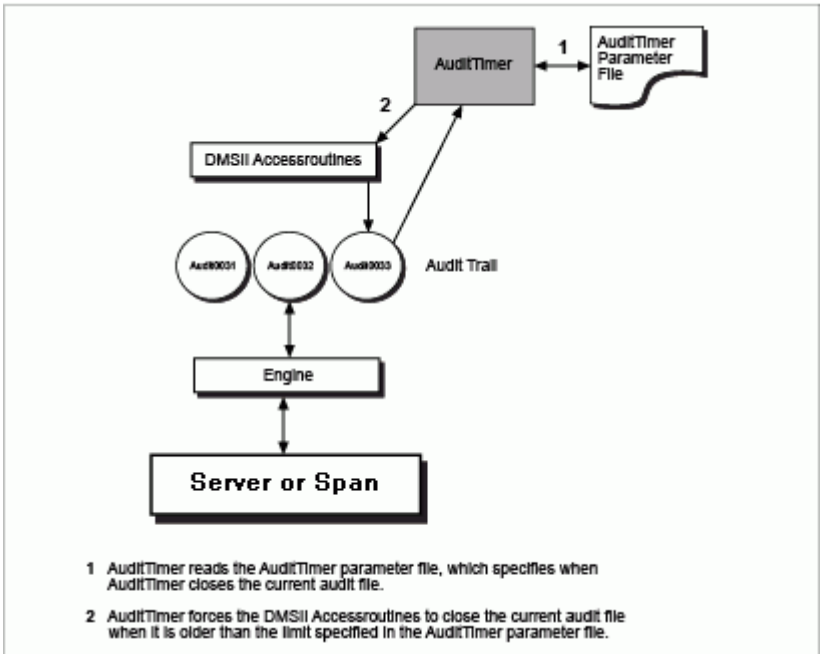
  If the update level of the DESCRIPTION file is 39 and the update level of the DMSUPPORT library is 40, the files would be copied using the following filenames:

  ```
  DESCRIPTION/CUSTOMERDB/39
  DMSUPPORT/CUSTOMERDB/40
  ```

  The copies will reside on the same families as the original files.

AuditTimer Utility

Databridge AuditTimer is a utility that closes the current audit file when it is older than the limit you specify in the AuditTimer parameter file (DATA/AUDITTIMER/*databasename*/CONTROL), as shown in the following diagram:



You can use the AuditTimer utility with both Span and Server as a way to periodically force an audit switch for a database.

The files related to the AuditTimer utility are as follows:

```
OBJECT/DATABRIDGE/AUDITTIMER
WFL/DATABRIDGE/AUDITTIMER
DATA/AUDITTIMER/SAMPLE/CONTROL
DATA/AUDITTIMER/databasename/CONTROL
```

AuditTimer Parameter File

The AuditTimer parameter file is named as follows:

DATA/AUDITTIMER/*databasename*/CONTROL

This file contains entries for days and times of a week that indicate how old the audit file can get before it is automatically closed, as in the following example:

```
day             time         interval
TUESDAY          8 AM        50 MINUTES
THURSDAY      5:00 PM        30 MINUTES
SATURDAY      8:00 AM         1 DAY      SYNC 4 HOURS
```

The interval (how old an audit file can get) for each entry applies to all times up to the next entry. In the example above, the 50-minute interval applies to all times between Tuesday at 8 a.m. and Thursday at 4:59 p.m. The 30-minute interval applies to all times between Thursday at 5 p.m. and Tuesday at 7:59 a.m.

In addition, the optional SYNC parameter ensures that a sync point will occur if any transactions are completed within a specified SYNC interval. This ensures that Databridge replicates all completed transactions even if there are not enough transactions to trigger the normal sync point mechanism.

When AuditTimer runs, it determines which entry corresponds to the current time. In the example above, if AuditTimer starts running at 9:30 a.m. on Wednesday, it will use the TUESDAY 8 a.m. 50 MINUTES entry. It then computes the age of the current audit file. If the audit file is older than the specified interval (50 minutes in this example), AuditTimer will force an audit close.

Note that these entries do not determine how often AuditTimer runs. To determine how often AuditTimer runs, set the STARTTIME parameter explained in "Starting AuditTimer" on page 184.

Configuring the AuditTimer Parameter File

Complete the following steps to configure AuditTimer.

To configure AuditTimer

1. Using CANDE or another editor, open the sample AuditTimer parameter file (`DATA/AUDITTIMER/SAMPLE/CONTROL`).
2. Using values from the table below, edit day, time, interval, and SYNC interval (optional). The format should be as follows:

   `daytimeinterval` `[ SYNC` *`interval`* `]`

   For example:

   ```
   MONDAY     7 PM      12 HOURS
   SATURDAY   8 AM       1 DAY    SYNC 30 MINUTES
   ```

   Save the sample parameter file as follows:

   `DATA/AUDITTIMER/`*`databasename`*`/CONTROL`

   where *databasename* is the name of the database for which you will be automatically closing the audit files.

Entry	Syntax
day	SUN or SUNDAY MON or MONDAY TUE or TUESDAY WED or WEDNESDAY THU or THURSDAY FRI or FRIDAY SAT or SATURDAY
time	`HH AM or PM` `HH:MM AM or PM` **NOTE:** If you omit the AM/PM, a 24-hour time format is assumed. For example, 23:30 is the same as 11:30 PM.
interval	MIN or MINUTES HR or HOURS SEC or SECONDS DAY or DAYS

Example 1

The following procedure is an example of how to configure AuditTimer to force an audit file close between 5:00 p.m. and 7:00 p.m. each evening.

1. Using CANDE or another editor, open the sample AuditTimer parameter file (`DATA/AUDITTIMER/SAMPLE/CONTROL`).
2. Enter the following values for day, time, and interval:

```
SUNDAY     5 PM      2 HOURS
SUNDAY     7:10 PM   24 HOURS
MONDAY     5 PM      2 HOURS
MONDAY     7:10 PM   24 HOURS
TUESDAY    5 PM      2 HOURS
TUESDAY     7:10 PM   24 HOURS
WEDNESDAY  5 PM      2 HOURS
WEDNESDAY  7:10 PM   24 HOURS
THURSDAY   5 PM      2 HOURS
THURSDAY   7:10 PM   24 HOURS
FRIDAY     5 PM      2 HOURS
FRIDAY     7:10 PM   24 HOURS
SATURDAY   5 PM      2 HOURS
SATURDAY   7:10 PM   24 HOURS
```

3 Save the sample parameter file as follows:

DATA/AUDITTIMER/*databasename*/CONTROL

where *databasename* is the name of the database for which you will be automatically closing the audit files.

4 Run AuditTimer as follows:

START WFL/DATABRIDGE/AUDITTIMER ("*databasename*");

AuditTimer will determine the parameter file entry corresponding to the current time and then compute the age of the current audit file. If the current audit file is too old, it will force an audit switch. Then AuditTimer will compute the next time that an audit file could possibly be too old and re-queues itself for that time.

For example, if AuditTimer starts running Monday at 5:30 p.m., it will ensure that the current audit file is not older than two hours. If it is 90 minutes old, AuditTimer will re-queue itself to run at 6:00 p.m.

On the other hand, if the current audit file is only 10 minutes old, it wouldn't be two hours old until 7:20 p.m., which is after the next parameter file entry (MONDAY 7:10 PM 24 HOURS) takes effect. Since audit files can be 24 hours old during that entry, the audit file created at 5:20 p.m. won’t be too old until 5:20 p.m. the next day, which is after the next parameter file entry (TUESDAY 5 PM 2 HOURS) takes effect. Therefore AuditTimer will re-queue itself for Tuesday at 5:00 p.m.

Example 2

The following example uses the SYNC option to ensure that a sync point is forced every four hours if any transactions are committed within that interval.

