

Administrator's Guide
Databridge FileXtract



Attachmate Databridge™

Databridge FileXtract

Version 6.3



© 2015 Attachmate Corporation. All rights reserved.

No part of the documentation materials accompanying this Attachmate software product may be reproduced, transmitted, transcribed, or translated into any language, in any form by any means, without the written permission of Attachmate Corporation. The content of this document is protected under copyright law even if it is not distributed with software that includes an end user license agreement. The content of this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Attachmate Corporation. Attachmate Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this document.

Attachmate and the Attachmate logo are registered trademarks and Databridge is a trademark of Attachmate Corporation in the USA.

All other trademarks, trade names, or company names referenced herein are used for identification only and are the property of their respective owners.

A copy of the Attachmate software license agreement governing this product can be found in a 'license' file in the root directory of the software distribution.

Third party notices (if any) can be found in a 'thirdpartynotices' file in the root directory of the software distribution.

Attachmate Corporation
705 5th Ave South, Suite 1100
Seattle, WA 98104 USA
+1.206.217.7100
<http://www.attachmate.com>

Contents

About This Guide	7
Abbreviations	7
Conventions	7
Related Documentation	8
Introducing FileXtract	9
Introducing FileXtract	9
Advantages of FileXtract	10
Getting Started with FileXtract	11
Installing FileXtract	13
System Requirements	13
Before You Install	13
Installing FileXtract	16
Using the Sample Reader Libraries	19
Creating a Custom Reader Library	19
Sample Reader Libraries	20
SUMLOG Reader Library	20
TTRAIL Reader Library	21
PRINTFILE Reader Library	21
BICSS Reader Library	22
DISKFILE Reader Library	22
LINCLOG Reader Library	23
BANKFILE Reader Library	24
USERDATA Reader Library	24
TEXT Reader Library	24
Using the COBOL-to-DASDL Utility	25
Understanding the COBOL-to-DASDL Utility	25
BANKFILE COBOL-to-DASDL Example	25
Generating Database Layout from COBOL 01-Level FileRecord Descriptions	27
Generating Database Layout from LINC Activity Log Files	29
Troubleshooting	33
Out-of-Order Processing	35
Before You Process	35
Process Archived SUMLOGs	35
Process Current SUMLOGs	36

Databridge Products and their Components	37
Databridge Host	37
Databridge Enterprise Server	38
Databridge Client	39
Databridge DMSII Client	40
Databridge FileXtract	40
DataBridge Plus	40
DataBridge Twin	40
Glossary of Terms	41

About This Guide

This guide contains instructions for installing, configuring, and using Attachmate Databridge FileXtract (hereafter referred to as FileXtract). 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)
- File layouts and the description of those layouts for the files you will be replicating

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

Conventions

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

This convention or term	Is used to indicate this
<code>this type style</code>	text that you type filenames and directory names onscreen messages
<i>italic</i>	variables emphasis document titles
square brackets ([])	optional items in a command For example, [true false]. (Do not type the brackets.)
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
<i>MCP server</i> <i>host</i> <i>mainframe</i> (terms)	Unisys ClearPath NX, LX or A Series mainframe
<i>DBEngine</i> (term)	Databridge Engine
<i>DBEnterprise</i> (term)	Databridge Enterprise Server
<i>DBServer</i> (term)	Databridge Server

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 Attachmate 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.

CHAPTER 1

Introducing FileXtract

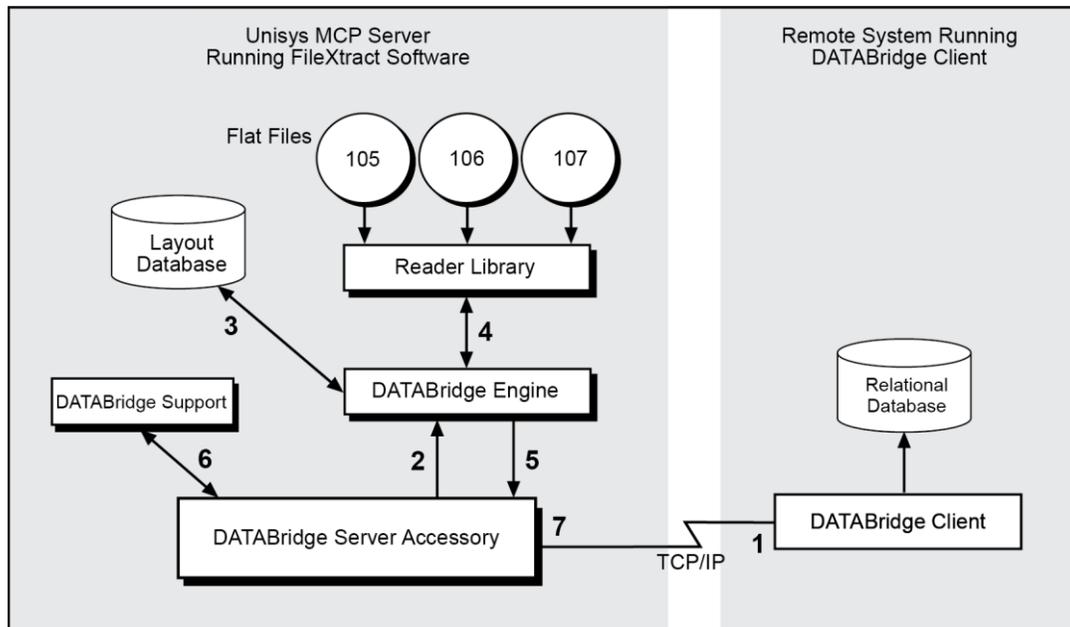
In this Chapter

Introducing FileXtract	9
Advantages of FileXtract	10
Getting Started with FileXtract	11

Introducing FileXtract

Databridge FileXtract is a product which allows you to clone and update non-DMSII data (that is, flat files) residing on Unisys ClearPath NX, LX, or A Series mainframes. FileXtract is bundled with the Databridge Host software and includes several Reader libraries and other associated files.

You can use FileXtract in conjunction with the Databridge Client, DBSpan, or DBSnapshot. When using FileXtract with the Databridge Client, FileXtract data sources resemble any other DMSII data source. The following diagram shows how the Databridge Client uses FileXtract to replicate flat files.



1. The Databridge Client calls the Databridge Server (DBServer) to replicate the specified flat files.
2. DBServer calls the Databridge Engine (DBEngine) to extract the data for the selected flat files.

3. The DBEngine accesses the specified FileXtract layout database to determine the layout of the flat files.

No updates are actually done to this database. The Engine gets all of the layout information it needs from the DESCRIPTION file. The DMSII CONTROL file and the DMSUPPORT library will exist, but the data sets and audit files will not.

4. DBEngine calls the appropriate Reader library, as specified in the layout database, to extract the data from the flat files.
5. DBEngine sends the data to DBServer.
6. DBServer calls Databridge Support to filter, alter, or transform the data, if needed.
7. DBServer sends the data to the Databridge Client.
8. The Databridge Client populates the relational database and then either waits for additional flat file information or terminates.

Advantages of FileXtract

FileXtract provides the following advantages:

- Any type of flat file (non-DMSII data) can be replicated to a Client database.
 - Sample Reader libraries are provided for system summary log files (SUMLOG), COMS Transaction Trail files (TTRAIL), printer backup files (PRINTFILES), BICSS log files (BICSS), LINC Activity logs (LINCLOG), and the USERDATA system file. You can use these sample Reader libraries without any modifications to the Reader libraries. DASDL files are provided to generate the corresponding layout database. For more information, see Chapter 3, "Using Reader Libraries."
 - The DISKFILE Reader library is provided to replicate COBOL created flat files. If you have COBOL 01-level file record descriptions for your flat files, you can use the COBOL-to-DASDL utility to generate the layout database. For more information, see Chapter 4, "Using the COBOL-to-DASDL Utility."
 - Sample Reader libraries are also provided to allow you to customize replication of any other type of flat file. For more information, see Chapter 3, "Using Reader Libraries."
- Cloning is required only one time. After the flat files are cloned, FileXtract updates the Client database as new records and flat files become available.
- Replicate only the parts you want from flat files. You don't have to replicate the entire flat file.
- By copying flat files to a secondary database, you offload decision support, queries, and reporting from the primary database. The secondary database provides a secure way to make data available to selected individuals, departments, or sites while protecting the flat files on the host.
- The data is available on the Client system even if the host is down or the data communication connection is broken. This eliminates long wait times for data availability. Users can use any database tool available on the Client system to access the data in the secondary database.

Note: FileXtract cannot track updates to flat file records. If record modifies or deletes occur, you must reclone.