Building on the previous example, AuditTimer forces an audit file close between 5:00 p.m. and 7:00 p.m. each weekday evening. On the weekend (Saturday, Sunday, and up to Monday at 5:00 PM), however, the audit file can be up to three days old. AuditTimer ensures that a sync point will occur within four hours of any transaction.

```
MONDAY     5 PM        2 HOURS
MONDAY     7:10 PM   24 HOURS
TUESDAY    5 PM        2 HOURS
TUESAY     7:10 PM   24 HOURS
WEDNESDAY  5 PM        2 HOURS
WEDNESDAY  7:10 PM   24 HOURS
THURSDAY   5 PM        2 HOURS
THURSDAY   7:10 PM   24 HOURS
FRIDAY     5 PM        2 HOURS
FRIDAY     7:10 PM   24 HOURS
SATURDAY   8 AM        3 DAYS    SYNC 4 HOURS
```

Sample AuditTimer Parameter File

Following is the sample AuditTimer parameter file (DATA/AUDITTIMER/SAMPLE/CONTROL).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                      %
%             A Series Databridge AuditTimer Parameter File            %
%                                                                      %
%       Source:         DATA/AUDITTIMER/SAMPLE/CONTROL                 %
%                                                                      %
%       Version:        6.6                                            %
%                                                                      %
%      Copyright (C) 2019 Micro Focus or one of its affiliates.        %
%                                                                      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%       Each entry specifies how "old" the current audit file can be.
%       If the current audit file is older than the given limit for the
%       time of day, the AuditTimer program will force an audit switch.
%       Also, if a syncpoint interval is specified, AuditTimer will
%       ensure that at least one syncpoint occurs within that interval
%       if any transactions were completed.
%
%       Syntax for each entry:
%
%               day time interval [ SYNC interval ]
%
%       where
%
%       'day' is:
%            ---- SUN or SUNDAY ------------|
%              |- MON or MONDAY ----|
%              |- TUE or TUESDAY ---|
%              |- WED or WEDNESDAY -|
%              |- THU or THURSDAY --|
%              |- FRI or FRIDAY ----|
%              |- SAT or SATURDAY --|
%
%       'time' is:
%            -- hh -----------------------|
%                    |- :mm -|   |- AM -|
%                                |- PM -|
%
```

```
%       If you omit the AM/PM a 24-hour time is assumed.
%
%       'interval' is:
%            -- n --- MIN -----------------|
%                  |- MINUTES -|
%                  |- HR ------|
%                  |- HOURS ---|
%                  |- SEC -----|
%                  |- SECONDS -|
%                  |- DAY  ----|
%                  |- DAYS ----|
%
%       If you omit 'SYNC interval' AuditTimer will not force any
%       syncpoints.
%
% Some (not necessarily realistic) examples:
%
%       TUESDAY   8 AM      50 MINUTES
%       THURSDAY 12:00       2 HOURS    % noon
%       THURSDAY 5:00 PM    30 MINUTES
%       THURSDAY 19:30      45 MIN      % 7:30 PM
%       SATURDAY 8:00 AM     1 DAY SYNC 30 MINUTES
%               Starting Saturday morning audit files can be a whole
%               day old but force a syncpoint every half hour if any
%               transactions are committed.
%-----------------------------------------------------------------------
SUNDAY    1 AM          4 HOURS
MONDAY    9 AM          1 HOURS
MONDAY    5 PM          2 HOURS
TUESDAY   9 AM          1 HOURS
TUESDAY   5 PM          2 HOURS
WEDNESDAY 9 AM          1 HOURS
WEDNESDAY 5 PM          2 HOURS
THURSDAY  9 AM          1 HOURS
THURSDAY  5 PM          2 HOURS
FRIDAY    9 AM          1 HOURS
FRIDAY    5 PM          3 HOURS
```

Starting AuditTimer

Use the following procedure to start AuditTimer.

To start AuditTImer

1. Configure the AuditTimer parameter file.
2. Run the AuditTimer WFL (WFL/DATABRIDGE/AUDITTIMER) to execute AuditTimer, as follows:

   ```
   START WFL/DATABRIDGE/AUDITTIMER ("databasename" [ ,
   "logicaldatabasename" ] ) [ ; STARTTIME = hh:mm ]
   ```

   where STARTTIME = *hh:mm* is an optional command that specifies when AuditTimer starts. AuditTimer will start immediately if you do not specify the optional STARTTIME command. If guard files prevent access to the physical database, specify a logical database name.

3 If you use AuditTimer for more than one database, execute it once for each database. For example, if you have three DMSII databases for which you want to periodically force an audit file switch, you would run the AuditTimer utility three times, once for each database, as follows:

```
START WFL/DATABRIDGE/AUDITTIMER ("databasename1")
START WFL/DATABRIDGE/AUDITTIMER ("databasename2")
START WFL/DATABRIDGE/AUDITTIMER ("databasename3")
```

After AuditTimer starts, it automatically queues itself so that it can periodically check the age of the current audit file. By default, it queues itself to run the next time it expects the audit file might be older than what the parameter file specifies. If you want to override this behavior, configure AuditTimer to run at specified intervals.

To change the frequency of AuditTimer

- Modify the STARTTIME option near the bottom of the AuditTimer WFL. The following excerpt shows a STARTTIME value of 2.5 hours:

```
% schedule this job to run again later
START WFL/DATABRIDGE/AUDITTIMER (DBPARAM, LDB, SYNCTIME, TRANCOUNT,
SYNCCOUNT);
STARTTIME = +2:30 (NOTE);
END JOB.
```

Forcing an Audit Switch

If you generate less than one audit file a day, you may want to force an audit switch daily and run Span or DBServer so that the secondary (replicated) database is at most one day behind. In this case, you can use the following procedure instead of running AuditTimer.

The following procedure works only for databases that are currently open for updating (that is, one or more programs have done an OPEN UPDATE*databasename*).

To force an audit switch

1 From the ODT or action line in MARC, transmit the following:

```
DBS
```

A list of active databases and their associated mix numbers appears.

2 Transmit the following for the database audit file you want to close:

```
mixnumber SM AUDIT CLOSE FORCE
```

where *mixnumber* is the mix number of an active database.

The current audit file is now closed, and a new one is created.

You can also use Audit Close utility to close the current audit file. See “Audit Close Utility” on page 188. For additional information on forcing an audit switch, see the Unisys DMSII Utilities documentation.

Terminating AuditTimer

DBAuditTimer is designed to stay in the mix so that it is always available to check audit files; therefore, it automatically re-queues itself after each run. If for some reason you want to quit DBAuditTimer, follow these steps:

1. From the ODT or MARC action line, type SQ and click **Transmit** to determine DBAuditTimer's next job number (the mix number). The following is an example of the information that appears when you transmit an SQ command. (Note that this example uses a sample database called BANKDB.)

   ```
   QUEUE 1 (FIRST OF 2 ENTRIES)
   2984 50 (usercode) DBAUDITTIMER ("BANKDB ON HOST","")
   QUEUED: 12/9/2009 AT 15:53:59 STARTTIME=16:53:58
   QUEUE 7
   NO ENTRIES
   ```

2. Type the following DS command for that job number and click **Transmit**:

   mixnumber DS

   where *mixnumber* is the mix number that you received from the SQ command. For the example above, the DS command is:

   ```
   2984 DS
   ```

   The following message appears, indicating that DBAuditTimer is no longer running:

   mixnumber OPERATOR DSED FROM QUEUE1

Copy Audit Utility

Typically, each time an audit file closes, the DMSII Accessroutines starts the standard DMSII DATABASE/WFL/COPYAUDIT, which does the following:

- Copies the closed audit file to tape
- Removes the closed audit file from disk

The Databridge Copy Audit utility (WFL/DATABRIDGE/COPYAUDIT) can intercept the copy audit parameters issued by DMSII, modify them, and then pass the new parameters to DATABASE/WFL/COPYAUDIT.

The Databridge Copy Audit utility enables you to do the following:

- Specify the number of closed audit files that should be saved on disk. The default is 2 to accommodate transactions that start in one audit file but end in another. The Copy Audit utility deletes any "REMOVE" from the original DMSII COPYAUDIT parameters and then starts DATABASE/WFL/COPYAUDIT with the modified parameter string.
- Automatically notify DBServer, which then notifies Enterprise Server to start a batch file running the Client each time an audit file becomes available. Using this feature keeps the Client database less than one audit file behind your primary database.
- Run Databridge Span each time an audit file becomes available.

To use the Copy Audit utility

1. Using CANDE or another editor, open the Copy Audit WFL (WFL/DATABRIDGE/COPYAUDIT).
2. To automatically notify Databridge Server and the Client each time an audit file becomes available, set the WANTNOTIFY entry to True.
3. For Databridge Server to notify the Client when an audit file becomes available, you must also configure the NOTIFY parameter in the DBServer parameter file. See "NOTIFY" on page 96.
4. To change the number of closed audit files saved on disk, set the CONSTANT KEEP entry to the number of closed audit files that you want to keep on disk.

   The default and minimum value is 2 to accommodate transactions that start in one audit file but end in another. In other words, the newly closed audit file and the previously closed audit file remain on disk. Only the third oldest audit file is removed from disk. This means that at all times three audit files (one current, two old) will be available on the host.
5. To run Databridge Span each time an audit file becomes available, remove the comment sign (%) before the following line and then enter the name of the database you are replicating:

   ```
   %START WFL/DATABRIDGE/SPAN (databasename);
   ```
6. If you only run one copy of Databridge Server on the host, skip to the last step (8). If you run more than one copy of Databridge Server on the same host, you must create a unique name for the port file (AUDITNOTIFY, by default) that each Databridge Server uses to listen for notifications.

   Change the following line

   ```
   FILE AUDITNOTIFY (FILENAME = AUDITNOTIFY);
   ```

   to

   ```
   FILE AUDITNOTIFY (FILENAME = TESTNOTIFY);
   ```
7. Change the corresponding line in the WFL/DATABRIDGE/NOTIFY and/or WFL/DATABRIDGE/COPYAUDIT files from

   ```
   FILENAME = AUDITNOTIFY,
   ```

   to

   ```
   FILENAME = TESTNOTIFY,
   ```
8. Save the WFL.

You can modify the DMSII DASDL to start the Copy Audit utility WFL instead of the DMSII default COPYAUDIT WFL for the database you want to replicate. For instructions, see "Modify the DASDL" on page 188.

Modify the DASDL

You can modify the DASDL to start the Copy Audit utility WFL/DATABRIDGE/COPYAUDIT instead of the DMSII default COPYAUDIT WFL. Full details on modifying DASDLs are available in the Unisys DMSII documentation.

To modify the DASDL, open the DASDL and modify the AUDIT TRAIL section. If you are using the copy option, specify WFL/DATABRIDGE/COPYAUDIT for JOB. If you are not using the copy option, add the JOB declaration as shown in example on the next page.

Following is an excerpt of how your existing DASDL might look:

```
AUDIT TRAIL ( option option
COPY TO TAPE AND REMOVE option option );
```

Following is the same excerpt with the correct information to start the Databridge Copy Audit WFL.

```
AUDIT TRAIL ( option option
COPY TO TAPE AND REMOVE JOB WFL/DATABRIDGE/COPYAUDIT option option
);
```

Audit Close Utility

The Databridge Audit Close utility enables you to close the current audit file and open a new one.

To close the current audit file, run the Audit Close utility as follows:

```
START WFL/DATABRIDGE/SAMPLE/AUDITCLOSE ("databasename")
```

where *databasename* is the name of the database whose audit file you want to close.

Audit Remove Utility

After a Databridge Accessory processes audit files that were copied to a separate audit pack, you can use the Audit Remove utility to remove the processed audit files.

To configure the Audit Remove utility

1. Using CANDE or another editor, open the sample Audit Remove WFL (`WFL/DATABRIDGE/SAMPLE/REMOVEAUDIT`).
2. Modify the values of KEEP, AUDITPREFIX, and AUDITPACK.
3. Save the sample WFL as follows:

   `WFL/DATABRIDGE/REMOVEAUDIT/databasename`

   where *databasename* is the name of the database you are replicating.
4. Configure the AUDIT JOB option in the DBServer or Databridge Span parameter file to specify the following:

   `WFL/DATABRIDGE/REMOVEAUDIT/databasename`

   See instructions for Databridge Server "AUDIT JOB" on page 93 or Databridge Span "AUDIT JOB" on page 115.

5 Run the Accessory as usual.

After the Accessory processes the audit file, it will start the REMOVEAUDIT WFL, which removes the specified audit file.

Audit Mirroring

You can find instructions for configuring the audit mirror utility in the *Databridge Enterprise Server Administrator Guide*.

DBInfo Utility

DBInfo enables you to create reports about a specified DMSII database, along with mainframe identification information and Databridge licenses and software. You can run DBInfo in either normal mode or interactive mode. The following table explains the benefits of each:

In this mode	You can
Normal mode	◆ Display the current (active) AFN ◆ Display the current database update level ◆ Return the number of sections in an audit file ◆ Generate a report that includes: various timestamps; update levels; DMSII release level; AFN; ABSN; whether the database uses guard files; and whether INDEPENDENTTRANS is set to true or false for a DMSII database. (This information is gathered from the DESCRIPTION file, the DMSUPPORT file, and the CONTROL file). The report also includes mainframe identification information, DBEngine runtime parameters, and Databridge licenses and software.
Interactive mode	◆ Find the first quiet point in an audit trail from a specified AFN and ABSN (you can run DBInfo in normal mode to determine the current AFN and ABSN). ◆ For each found quiet point, DBInfo displays the AFN, ABSN, SEG, and INX. You can then use this information to configure Databridge Span or the Databridge Client after a rollback.

Run DBInfo in Normal Mode

Use this procedure to run DBInfo in normal mode. You can run DBInfo in normal mode at any time. This allows you to do any of the following:

◆ Generate a report of various timestamps, update levels, etc., for a DMSII database by reading the `DESCRIPTION` file, `DMSUPPORT` file, and `CONTROL` file.

The report also includes mainframe identification information, runtime parameters that DBEngine uses for tracking a particular database, and Databridge licenses and software.

The runtime parameters reflect default values, values in the global parameter file (DATA/ENGINE/CONTROL), and values in the database-specific parameter file (DATA/ENGINE/*databasename*/CONTROL or DATA/ENGINE/*databasename/logicaldatabasename*/CONTROL).

- Display the current (active) AFN and ABSN.
- Determine the DMSII release level, whether guard files are in use, and the setting for INDEPENDENTTRANS.
- Return the number of sections in an audit file (See the REMOVEAUDITFILE subroutine in WFL/DATABRIDGE/SAMPLE/REMOVEAUDIT for an example.).

If you need to find the first quiet point in an audit trail from a specified AFN and ABSN, run DBInfo in interactive mode. See "Run DBInfo in Interactive Mode" on page 196.

If you are interested in the layout of your DMSII database, run the Lister Accessory, explained in "Chapter 8: Lister" on page 165.

To run Info Utility in normal mode

1. Start DBInfo in normal mode by doing one of the following:
   - To display the current (active) AFN for a specific database, type the command (including quotation marks)

     ```
     START WFL/DATABRIDGE/DBINFO ("databasename", "AFN"
     [, logicaldatabasename])
     ```

   - To display the current database update level, type the command (including quotation marks)

     ```
     START WFL/DATABRIDGE/DBINFO ("databasename", "LEVEL"
     [, "logicaldatabasename"])
     ```

   where *databasename* indicates the title of the DESCRIPTION file without the DESCRIPTION node.

   The WFL displays the information (AFN or update level). Since the current audit file number is returned in the DBInfo TASKVALUE attribute, you can alter the WFL to use that value in any way you see fit. You can have DBInfo return values to a WFL.
2. (Optional) If you want to run DBInfo using your own WFL so that it returns certain values to your WFL, use the following table:

Include this TASKVALUE	To return this
2	Current AFN
3	Number of sections in the audit file
4	Database update level

```
Databridge Database Info
Version 6.6.0.000 compiled Saturday, December 15, 2018 @
01:05:22

Databridge DBInfo Report generated Monday, April 29, 2019 @
13:51:04
LX100:1234 HYLOZOIST SSR 58.150.0453, Microcode 11.609 Thursday,
January 30, 2014

(DB66)DESCRIPTION/BANKDB ON HOST
------------------------------------
Desc. timestamp Tuesday, April 23, 2019@11:36:32
DASDL release 58.189
Property level 050810

DB name    BANKDB
DB timestamp Tuesday, April 23, 2019@11:36:32
Update level 1
Update timestamp Tuesday, April 23, 2019@11:36:38
DMSII release 58.189
DB prop. level 050810
ACCESSROUTINES SYSTEM/ACCESSROUTINES
RECOVERY   SYSTEM/DMRECOVERY
DATARECOVERY SYSTEM/DMDATARECOVERY
RECONSTRUCT RECONSTRUCT/BANKDB
GUARDFILE   <none>
Reorg required False
INDEPENDENTTRANS True
REAPPLYCOMPLETED True
Backout to SYNC False
RDSSTORE   False
Statistics  False
Is a MODEL  False

(DB66)DMSUPPORT/BANKDB ON HOST
---------------------------------
DB name    BANKDB
DB timestamp Tuesday, April 23, 2019@11:36:32
Update level 1
Update timestamp Tuesday, April 23, 2019@11:36:38
INDEPENDENTTRANS True
REAPPLYCOMPLETED True
Last structure 56
Phys. structures 41

(DB66)BANKDB/CONTROL ON HOST.
```

```
--------------------------------
CF format level 3900
CF release   58.189.8007
CF creation  Tuesday, April 23, 2019@11:36:41
DB timestamp  Tuesday, April 23, 2019@11:36:32
Update level  1
Update timestamp Tuesday, April 23, 2019@11:36:38
Reorg state  0
INDEPENDENTTRANS True
REAPPLYCOMPLETED True
INUSE     False
REBUILD    False
ROLLBACK    False
RECONSTRUCT  False
TRACKER    False
FAMILY change  False
Audit file nbr. 1
Audit block nbr. 123