Getting Started with FileXtract

The following is a description of how to use FileXtract to replicate flat files.

- 1 Install FileXtract. See Chapter 2, “Installing FileXtract.”
- 2 Decide which flat files you want to replicate.
- 3 Select a Reader library to use from the following table, and then select or generate a layout database for the flat files you want to replicate.

For these types of files	Use this Reader library
System summary log files	SUMLOG
COMS Transaction Trail files	TTRAIL
Printer backup files	PRINTFILE
BICSS log files	BICSS
Flat file using a COBOL FD	DISKFILE
LINC Activity logs created by LINC systems	LINCLOG
System USERDATA file	USERDATA
Flat files that contain information in a “proprietary” format (for example, a flat file that requires reading the nth word to find the address of the next record)	A custom, user-written Reader library
The BANKFILE sample flat file supplied with FileXtract	BANKFILE

- 4 Start the replication process.
- 5 If you will be using DBServer, define the SOURCE and READER options in the DBServer parameter file. If you will be using Databridge Span or Snapshot, define the READER option in the appropriate parameter file.

Refer to the *Databridge Host Administrator’s Guide* for instructions.

- 6 Run the accessory (DBServer, DBSpan, or DBSnapshot). Refer to the *Databridge Host Administrator’s Guide* for instructions on configuring and running Databridge accessories.
- 7 If you are replicating to the Databridge Client, run the Databridge Client DEFINE command against the SOURCE option in Databridge Server.

Refer to the *Databridge Client Administrator’s Guide* for instructions on running the Databridge Client.

CHAPTER 2

Installing FileXtract

In this Chapter

System Requirements	13
Before You Install	13
Installing FileXtract	16

System Requirements

FileXtract has the following requirements:

- Unisys MCP-hosted server running MCP level SSR 49.1 to 55.1 and DMSII software 49.1 to 55.1 (including the DMALGOL compiler)
- Access to the flat files
- Flat files must meet the following criteria:
 - Have the same layout and a common naming convention
 - Consist of only creates added to the end of the file, not modifications to existing records
 - Flat files in a group must have the same Reader type (interface). For example, all are SUMLOG files

Before You Install

If the Databridge host database replication software is installed on the mainframe where you want to install FileXtract, it is recommended that you install FileXtract to the same usercode and pack as Databridge.

Caution: The FileXtract installation WFL overwrites the Databridge host database replication software files listed in this section; therefore, if you made changes to any of these files, write down the changes before you start the FileXtract installation WFL. When the installation WFL finishes, update the new files to match the settings in the previous ones

DBSupport

If you changed SYMBOL/DATABRIDGE/SUPPORT, note the changes you made and make the corresponding changes to the new SYMBOL/DATABRIDGE/SUPPORT.

Databridge Compile WFL

If you changed WFL/DATABRIDGE/COMP, note the changes you made and make the corresponding changes to the new WFL/DATABRIDGE/COMP.

Option	Your Setting
QUEUE=	
STARTTIME=	
BDNAME=	
Other:	

Other Databridge WFL Files

In all of the WFL/DATABRIDGE files, you can modify the following:

- STARTTIME
- QUEUE
- Name of printer backup files

If you changed any of these items, write your changes in the tables that follow. After installation, update the files accordingly.

WFL/DATABRIDGE/SPAN

Option	Your Setting
QUEUE=	
STARTTIME=	
BDNAME=	

WFL/DATABRIDGE/LISTER

Option	Your Setting
QUEUE=	
STARTTIME=	
BDNAME=	

WFL/DATABRIDGE/SERVER

Option	Your Setting
QUEUE=	
STARTTIME=	
BDNAME=	

USERCODE=

WFL/DATABRIDGE/SNAPSHOT

Option	Your Setting
QUEUE=	
STARTTIME=	
BDNAME=	

WFL/DATABRIDGE/DBINFO

Option	Your Setting
QUEUE=	
STARTTIME=	
BDNAME=	

SAMPLE Files

If you did not rename SAMPLE files before you changed them, save them with a different name now. The SAMPLE files are as follows:

DATA/ENGINE/SAMPLE/CONTROL
 DATA/GENFORMAT/SAMPLE/CONTROL
 DATA/LISTER/SAMPLE/CONTROL
 DATA/SERVER/SAMPLE/CONTROL

DBEngine Control File

The DATA/ENGINE/CONTROL file is overwritten with the new Databridge key. If you made changes to the parameter file, make a note of them now.