Product (installs)      License expiration
--------------------------------------- ------------------------------
---------
Host (10)           never
Client (99)          never
Twin (10)           never
DMSII Client (10)        never
Enterprise (10)         never
AuditMirror (10)         never
File Transfer (10)        never
Databridge Cluster (10)     never


Parameters
----------
AFN change doc record = false
Checkpoint every QPT = false
Checkpoint idle DB  = false
Checkpoint Long Trans = false
Consolidate StateInfo = false
Documentation records = false
Don't wait on NO FILE = false
Embedded extracts  = false
Filtered item count  = false
Ignore reorgs    = false
Include link items  = false
Mirrored audit    = false
Mirrored database  = false
Modifies Before-After = false
Offline clone    = false
Population in DSInfo = false
Pre-error StateInfo  = false
Print statistics   = false
Read active audit  = false
Read uncached only  = false
```

```
Remote host readahead = false
Reversals as updates = false
StateInfo update lvl. = false
Unformatted updates = false
Ungrouped updates  = false
Wait rather than AX = false
Extract Workers  default: 10 (allow 1 - 4095)
Enterprise Workers  default: 0 (allow any)
Checkpoint Records  default: 1000 (allow any)
Checkpoint Blocks  default: 100 (allow any)
Checkpoint Transactions default: 0 (allow any)
Checkpoint Seconds  default: 0 (allow any)
Primary audit   = (no access)
DMUtility   = <default> SYSTEM/DMUTILITY.
DMControl   = <default> SYSTEM/DMCONTROL.
DMRecovery   = <default> SYSTEM/DMRECOVERY.
DMSIISupport   = DMSIISUPPORT
         []
Database available anytime

Version                    Program
Compile Timestamp
-----------------------  ------------------------------------------
---------------------------- ------------------------------
06.006.0003 SSR 58.189    (DB66)OBJECT/DATABRIDGE/SPAN ON HOST
Wednesday, February 13, 2019 @ 01:01:10
06.006.0000 SSR 58.189    (DB66)OBJECT/DATABRIDGE/DBINFO ON HOST
Saturday, December 15, 2018 @ 01:05:20
06.006.0014 SSR 58.189    (DB66)OBJECT/DATABRIDGE/ENGINE ON HOST
Friday, January 11, 2019 @ 01:02:02
06.006.0014 SSR 58.189 (DB66)OBJECT/DATABRIDGE/ENGINE/DEBUG ON HOST
Friday, January 11, 2019 @ 01:02:02
06.006.0014 SSR 58.189 (DB66)OBJECT/DATABRIDGE/ENGINE/STATS ON HOST
Friday, January 11, 2019 @ 01:01:37
06.006.0014 SSR 58.189 (DB66)OBJECT/DATABRIDGE/ENGINE/NOSTATS ON HOST
Friday, January 11, 2019 @ 01:01:09
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/LISTER ON HOST
Saturday, December 15, 2018 @ 01:04:39
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SAMPLE/SQLGEN ON HOST
Saturday, December 15, 2018 @ 01:05:34
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SAMPLE/READDOC ON HOST
Saturday, December 15, 2018 @ 01:05:45
06.006.0000  SSR 58.189  (DB66)OBJECT/DATABRIDGE/SAMPLE/COBOLGEN ON
HOST         Saturday, December 15, 2018 @ 01:05:36
06.006.0000  SSR 58.189  (DB66)OBJECT/DATABRIDGE/SAMPLE/DASDLGEN ON
HOST         Saturday, December 15, 2018 @ 01:05:42
06.006.0000  SSR 58.189  (DB66)OBJECT/DATABRIDGE/SAMPLE/REFORMAT ON
HOST         Saturday, December 15, 2018 @ 01:05:30
06.006.0000  SSR 58.189  (DB66)OBJECT/DATABRIDGE/SAMPLE/AUDITCLOSE ON
HOST        Saturday, December 15, 2018 @ 01:05:40
00.000.0000  SSR 58.150  (DB66)OBJECT/DATABRIDGE/SAMPLE/EXTRACTADDRESS
ON HOST      Saturday, December 15, 2018 @ 01:05:31
06.006.0007 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SERVER ON HOST
Wednesday, February 13, 2019 @ 01:01:41
06.006.0007 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SERVER/DEBUG ON HOST
```

```
Wednesday, February 13, 2019 @ 01:01:41
06.006.0007 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SERVER/STATS ON HOST
Wednesday, February 13, 2019 @ 01:01:31
06.006.0007 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SERVER/NOSTATS ON HOST
Wednesday, February 13, 2019 @ 01:01:22
00.000.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/DCKEYIN ON HOST
Saturday, December 15, 2018 @ 01:00:56
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SUPPORT ON HOST
Saturday, December 15, 2018 @ 01:04:30
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SUPPORT/BANKDB ON HOST
Tuesday, April 23, 2019 @ 11:38:38
06.006.0000  SSR 58.189  (DB66)OBJECT/DATABRIDGE/SUPPORT/BANKDB/1 ON
HOST           Tuesday, April 23, 2019 @ 11:38:38
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SUPPORT/TESTDB ON HOST
Wednesday, April 10, 2019 @ 17:52:11
06.006.0000  SSR 58.189  (DB66)OBJECT/DATABRIDGE/SUPPORT/TESTDB/23 ON
HOST         Wednesday, April 10, 2019 @ 17:52:11
06.003.0009 SSR 58.189 (DB66)OBJECT/DATABRIDGE/BCNOTIFY ON HOST
Saturday, December 15, 2018 @ 01:01:07
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/SNAPSHOT ON HOST
Saturday, December 15, 2018 @ 01:05:23
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/GENFORMAT ON HOST
Saturday, December 15, 2018 @ 01:03:53
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/AUDITTIMER ON HOST
Saturday, December 15, 2018 @ 01:04:20
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/CHANGEUSER ON HOST
Saturday, December 15, 2018 @ 01:05:17
06.006.0001 SSR 58.189 (DB66)OBJECT/DATABRIDGE/AUDITMIRROR ON HOST
Saturday, December 15, 2018 @ 01:04:22
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/COBOLSUPPORT ON HOST
Saturday, December 15, 2018 @ 01:05:27
06.006.0000  SSR 58.189  (DB66)OBJECT/DATABRIDGE/DMSIISUPPORT/BANKDB/
DB66 ON HOST     Tuesday, April 23, 2019 @ 11:38:10
06.006.0000  SSR 58.189  (DB66)OBJECT/DATABRIDGE/DMSIISUPPORT/TESTDB/
TESTDB ON HOST    Wednesday, April 10, 2019 @ 17:26:03
06.006.0001 SSR 58.189 (DB66)OBJECT/DATABRIDGE/LICENSEMANAGER ON HOST
Saturday, December 15, 2018 @ 01:01:00
06.006.0000 SSR 58.189 (DB66)OBJECT/DATABRIDGE/LICENSESUPPORT ON HOST
Saturday, December 15, 2018 @ 01:01:04
06.006.0000 SSR 58.189 (DB66)OBJECT/FILEBRIDGE/READER/COMS ON HOST
Saturday, December 15, 2018 @ 01:06:04
06.006.0000 SSR 58.189 (DB66)OBJECT/FILEBRIDGE/READER/BICSS ON HOST
Saturday, December 15, 2018 @ 01:06:09
06.006.0000 SSR 58.189 (DB66)OBJECT/FILEBRIDGE/READER/SUMLOG ON HOST
Saturday, December 15, 2018 @ 01:05:59
06.006.0000 SSR 58.189 (DB66)OBJECT/FILEBRIDGE/READER/TTRAIL ON HOST
Saturday, December 15, 2018 @ 01:06:07
06.006.0000 SSR 58.189 (DB66)OBJECT/FILEBRIDGE/READER/LINCLOG ON HOST
Saturday, December 15, 2018 @ 01:06:02
06.006.0000  SSR 58.150  (DB66)OBJECT/FILEBRIDGE/READER/BANKFILE ON
HOST         Saturday, December 15, 2018 @ 01:05:57
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/READER/DISKFILE ON
HOST         Saturday, December 15, 2018 @ 01:05:55
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/READER/PRINTFILE ON
HOST        Saturday, December 15, 2018 @ 01:05:53
```