Option	Your Setting
KEY or EVALKEY	
Audit Level =	
Property Level =	
Checkpoint client...	
Workers =	
Available From...To... =	
ReadAhead =	
Print Statistics =	

Links =

DBPlus =

DMSII program titles

Mirrored Audit

Installing FileXtract

Complete the following steps to install FileXtract.

- 1 Determine the usercode. Installing to a privileged usercode prevents most problems running the installation WFL.

Caution: If the Databridge host database replication software is installed on the mainframe where you want to install FileXtract, see [Before You Install](#) (page 13).

If you plan to replicate flat files

Then install FileXtract

Under the same usercode where you run FileXtract

Under the same usercode from which you plan to run it.

Under two or more usercodes

With no usercode (that is, nonusercoded).

If you don't install FileXtract as nonusercoded, establish the DBEngine as a system library (SL).

If you don't establish DBEngine as an SL, you must put a copy of DBEngine under every usercode where you will run FileXtract, or you must file equate each time you run FileXtract.

- 2 Choose the primary or secondary pack from the usercode's FAMILY substitution statement.

For example, if the usercode has the following substitution

```
FAMILY DISK = PRODPK OTHERWISE DISK
```

store the FileXtract software on either PRODPK or DISK

- 3 Copy the installation WFL from the installation image or DVD to the host.

- If you install from the DVD, use the following command:

```
WFL UNWRAP *WFL/DATABRIDGE/INSTALL AS  
WFL/DATABRIDGE/INSTALL OUTF *INSTALL FROM DBXX  
(DVD) TO DISK (RESTRICTED = FALSE)
```

where XX is the version of the software with no decimals (for example, DB62).

- If you install from uploaded files, upload DISKINSTALL and IMAGE files using a file transfer tool capable of binary/image transfers to the usercode where Databridge is to be installed. Sign on to the usercode and then use the following command:

```
WFL UNWRAP *WFL/DATABRIDGE/INSTALL AS WFL/DATABRIDGE/
INSTALL OUTOF DISKINSTALL
```

4 Run the installation WFL.

- a) From CANDE or another editor, start the installation WFL as follows:

```
START WFL/DATABRIDGE/INSTALL ("FILEEXTRACT"
[, "familyname"])
```

where *familyname* is the optional name of the pack you determined in step 2 and is only required if you did not install the WFL to the primary family. (If you don't include the *familyname* parameter, the installation defaults to DISK.)

- b) Enter the access code (license key) or, if using an evaluation copy, the evaluation code located on the Databridge host DVD package.
- c) If the installation WFL pauses, complete the remaining steps. Otherwise, skip the remaining steps.
- d) If you installed from DVD to a nonprivileged usercode and the installation WFL is paused, you must unrestrict OBJECT/DATABRIDGE/KEYENTRY, as in this example:

```
RESTRICT -FILE (ENGR)OBJECT/DATABRIDGE/KEYENTRY ON PRODUCTION
```

- e) After you have unrestricted OBJECT/DATABRIDGE/KEYENTRY, transmit `mixnumberOK` to the install WFL job.

OBJECT/DATABRIDGE/KEYENTRY now runs.

- f) When prompted, enter the access code (same as license key) located on your Databridge host DVD package.
- g) After the installation WFL completes, mark all of the Databridge object files as unrestricted by entering the following from either the ODT or from a privileged usercode in MARC:

```
RESTRICT -FILE objectfilename
```

Databridge Accessories and other programs will need to specify a location for the Databridge Engine each time they access it, unless you establish the Engine as a System Library (SL).

CHAPTER 3

Using the Sample Reader Libraries

In this Chapter

Creating a Custom Reader Library	19
Sample Reader Libraries	20

FileXtract includes several sample Reader libraries, which are configured to replicate flat files as standard data sets for the given file type. You can typically use the sample Reader libraries without modifications, except in the following circumstances:

- Your flat files contain data in a proprietary format
- New fields have been added to a standard layout
- You want to replicate additional types of records that the Reader library does not support (for example, a new SUMLOG record type)

The Reader libraries work in conjunction with a physical database that DBEngine uses to determine the layout of the flat files. DBEngine gets the information it needs from the DESCRIPTION file for the layout database. The DMSII CONTROL file and the DMSUPPORT library will exist, but the data sets and audit files will not.

A DASDL file is provided for each sample Reader library so that you can create the corresponding layout database. For example, if you want to replicate SUMLOGs, you would use the SYSFILESDB DASDL to create a physical database called SYSFILESDB. Within this database is a logical database called SUMLOG that contains typical layout information for SUMLOG files. If needed, you can modify this information. If you customize the layout information in the logical database, you must make the corresponding changes in the Reader library.

Creating a Custom Reader Library

You must create a custom Reader library in either of the following situations:

- The flat files that you want to replicate contain information in a “proprietary” format (for example, a flat file that requires reading the n th word to find the address of the next record).
- The sample Reader library does not meet your needs. For example, you want to replicate additional types of SUMLOG records.

The easiest way to create a custom Reader library is simply to customize one of the sample Reader libraries to meet your needs. If this is not possible, complete the following steps to create a custom Reader library.

To create a custom library

- 1 Describe the file layouts in either DASDL or COBOL FD format.

Note: If you use COBOL FD format, use the COBOL-to-DASDL utility to generate the DASDL. For instructions, see [Generating Database Layout from COBOL 01-Level File Record Descriptions](#) (page 27).

Append a VERIFY clause to each data set that indicates when a flat file record is considered part of that data set. The DASDL must group the data sets into a logical database, as follows:

```
Logical_dbname ["reader [; file_spec]"] DATABASE (dataset_list)
```

where *reader* is the title of the Reader library object code and *file_spec* is a file or directory containing the flat files. Note that the semicolon is required before the *file_spec*, if present.

- 2 Write the Reader library source code in ALGOL or COBOL. If you are writing the Reader library in ALGOL, use the TEXT, SUMLOG, or TTRAIL Reader library as an example. If you are writing the Reader library in COBOL, use the BANKFILE Reader library as an example.

The procedure called in the Reader library must be named FileRead. Use DBFileReaderHead to define the procedure in ALGOL. Use \$FEDLEVEL =5, \$ SET LIBRARYPROG, and set PROGRAM-ID to FILEREAD for COBOL. FileRead is passed two arrays—the FileInfo structure and FileRecord for the resulting record image.

- 3 Perform required initialization the first time FileRead is called. This might include linking to DBEngine, getting the structure index for the data sets, identifying the location of the flat files, etc.

Note: Linking to DBEngine, whether explicit or implicit, must not occur before the first call of FileRead.

- 4 Configure the Reader library to read the file, filter records, copy the data to the FileRecord array, and set items in FileInfo such as the structure index, file record location, timestamp, etc.

Set the appropriate structure index, as defined in the *Databridge Programmer's Reference*, in the FileInfo array to determine the data set to which a record belongs. Note that by setting the structure index to 0 in the FileInfo array, you can rely on the DASDL VERIFY clauses to determine the data set to which a record belongs; however, this is not the preferred method because it is slower.

Note: In addition to filtering and formatting done by the Reader library, DBSupport can also provide filtering and formatting. See the *Databridge Host Administrator's Guide* for more information about DBSupport.

Sample Reader Libraries

SUMLOG Reader Library

This Reader library reads system summary log files and returns the data records to DBEngine. This Reader library uses the SUMLOG logical database of the SYSFILESDB layout database.

Closed SUMLOG files are typically titled:

```
*SUMLOG/sss/mddy/nnnnn ON familyname
```

where *ssss* is the system serial number, *mmdyy* is the date in month-day-year format, and *nnnnn* is a sequential number.

The Reader option in the Accessory's parameter file should specify the directory and family name where the closed log files are located. The following is an example of the DBServer SOURCE declaration for this Reader library:

```
Source SUMLOG:
% -----
Database = SUMLOG of DESCRIPTION/SYSFILESDB ON DISK,
Support = OBJECT/DATABRIDGE/SUPPORT/SYSFILESDB ON DISK,
Filter = SUMLOGFILTER,
Reader using "SUMLOG/= ON DISK"
```

If the Reader option is missing or empty, the SUMLOG Reader library will use the following:

```
"*SUMLOG/ssss/= ON sumlogpack"
```

where *ssss* is the serial number of the system it is running on and *sumlogpack* is the pack containing *SYSTEM/SUMLOG. When all of the closed SUMLOGs have been processed, the Reader library will automatically switch to reading the active SUMLOG.

TTRAIL Reader Library

This Reader library reads COMS Transaction Trail files and returns the data records to DBEngine. This Reader library uses the TTRAIL logical database of the SYSFILESDB layout database.