```
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/READER/USERDATAFILE ON
HOST       Saturday, December 15, 2018 @ 01:06:14
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/COMS ON
HOST       Saturday, December 15, 2018 @ 01:06:04
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/BICSS ON
HOST       Saturday, December 15, 2018 @ 01:06:09
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/SUMLOG
ON HOST      Saturday, December 15, 2018 @ 01:05:59
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/TTRAIL
ON HOST      Saturday, December 15, 2018 @ 01:06:07
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/LINCLOG
ON HOST      Saturday, December 15, 2018 @ 01:06:02
06.006.0000  SSR 58.150  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/BANKFILE
ON HOST      Saturday, December 15, 2018 @ 01:05:57
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/DISKFILE
ON HOST      Saturday, December 15, 2018 @ 01:05:55
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/
PRINTFILE ON HOST    Saturday, December 15, 2018 @ 01:05:53
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/SAMPLE/READER/
USERDATAFILE ON HOST   Saturday, December 15, 2018 @ 01:06:14
06.006.0000 SSR 58.189 (DB66)OBJECT/FILEBRIDGE/INITIALIZE ON HOST
Saturday, December 15, 2018 @ 01:05:48
06.006.0000 SSR 58.189 (DB66)OBJECT/FILEBRIDGE/PATCHDASDL ON HOST
Saturday, December 15, 2018 @ 01:05:47
06.006.0000 SSR 58.189 (DB66)OBJECT/FILEBRIDGE/COBOLTODASDL ON HOST
Saturday, December 15, 2018 @ 01:05:49
06.006.0000  SSR 58.189  (DB66)OBJECT/FILEBRIDGE/USERDATATODASDL ON
HOST         Saturday, December 15, 2018 @ 01:06:11
End of DBInfo report
```

Displaying the Current AFN

Follow these steps to display the current (active) AFN for a specific DMSII database:

1. Start DBInfo in normal mode by typing the following command. You must include the quotes.

   ```
   START WFL/DATABRIDGE/DBINFO ("databasename", "AFN" [,
   logicaldatabasename])
   ```

   where *databasename* indicates the title of the DESCRIPTION file without the DESCRIPTION node.

2. The WFL displays the current AFN. Since the current audit file number is returned in the DBInfo TASKVALUE attribute, you can alter the WFL to use that value in any way you see fit.

Displaying the Current Database Update Level

Follow these steps to display the current database update level for a specific DMSII database:

1. Start DBInfo in normal mode by typing the following command. You must include the quotes.

   ```
   START WFL/DATABRIDGE/DBINFO ("databasename", "LEVEL" [,
   "logicaldatabasename"])
   ```

where *databasename* indicates the title of the DESCRIPTION file without the DESCRIPTION node.

2 The WFL displays the current update level. Since the current update level is returned in the DBInfo TASKVALUE attribute, you can alter the WFL to use that value in any way you see fit.

Retrieving the Number of Sections in an Audit File

See the REMOVEAUDITFILE subroutine in WFL/ DATABRIDGE/SAMPLE/REMOVEAUDIT for an example of how to determine how many sections an audit file has.

Run DBInfo in Interactive Mode

You can run DBInfo in interactive mode to find the first quiet point in an audit trail from a specified AFN and ABSN and use that information (that is, AFN, ABSN, SEG, and INX) to configure Databridge Span or a Databridge Client after a rollback. (Run DBInfo in normal mode to determine the current AFN and ABSN.)

In interactive mode, DBInfo uses a remote file interface.

To run DBInfo in interactive mode

1 Start DBInfo in interactive mode by typing the following command. You must include the quotation marks.

```
START WFL/DATABRIDGE/DBINFO ("databasename", "parameter [ON
diskfamilyname] " [, "logicaldatabasename"])
```

where *databasename* indicates the title of the DESCRIPTION file without the DESCRIPTION node, *parameter* is one of the following parameters that initiate interactive mode, and *diskfamilyname* is the pack DBInfo will use for reading the audit files (the default is the primary audit pack specified in the DASDL).

All of the following parameters initiate interactive mode, so use the parameter that is easiest for you to remember:

```
FINDQPT
FIND
QPT
FIND QPT
FIRSTQPT
FIRST
FIRST QPT
```

2 When prompted, type the AFN and the ABSN in the following format. The ABSN must be a value of two or greater. The ABSN does not have to be in the specified audit file. DBInfo will stop at the first QPT that has an ABSN greater than or equal to the specified ABSN.

NOTE: If you do not know the ABSN, enter 2 and DBInfo will use the first quiet point in the specified file.

<*AFN*>space<*ABSN*>

3 Press transmit.

DBInfo returns the AFN, ABSN, SEG, INX, and time stamp for the first quiet point after the specified AFN and ABSN.

For example:

```
Quiet point found on December 9, 2009 @ 09:12:17 at audit location
AFN = 18 ABSN = 17090 SEG = 9 INX = 10
Please enter <AFN> space <ABSN> or ?END
```

4 To search for another quiet point, type the AFN and the ABSN and then press transmit. To quit DBInfo, type ?END and then press transmit.

Example

The following example uses a sample database named BANKDB:

```
START WFL/DATABRIDGE/DBINFO ("(USER)BANKDB ON HOST","QPT")
```

The following prompt appears:

```
Please enter <AFN> space <ABSN> or ?END
```

Enter the AFN and ABSN to specify the starting point of the search. Using the example in step 2, you would enter:

```
18 17090
```

DBInfo returns the following:

```
Searching from AFN 18 ABSN 17090 ...
Quiet point found on December 9, 2009 @ 09:12:17 at audit location
AFN = 18 ABSN = 17090 SEG = 9 INX = 10
Please enter <AFN> space <ABSN> or ?END
```

DBInfo WFL

You can modify the DBInfo WFL for the following:

- STARTTIME
- QUEUE—If you want the program to enter the system through job queue 10, you must modify the WFL to include the QUEUE (or CLASS) = 10 declaration.
- BDNAME—This is the default location where the printer backup files are created.

11 Chapter 11: DMSII Reorganizations and Rollbacks

This chapter discusses how to accommodate reorganizations (page 213) and rollbacks (page 213) on the DMSII database.

In this Chapter

Prepare for a DMSII Reorganization

Use this procedure before you reorganize your DMSII database.

To prepare for a DMSII reorganization

1. Run WFL/DATABRIDGE/BACKUPTAILORED to create copies of the DESCRIPTION file and DMSUPPORT library with the update level as the last node in the file titles. This ensures that Databridge has access to the previous layout information (in the old DESCRIPTION file) to correctly process the audit trail files created before the reorganization. For instructions, see "Save with Update Level (WFL)" on page 179.
2. Make the necessary changes to the DASDL and then compile the new DASDL, using the method you normally use at your site.

   When the DASDL is finished compiling, a message states that a database reorganization is required. For example, if the database is BANKDB and the update level is 51, the DASDL compile creates the file DESCRIPTION/BANKDB with update level 52. (The file DMSUPPORT/BANKDB with update level 52 is also created if the DASDL is set to compile DMSUPPORT automatically.)
3. Compile DMSUPPORT if it was not set to compile automatically when you compiled the DASDL. The DMSUPPORT compile creates the file DMSUPPORT/BANKDB with the next update level (52 in our example).
4. Run WFL/DATABRIDGE/BACKUPTAILORED to back up the new DESCRIPTION/BANKDB and DMSUPPORT/BANKDB files with the next update level in the filename.
5. Generate the reorganization program using BuildReorg, choosing global control options (INQUIRYONLY, EXCLUSIVE, OFFLINE, and USEREORGDB), as needed.

IMPORTANT: If you use USEREORGDB, be aware that RSNs in the records that USEREORGDB creates will change when the new records are merged into the live database. This creates problems in the client database when records get updated, if the RSNs are used as keys. Starting in DMSII 55.1 (MCP 14.0), the BuildReorg GENERATE statement lets you specify the option KEEPRSN, which prevents these RSNs from changing.

For example,

```
GENERATE EMPLOYEES KEEPRSN;
```

6 Compile and run the DMSII reorganization program.

The reorganization closes the current audit file and opens a new one. Then, it copies the old DESCRIPTION and DMSUPPORT files before it installs the new DMSUPPORT library.

7 If you have a tailored Support Library, update the GenFormat parameters and custom VIRTUAL and/or ALTER data set code, as needed.

8 If you use Databridge Span for replication, perform the procedure in "Complete a DMSII Reorganization" on page 200.

If you use the Databridge Client, see Chapter 11, "DMSII Reorganizations and Rollbacks" in the *Databridge Client Administrator's Guide*.

Complete a DMSII Reorganization

If you use Databridge Span for replication, follow this procedure after a DMSII database reorganization occurs.

To complete a DMSII reorganization

1 Complete the steps in "Prepare for a DMSII Reorganization" on page 199, which include creating backup copies of the current DESCRIPTION file and DMSUPPORT library.

2 Run Databridge Span. When it detects the reorganization, Databridge Span stops and displays the following message:

```
Databridge Engine: >> [0020] Table reorganization required for
datasetname; DMS dataset level = updatelevel, Table level =
clienttablelevel <<
```

If the DESCRIPTION/*databasename* file has a different update level than the audit file, DBEngine tries to read one of the following files (in the order shown):

- DESCRIPTION/*databasename/audit_file_update_level*
- *audit_file_update_level*/DESCRIPTION/*databasename*

If the DMSUPPORT library has a different level than the audit file, DBEngine tries to read one of the following files (in the order shown):

- *normal_DMSUPPORT_title/audit_file_update_level*
- *audit_file_update_level/normal_DMSUPPORT_TITLE*

Continuing with the previous example, when you run Databridge Span, it might try to process an audit file with update level 51. If so, Databridge automatically tries to read DESCRIPTION/BANKDB/51 or 51/DESCRIPTION/BANKDB and tries to link to DMSUPPORT/BANKDB/51 or 51/

DMSUPPORT/BANKDB to determine how to process that audit file. When Databridge finishes processing audit files with update level 51, it searches for a DESCRIPTION file and DMSUPPORT file with update level 52.

3. Process the Databridge Span output files as you usually do. For example, you may transfer them to the client system. This prevents from having data (if record sizes are equal) or an /OLDATTS node appended to the filenames of your old output (if record sizes are not equal) the next time you run Databridge Span.
4. Prepare the client system to process files corresponding to the new format for the reorganized data sets. The changes you make will depend on how the client is set up at your site.
5. In the Databridge Span parameter file, identify the reorganized data sets which will now have a mode of 3, and do the following:
   - **5a** Change the mode from 3 (indicating reorganization) to 2 (indicating “ready to update”).
   - **5b** Change the client format level to match the DMSII format level.
6. Run Databridge Span as usual.

   The reorganized records will appear in the Databridge Span output files.

Preparing for a DMSII Rollback

You can prepare for possible DMSII rollbacks by backing up Databridge Span parameter files after each replication run. If you typically let Databridge Span append data to existing data files from previous runs, you will also need to back up the data files at the end of each run such that they can be matched with any particular parameter file backup. This allows you to easily reposition the Databridge Span parameter file to a commit point prior to the restore point on the primary database.

If you anticipate having to roll back up to three days of transactions from your database, save backup copies of Span Accessory parameter files for the last three days that you ran it. Databridge Span stores audit location information in its parameter file and the information is updated each time you run it.

Consider saving the parameter files in one of the following ways:

- If you run Span several times a day:

```
DATA/SPAN/databasename/CONTROL/001
DATA/SPAN/databasename/CONTROL/002
DATA/SPAN/databasename/CONTROL/003
```

- If you run Span once a day:

```
DATA/SPAN/databasename/CONTROL/FEBRUARY4
DATA/SPAN/databasename/CONTROL/FEBRUARY5
DATA/SPAN/databasename/CONTROL/FEBRUARY6
```

Recover from a DMSII Rollback

Use the following procedure to recover from a DMSII rollback in Databridge Span. This procedure requires that you routinely back up your Databridge Span parameter file. See “Preparing for a DMSII Rollback” on page 201. If you don't have these files, use the procedure in “Manual Recovery from a Rollback” on page 202.

To recover from a DMSII rollback

1. Read the DMSII rollback report to determine the audit location of the rollback point. The report will indicate the AFN (audit file number) and ABSN (audit block serial number) of the rollback point, or earlier.

   Alternatively, if you run Databridge Span before reloading an older parameter file, DBEngine will detect an audit location mismatch and try to find the rollback point in the audit trail. If it is successful, Databridge Span will receive the result code 0120 `Databridge Engine: Database rolled back to AFN=afn ABSN=absn Seg=seg Inx=inx timestamp`
2. Restore the Databridge Span parameter file that corresponds to the rollback point, or earlier.
3. Remove any Databridge Span output files that were created after the rollback point.
4. If Databridge Span appends updates to data files created by previous replication runs, you must copy the data files from the backup you made matching the parameter file you loaded in step 2.
5. Run Databridge Span as usual.

   If you get an error similar to either of the following, an audit discontinuity has occurred. See the next procedure, “Manual Recovery from a Rollback” on page 202.

   ```
   Databridge Engine: >> [0033] tablename: Audit location mismatch,
   subtype = value is wrong. Check for DMS rollback <<
   Databridge Engine: >> [0092] Expected ABSN=nnnn in AUDITnnn at segment
   nnnn but found ABSN=mmmm <<
   ```

Manual Recovery from a Rollback

Use this procedure to manually recover the Databridge Span parameter and output files after a DMSII rollback occurs. This procedure is appropriate for situations where a copy of the Databridge Span parameter file prior to the rollback point is unavailable. Databridge cannot continue replication until the audit locations in the parameter file are manually corrected using the following instructions.

To manually recover from a rollback

1. “Run DBInfo in Interactive Mode” on page 196.
2. When prompted, enter the AFN and ABSN of the rollback point and record the information that appears. For example:

   ```
   Quiet point found on January 30, 2010 @ 09:12:17 at audit location
   AFN = 18 ABSN = 17090 SEG = 9 INX = 10
   ```

If Databridge Span sometimes appends updates to data files created by previous replication runs, you must use an AFN and ABSN corresponding to the last time the data files were created from scratch prior to the rollback point. For example, if the rollback point is AFN 18 and ABSN 17090 but the last time the output files were created from scratch was at AFN 17 and ABSN 15885, you need to enter AFN 17 and ABSN 15885 in the Database Info utility.

3 Get the Databridge Span parameter file.

4 In the replication status information, enter the values from the DBInfo interactive report for each data set:

CAUTION: Take precautions to enter the information from the DBInfo interactive report correctly. If the numbers aren't entered correctly, valid updates may be skipped. For example, if the AFN is 100, make sure that you enter `100` and not `102`. Entering `102` would prevent you from getting updates from audit files 100 and 101.

- From the Info interactive report, enter the AFN value for File Nbr.
- From the Info interactive report, enter the ABSN value for Aud Block SerialNbr.
- From the Info interactive report, enter the SEG value for Segment Number.
- From the Info interactive report, enter the INX value for Word Index.
- Change the date and time values to all zeros. (If you omit this, the Span Accessory will fail with an invalid timestamp error or 0033, which indicates a corrupt audit file because of the invalid timestamp.)
- Change the Mode to 2 if it is not already 2. (If you know that the value was something else, use that value; in most cases, however, the value will be 2. A value of 2 indicates that the data set is ready to be updated.)

An example of the pertinent columns in the replication status information follows.