TTRAIL files are typically titled:

```
*COMS/TTRAIL/dbname/nnnn ON familyname
```

where *dbname* is the database that participates in synchronized recovery and *nnnn* is a sequential number. The TTRAIL for all non-synchronized recovery programs has a database name of TPLIBRARY.

The Reader option in the accessory parameter file must specify the directory and family name where the TTRAIL files for a particular database are located. The following is an example of the DBServer SOURCE declaration for this Reader library:

```
Source TTRAIL:
% -----
Database = TTRAIL of DESCRIPTION/SYSFILESDB ON
DISK,
Support = OBJECT/DATABRIDGE/SUPPORT/SYSFILESDB
ON DISK,
Filter = TTRAILFILTER,
Reader using "COMS/TTRAIL/= ON DISK";
```

PRINTFILE Reader Library

This Reader library reads printer backup files and returns the printer file attributes and print lines to DBEngine. This Reader library uses the PRINTFILE logical database of the SYSFILESDB layout database.

The Reader option in the Accessory's parameter file should specify the directory and family name where the print files are located. The following is an example of the DBServer SOURCE declaration for this Reader library:

```
Source LISTER:
% -----
```

```
Database = REPORTS of DESCRIPTION/PRINTFILESDB ON
DISK,
Reader using "DBBD/RUN/LISTER/= ON DISK";
```

If the Reader option is missing or empty, the PRINTFILE Reader library will use:

```
"*BD ON printerbackuppack"
```

where *printerbackuppack* is the value identified by issuing a DL BACKUP command.

BICSS Reader Library

This Reader library reads BICSS log files and returns the data records to DBEngine. This Reader library uses the UVMSBICSS or IVRBICSS logical database of the NAPFILESDB layout database.

BICSS log files are typically titled:

```
(usercode)BICSSLOG/ACTIVITY/yymmdd/hhmnss ON familyname
```

where *yymmdd* is the date in year-month-day format and *hhmnss* is the time in hours-minutes-seconds format.

The Reader option in the accessory parameter file must specify the directory and family name where the log files are located. Optionally, it can also specify the particular application system name that must appear in all records by appending a colon and the LINC application name, such as IVRSYS (see the example below) or UVMS. If you want records from only one application and the log contains records from more than one application, specifying the application name after the colon speeds up the processing.

The following is an example of the DBServer SOURCE declaration for this Reader library:

```
Source IVRBICSS:
% -----
Database = IVRBICSS of DESCRIPTION/NAPFILESDB ON
DISK,
Support = OBJECT/DATABRIDGE/SUPPORT/NAPFILESDB ON
DISK,
Filter = BICSSFILTER,
Reader using "(BICSS)BICSSLOG/ACTIVITY/= ON DISK:
IVRSYS";
```

DISKFILE Reader Library

This Reader library reads flat files and returns the data records to DBEngine. When you use the COBOL-to-DASDL utility to create the various structures needed to replicate flat files that use COBOL 01-level file record descriptions, as explained in [Generating Database Layout from COBOL 01-Level File Record Descriptions](#) (page 27), the DISKFILE Reader library is the default Reader library.

The DISKFILE Reader library processes disk files in creation timestamp order. It assumes that the file it is reading can be extended as long as there is no file with a later creation timestamp. During processing, the creation timestamp of the disk file is used as the timestamp associated with each record. If you need to process a flat file having a creation timestamp prior to what FileXtract has already processed, see [Out-of-Order Processing](#) (page 35).

The Reader option in the accessory parameter file must specify the directory and family name where the disk files are located. The following is an example of the DBServer SOURCE declaration for this Reader library:

```
Reader using "(PROD)DATA/TRANLOG/= ON USERPACK"
```

If you want to change this behavior, you may modify the DISKFILE Reader library or substitute your own Reader library. If you use a name other than BJECT/FILEBRIDGE/READER/DISKFILE, make sure to specify the title of the Reader library code file in the READER "*codefiletitle*" USING "... " option of the accessory parameter file.

Some programs, such as CANDE and FileXpress®-XST, rewrite the entire file when records are added, deleted, or modified. The (modified) file will have a new creation timestamp. When the DISKFILE Reader searches the disk directory for files to process, it will find the file with a new creation timestamp and will attempt to process it from the beginning. This is likely to cause many "duplicates" errors because it will send each record as an "add" and all of the records from the old file are already present in the client database. To avoid these errors, you must configure the client to reclone every time it is run.

Similarly, if the mainframe application updates records in place, the DISKFILE Reader will not detect the update if it has already processed that file. In this situation, you must also configure the client to reclone every time it is run.

LINCLOG Reader Library

This Reader library reads LINC Activity logs and returns the data records to DBEngine.

LINC log files are typically titled:

```
(usercode)lincsystem/LINCLOG/nnn ON familyname
```

where *nnn* is the Activity log number.

The Reader option in the accessory parameter file should specify the directory and family name where the closed log files are located.

To generate the LINCLog database so that Databridge can replicate the log records to the client database, perform the following steps.

- 1 Edit the DATA/COBOLTODASDL/SAMPLE/LINCLOG/FD file, which describes the layout of the LINC Activity Log records, to reflect your particular LINC system. (Different LINC implementations sometimes have new fields added to the log record layout.)
- 2 Save the file as DATA/COBOLTODASDL/LINCLOG/FD, i.e., without the SAMPLE node.
- 3 Edit the DATA/COBOLTODASDL/SAMPLE/LINCLOG/CONTROL file with the proper location for the LINC log files.
- 4 Save the file as DATA/COBOLTODASDL/LINCLOG/CONTROL, again without the SAMPLE node.
- 5 Use the COBOL-to-DASDL utility to generate the database:

```
START WFL/FILEBRIDGE/COBOLTODASDL ("LINCLOG")
```

This also generates a file called DATA/SERVER/LINCLOG/CONTROL that you can insert into DATA/SERVER/CONTROL to define the LINCLog SOURCE.

- 6 If you want to filter the log records, declare a filter and generate a tailored DBSupport library using DBGenFormat.
- 7 Use the Databridge Client or an accessory such as DBSpan to replicate the LINC Log data just like any other database.

The following is an example of the DBServer SOURCE declaration for this Reader library:

```
Source "LINCLOGFILES":
% -----
```

```
Database = "LINCLOGFILES" OF
"DESCRIPTION/LINCLOG",
Reader using "(PROD)G2/LINCLOG/= ON AUDIT";
Reader processes the disk files in the LINC log file number order (with
wraparound from 999 to 1).
```

BANKFILE Reader Library

This Reader library is a sample flat file Reader library that demonstrates how to write a FileXtract Reader in COBOL. For an example of how to use this Reader library to replicate a sample flat file called DATA/LOAD/SAMPLE/DATABASE, see BANKFILE COBOL-to-DASDL Example (page [25](#)).

USERDATA Reader Library

This Reader library reads the system USERDATA file layout and returns the usercode attributes to DBEngine. It interprets the structure of the system USERDATA file, which contains attributes associated with usercodes. Because the layout of the USERDATA file can vary from site to site, this Reader library generates a DMSII database layout corresponding to your site-specific USERDATA file.

The USERDATA Reader library generates the following:

- The corresponding DASDL layout for a DMSII database for FileXtract
- The corresponding source code for the USERDATA Reader library

To generate and compile the DASDL layout and the USERDATA Reader library, do the following:

```
START WFL/FILEBRIDGE/COMP ("USERDATA")
```

The Reader option in the accessory parameter file should specify the directory and family name where the USERDATA file is located. The following is an example of the DBServer SOURCE declaration for this Reader library:

```
Source USERDATA:
% -----
Database = USERFILE of DESCRIPTION/USERDATADB
Support = OBJECT/DATABRIDGE/SUPPORT/USERDATADB
Filter = USERDATAFILTER
%- Reader using "SYSTEM/USERDATAFILE/othersystem ON
DISK";
```

If the Reader option is missing or empty, the USERDATA Reader library will use:

```
*SYSTEM/USERDATAFILE ON userdatapack
```

where *userdatapack* is disk location specified by the DL USERDATA command.

TEXT Reader Library

This is a skeleton of a Reader library written in ALGOL. It is intended to be used when writing a custom ALGOL Reader library. See Chapter 5, "Creating a Custom Reader Library," for information about writing a custom Reader library.

CHAPTER 4

Using the COBOL-to-DASDL Utility

In this Chapter

Understanding the COBOL-to-DASDL Utility	25
BANKFILE COBOL-to-DASDL Example	25
Generating Database Layout from COBOL 01-Level FileRecord Descriptions	27
Generating Database Layout from LINC Activity Log Files	29

This chapter explains how to use the COBOL-to-DASDL utility to replicate flat files that use COBOL 01-level file record descriptions.