```
%Data  Rec  File  Aud Block  Segment   Word   Date  Time  Mo- Format-Lvl
%-set Type   Nbr  SerialNbr   Number  Index YYYYMMDDhhmmss de DMS Client
%----  ---  ----  ---------  -------  ----- -------------- -  ----  ----
% .
% .
% .
%00003  000  0018 0000017090  0000009  00010 0000000000000 2 00000 00000
```

5 When you have finished making the changes to each data set, save the Span Accessory parameter file.

6 Remove all of the current Databridge Span output files to prevent the new updates from appending on to the current (yet now obsolete) files.

7 Run Databridge Span as usual.

CAUTION: If the entered audit location does not exactly match the secondary database, or if you do not know the exact state of the secondary database, there is a possibility of record duplication or loss.

A Appendix A: Troubleshooting

This appendix contains important troubleshooting information.

In this Appendix

General Troubleshooting Procedure

Complete these steps if you have problems using Databridge.

To troubleshoot Databridge

1. Check to see that your system meets the minimum hardware and software requirements necessary to use the product. For a list of system requirements, see the *Databridge Installation Guide*.
2. Check your system. You may be using peripheral equipment or other software that is not be compatible with this product.
3. Check the usercodes for your DMSII databases and the usercode for the Databridge files. Make sure that the Databridge software can access the DMSII DESCRIPTION, CONTROL, DMSUPPORT, and audit files.
4. Check parameter file options for the Databridge Accessory you are using. Make sure all tailored support library transforms, filters, and formats are also entered into the GenFormat parameter file and are spelled correctly.
5. Resolve errors. Check the mainframe console ODT for messages from any Databridge Accessory or utility. Look for Databridge printer files in the DBBD/RUN/= directory. Also check the Databridge Client logs, if used. If you are receiving error messages or status messages that you don't understand, see the *Databridge Message and Error Guide* included with the product documentation.
6. If you cannot identify and solve the problem without assistance, contact your product distributor. Call from a location where you have access to the problem mainframe.
7. Troubleshoot the problem using information available from Technical Support.

   http://support.attachmate.com/techdocs/

   This service directly links you to our internal help desk system, 24 hours a day, 7 days a week.
8. Contact Technical Support:

   http://support.attachmate.com/contact/

Troubleshooting for All Accessories

Problem	Solutions
An Accessory cannot find DBEngine	Most likely the Accessory is running under a different usercode than where the DBEngine is located. If this is unavoidable, use the SL command to define the DBEngine function name, as follows (assuming DBEngine is located under the DBA usercode on a pack called DBAPACK): `SL DBENGINE = (DBA)OBJECT/DATABRIDGE/ENGINE ON DBAPACK`
An Accessory displays a message indicating a version mismatch and then terminates	Most likely the Databridge software you are running was compiled using different versions of the Databridge API (SYMBOL/DATABRIDGE/INTERFACE). Make sure that all of the Databridge software is from the same release.
A task called DATABRIDGE/MISSING_AUDIT_FILE displays the title of a missing file and then waits for an accept.	A necessary audit file is missing. Restore the audit file to the proper disk family and then enter the AX command. (No additional text is required after the AX.) This will tell DBEngine to retry the audit file search. You cannot do an FA to redirect the search to another disk family. Alternatively, if the file cannot be restored, DS the waiting task (DATABRIDGE/MISSING_AUDIT_FILE). DBEngine will terminate the current operation with an error.
A worker displays the following message: `Possible AUDIT JOB 'runaway zipper' problem` `OK to start jobs for AFN` *`firstAfn`* `thru` *`lastAfn`* The Worker is then suspended in the Waiting Entries.	This message can occur in the rare event that DBServer would zip more than 10 AUDIT JOB jobs one after the other, and it is intended to trap a bug that has been difficult to reproduce. You can either enter an OK to continue or a DS to terminate the Worker, but please do a DS ARRAYS and send the program dump to technical support.

NOTE: Most programs, including DBEngine and DBSupport, will display their compile versions if you simply run them directly from CANDE. (The exception is DBGenFormat because it requires a Boolean parameter.) This information can be helpful in determining which patches a program has.

Outdated Filters and Formats

When a filter or format is out-of-date, Databridge Accessories attempt to recompile the Support Library. If that is unsuccessful, the Accessory displays an error message informing you that the support library must be recompiled. Use WFL/DATABRIDGE/COMP to recompile a tailored support library.

B Appendix B: Compiling Programs

In this Appendix

WFL/DATABRIDGE/COMP

WFL/DATABRIDGE/COMP is used to compile any of the following programs:

- DBSupport library, either tailored or nontailored. For information on using WFL/DATABRIDGE/COMP to compile a support library, filtering routine, or formatting routine, see "Understanding GenFormat" on page 46.
- DBDMSIISupport library
- DBCobolSupport library
- SAMPLE/SQLGEN
- SAMPLE/DASDLGEN
- SAMPLE/EXTRACTADDRESS
- SAMPLE/COBOLGEN
- SAMPLE/AUDITCLOSE
- SAMPLE/REFORMAT
- SAMPLE/READDOC

Before you start WFL/DATABRIDGE/COMP, make sure that all of the parameters are UPPERCASE.

CAUTION: Make sure that WFL/DATABRIDGE/INCLUDE/SSRTITLES has the correct system software titles and locations for the current release.

Compiler Control Card Options

As with all other Databridge WFLs, you can set QUEUE and STARTTIME. In addition, this WFL/DATABRIDGE/COMP contains compiler control cards ($ card options) for each Databridge component.

The compiler control cards provide you with a convenient way of changing settings such as $ RESET LIST and $ SET MERGE LINEINFO ERRORLIST. However, most sites do not need to change any of the compiler control card options.

Resulting Files

If there are any compile errors or warnings when you run WFL/DATABRIDGE/COMP, they are written to the following file:

ERRORS/programname

where *programname* is the Databridge component you are trying to compile.

In addition, any printer listings that the compile generates are written to the following directory:

DBBD/COMP/*programname*

Patches

If you want to add a user-written filter or format to DBSupport, use the INTERNAL FILTER or INTERNAL FORMAT options to include the patch file.

See “Declaring Internal and External Formats” in “Creating a Format” on page 66.

For other custom patches, you can modify the source directly, or you can put the patch in the appropriate CARD file in WFL/DATABRIDGE/COMP. We recommend that you put the patch in the CARD file and that you use the $INCLUDE statement, as shown in the following example:

```
DATA CARD/SUPPORT
$ RESET LIST STATISTICS (RESET LABELS) RESET VERBOSE
$ SET MERGE LINEINFO ERRORLIST VERSION 4.0
$ INCLUDE "PATCH/DATABRIDGE/SUPPORT/STARTUP" 76100000
```

NOTE: The position of the dollar sign ($) for the INCLUDE statement must be in column 2 or beyond. The space between the $ and the literal INCLUDE does not matter. The line number, 76100000 in this example, must begin in column 73.

Glossary of Terms

ABSN. Audit block serial number—The audit block serial number is a 10-digit number that identifies an audit block.

absolute address (AA) value. AA is a DMSII term that stands for absolute address. An absolute address value is an A Series WORD (48-bits in length). In the Databridge Client, AA is the hexadecimal representation (12 character strings containing the characters 0–9 and A–F) of the AA Value on the host. Databridge Client uses the AA Values to implement unique keys for the parent structures of embedded data set records. It also uses AA Values to reference the records of data sets that do not have DMSII SETS with the NO DUPLICATES ALLOWED attribute.

AA Values are not constant. Any DMSII reorganization (record conversion, file format, or garbage collection) changes these values.

Databridge Client supports numeric AA Values that are stored as NUMBER(15) in Oracle, BIGINT in SQL Server, and DECIMAL(15) in DB2. It also supports binary AA Values that are stored as RAW(6) in Oracle and BINARY(6) in SQL Server.

Accessories. Databridge Accessories access the services in DBEngine and DBSupport. Some of the Accessories provided with Databridge are as follows:

- DBServer, which provides communication and DMSII database replication services to Databridge Clients.
- DBSpan, which produces a replication of one or more data sets into flat sequential disk files. DBSpan updates the cloned flat files by appending the changes to the end of the flat files (unlike DBSnapshot, which replaces the changed records).
- DBSnapshot, which produces a one-time replication of one or more data sets into flat sequential disk files or tape.
- DBInfo, which produces a report of your DMSII database timestamps, update levels, DMSII release levels, etc.
- DBLister, which produces a report of the layout of the structures in your DMSII database, including structure numbers and key sets.
- DBAuditTimer, which closes the current audit file when it is older than a specified length of time.

Accessroutines. The Accessroutines program is a DMSII library program that controls access to the database, reads and writes records, and creates the audit trail.

AFN. The audit file number is a four-digit number that identifies an audit file.

audit file. An audit file is created by the DMSII Accessroutines and contains the raw format of changes made to the DMSII database by update programs. Audit file records contain the deletes, creates, and modifies that were made to the various structures. Depending on the frequency of changes made to a database, the information in an audit file can span a few hours or several weeks.

Databridge uses the audit file for the raw data of each database change to exactly replicate the primary database. Databridge records the audit location (AFN, ABSN, SEG, IDX) between runs, so it can restart without losing any records.

audit trail. The audit trail contains all of the audit files generated for a database. The Databridge Engine reads the audit files to extract updates. It then passes the updates to the Client to be applied to the relational database. After the updates have been successfully extracted, the Client saves the state information, which includes the location in the audit trail from which the last group of updates for the data set were read.

Batch Console. The Batch Console automates routine Client tasks by allowing command files/shell scripts launched by the Client service to interact with the service.

caching. A process that filters files before they're requested by the Databridge Client. Caching allows Databridge Enterprise Server to send Client data requests quickly and without placing an additional resource burden on the mainframe.

client. The client is the computer system that will receive DMSII records from the primary database. The client could be a Windows computer, a UNIX computer, or an MCP server. The client can have a relational or a DMSII database.

cloning. Cloning is the one-time process of generating a complete snapshot of a data set to another file. Cloning creates a static picture of a dynamic database. Databridge uses the DMSII data sets and the audit trail to ensure that the cloned data represents a synchronized snapshot of the data sets at a quiet point, even though other programs may be updating the database concurrently. Databridge clones only those data sets you specify.

Cloning is one phase of the database replication process. The other phase is tracking (or updating), which is the integration of database changes since the cloning.

consolidated file. A file created by Databridge Span that contains all replicated records from various data sets.

CONTROL file. The DMSII CONTROL file is the runtime analog of the DESCRIPTION file. The DESCRIPTION file is updated only when you compile a modified DASDL. The CONTROL file controls database interlock. It stores audit control information and verifies that all database data files are compatible by checking the database timestamp, version timestamp, and update level. The CONTROL file is updated each time anyone opens the database for updates. The CONTROL file contains timestamps for each data set (when the data set was defined, when the data set was updated). It contains parameters such as how much memory the Accessroutines can use and titles of software such as the DMSUPPORT library (DMSUPPORT/databasename).

Databridge uses the CONTROL file for the following information:

- Timestamps
- INDEPENDENTRANS option
- AFN for the current audit file and ABSN for the current audit block
- Data set pack names
- Audit file pack name
- Database user code

DASDL. Data and Structure Definition Language (DASDL) is the language that defines DMSII databases. The DASDL must be compiled to create a DESCRIPTION file.

data set. A data set is a file structure in DMSII in which records are stored. It is similar to a table in a relational database. You can select the data sets you want to store in your replicated database.

Databridge Director. Databridge Director (also referred to as DBDirector) is a Windows Service installed with Enterprise Server that starts Enterprise Server whenever a connection request is received.

When you start your computer, DBDirector starts and reads the ListenPort registry value to determine which TCP/IP port communicates with Databridge Clients.

Databridge Engine. The Databridge Engine (also referred to as DBEngine) is a host library program that uses the DMSII Support Library to retrieve data records from the DMSII database for cloning.

Databridge Server. Databridge Server (also referred to as DBServer) is a Databridge Host Accessory that responds to Databridge Client requests for DMSII data or DMSII layout information and provides communications between the following components:

- DBEngine and Databridge Enterprise Server
- DBEngine and the Databridge Client

NOTE: When Enterprise Server is used with the Databridge Client, Enterprise Server takes over much of the functionality of DBServer and DBEngine.

DBClient. A Client program that is launched by the Client Manager service. DBClient handles the processing of DMSII data and updates the same as dbutility, except that it runs as a background run and uses the Client Console to display its output and interact with the operator.

DBClntCfgServer. A program that handles all requests from the Client Console specific to a data source. These requests include updating the client configuration file, providing access to the client control tables, and handling the Client Configurator. Like DBClient, this program is run by the Client Manager service as a background run.

DESCRIPTION file. The DESCRIPTION file contains the structural characteristics of a database, physically and logically. This file is created from the DASDL source by the DASDL compiler and contains the layout (physical description), timestamp, audit file size, update level, logical database definition, and any static information about the database. It contains information about the database, not the data itself.

There is only one current DESCRIPTION file for each DMSII database. Databridge must have access to the DESCRIPTION file before it can replicate a database. Additionally, Databridge uses the DESCRIPTION file information for consistency checks between the primary database and the secondary or replicated database.

The DESCRIPTION file corresponds to the schema in a relational database.

direct disk. A replication method that allows Databridge Enterprise Server to clone and track DMSII data sets without using any significant mainframe resources. Direct disk replication requires a SAN (Storage Area Network) or Logical Disks configured to make MCP disks visible in Windows.

DMSII Support. DMSII Support is a Databridge library that retrieves data records from the DMSII database for cloning. The Databridge Engine links to this library to perform database functions such as reading records, switching the audit file, and getting database statistics.

entry point. A procedure in a library object.

extraction. Extraction is the process of reading through a data set sequentially and writing those records to a file (either a secondary database or flat file).

file format conversion. A type of DMSII reorganization affects file size values (for example, AREASIZE, BLOCKSIZE, or TABLESIZE), but it does not change the layout of the records in a DMSII database.

filler substitution. A DMSII filler substitution is a technique for avoiding a reorganization. It changes record layouts, but does not move records around.

fixup records. Changes that occur to the DMSII database while a clone is taking place.

flat files. A flat file is a plain text or mixed text and binary file which usually contains one record per line. Within the record, individual fields may be separated by delimiters, such as commas, or have a fixed length and be separated by padding. An example of a flat file is an address list that contains fields for *Name* and *Address*.

garbage collection reorganization. A garbage collection reorganization moves records around, but it doesn't change the layout of the DMSII database. Its primary function is to improve disk and/or I/O efficiency by eliminating the space occupied by deleted records. Optionally, a garbage collection reorganization reorders the remaining records in the same sequence as one of the sets.

GenFormat. A host utility that creates translation, filter, and format routines. The GenFormat utility interprets the GenFormat parameter file to generate ALGOL source code patches, which are included in the tailored Support Library.

lag time. The lag time is defined as the elapsed time between the time a record in the DMSII database is updated and the time where this update appears in the relational database. This value accounts for any difference between the clock on the mainframe and that on the client machine.

Lister Accessory. A Databridge Host Accessory that produces a report of the layout of the structures in your DMSII database, including structure numbers and key sets.

mutex. A mutex is an operating system resource that is used to implement a critical section and prevent multiple processes from updating the same variables at the same time.

null value. The value defined in the DASDL to be NULL for a data item. If the DASDL does not explicitly specify a NULL value for a data item, the NULL value is all bits turned on.

primary database. This is the original DMSII database that resides on the host. Databridge replicates from the primary database to one or more client databases. The client databases can be another DMSII database or one of several relational databases. Compare this to the replicated (or secondary) database.

quiet point (QPT). A quiet point is a point in the audit trail when the DMSII database is quiet and no program is in transaction state. This can occur naturally, or it can be forced by a DMSII sync point.

record format conversion. A type of DMSII reorganization that occurs when a data set or set (group of keys) is reordered or reformatted. It indicates that changes were made to a data set format, or to data items, such as changing the length of an item, for example, BANK-ID NUMBER (10) to BANK-ID NUMBER (15).

record serial number (RSN). Record sequence numbers (RSN) are 48-bit quantities used by the Databridge Engine, in the case of DMSII XE, to uniquely identify a record. RSNs will always be used instead of AA Values when available except for data sets having embedded data sets. RSNs are always static; they will not change after a garbage collection reorganization.

reorganization. Structural or formatting changes to records in the DMSII database, which may require parallel changes to (or re-cloning of) records in the secondary, or relational, database. See also file format conversion and record format conversion.

replicated database. The replicated database is the database that usually resides on the client machine and contains records cloned from the DMSII database. The replicated database is updated periodically with changes made to the primary (original) DMSII database. Compare this to the primary database.

replication. Replication is the ongoing process of cloning and tracking changes to a DMSII database.

rollback. A systematic restoration of the primary or secondary database to a previous state in which the problem or bad data is no longer found.

secondary database. The replicated database. The replicated database is the database that usually resides on the client machine and contains records cloned from the DMSII database. The replicated database is updated periodically with changes made to the primary (original) DMSII database. Compare this to the primary database.

semaphores. Operating system resources that are mainly used to implement thread synchronization and signaling.

service. The service (Windows) or daemon (UNIX) that automates most Client operations. It handles operator requests from the Client Console and routes all log and informational messages to the consoles.

set. An index into a data set. A set has an entry (key + pointer) for every record in the data set.

state information. Data that reflects information about the cloned data, such as the audit location and format level.

structure. A data set, set, subset, access, or remap. Each structure has a unique number called the structure number.

subset. An index into a data set. A subset does not necessarily have an entry (key + pointer) for every record in the data set. Subsets are used to access selected members of a data set and to represent relationships between data set records. Subsets typically contain fewer entries than normal sets.

An automatic subset is any subset that contains a WHERE clause and is maintained by DMSII.

A manual subset is any subset that is maintained by an application.

Support Library. A library that provides translation, formatting, and filtering to the DBServer and other Accessories. After DBServer receives data from the Databridge Engine, it calls the Support Library to determine if the data should be replicated, and if so, passes the data to the Support Library for formatting.

system library. A library code file registered with the MCP on the host that is associated with a function name. Programs can link to the library code file by specifying the function name.

table. A data structure in the client database corresponding to a data set or remap in the host DMSII database.

tracking. Tracking is an ongoing process for propagating changes made to records in the DMSII primary database to the replicated database after the initial clone. The Databridge Engine performs extraction as well as tracking.

undigits. A NUMBER data item containing bit values from 10 to 15 in one or more digits. The digits in a NUMBER data item should contain values from 0 to 9; however, it is possible for the digits in NUMBER data item to contain values 0 to 15. Because values 10 to 15 are not valid digit values, the digits in NUMBER data items containing values from 10 to 15 are called undigits.