Understanding the COBOL-to-DASDL Utility

The COBOL-to-DASDL utility allows you to generate a FileXtract layout database from COBOL 01-level file record descriptions. It also generates the appropriate DBServer SOURCE declaration that you insert into the DBServer parameter file so that Databridge Clients can access the replicated flat files.

- For normal disk files, the DISKFILE Reader library is used to read the flat files that are being replicated. For a description of the DISKFILE Reader library, see DISKFILE Reader Library (page [22](#)). Otherwise, see Generating Database Layout from COBOL 01-Level File Record Descriptions.

Note: You must have or create a COBOL file containing the 01-level file record descriptions for the flat files you want to replicate. A sample COBOL file (DATA/COBOLTOASDL/SAMPLE/LINCLOG/FD) containing a layout of the LINC log is installed with the FileXtract software.

- For LINC log activity files, the LINCLOG Reader library is used to read the flat files that are being replicated. For a description of the LINCLOG Reader library, see LINCLOG Reader Library (page [23](#)). Otherwise, see Generating Database Layout from LINC Activity Log Files.

For an example of how to use the COBOL-to-DASDL utility to replicate a sample flat file called DATA/LOAD/SAMPLE/DATABASE, see the BANKFILE COBOL-to-DASDL Example that follows. We recommend that you complete this example before using the COBOL-to-DASDL utility for your own disk files.

BANKFILE COBOL-to-DASDL Example

This example illustrates how to use the COBOL-to-DASDL utility to replicate a sample flat file called DATA/LOAD/SAMPLE/DATABASE. Everything you need to run this example is installed with FileXtract.

- 1 View DATA/LOAD/SAMPLE/DATABASE and familiarize yourself with the records in this flat file.
- 2 View and familiarize yourself with DATA/COBOLTODASDL/SAMPLE/BANKFILE/FD, which is the COBOL FD for DATA/LOAD/SAMPLE/DATABASE.
- 3 Copy the BANKFILE parameter file as follows:

```
GET DATA/COBOLTODASDL/SAMPLE/BANKFILE/CONTROL AS  
DATA/COBOLTODASDL/BANKFILE/CONTROL
```

- 4 Define the SOURCE.

The SOURCE name will be used as the DBServer SOURCE name as well as the name of logical database that the COBOL-to-DASDL utility creates. Note that this step has already been completed for you in this sample control file, as follows:

```
DEFINE Source COBOLBankFile USING "DATA/LOAD/SAMPLE/DATABASE" FROM FD IN  
"DATA/COBOLTODASDL/SAMPLE/BANKFILE/FD";
```

- 5 Declare the data sets to be generated from the various flat files. Note that this step has already been completed for you in this sample control file. The following are the data sets that have been defined:

```
Dataset Comments
```

```
uses INPUT-DATA-REC
```

```
where COMMENT-REC; % 88-level item
```

```
Dataset BANK
```

```
uses INPUT-BANK % => use the INPUT-BANK REDEFINE
```

```
where BANK-TYPE and (BANK-ID not greater than 5000
```

```
or BANK-NAME ^= SPACES); % a more complex expression
```

```
Dataset BRANCH
```

```
uses INPUT-BRANCH, DATA-TYPE
```

```
where BRANCH-TYPE;
```

```
Dataset CUSTOMER
```

```
uses INPUT-CUSTOMER
```

```
where CUSTOMER-TYPE;
```

```
Dataset TELLER
```

```
uses INPUT-TELLER
```

```
where TELLER-TYPE;
```

```
Dataset ACCOUNT
```

```
uses INPUT-ACCOUNT
where ACCOUNT-TYPE;
```

```
Dataset HISTORY
uses INPUT-HISTORY
where HISTORY-TYPE;
```

- 6 Save the parameter file.
- 7 Run the COBOL-to-DASDL utility using the following command:

```
START WFL/FILEBRIDGE/COBOLTODASDL ("BANKFILE")
```

The COBOL-to-DASDL utility processes the COBOL file description, generates a DASDL called SOURCE/FILEBRIDGE/COBOLTODASDL/BANKFILE, and generates a DBServer parameter file fragment called DATA/SERVER/BANKFILE/CONTROL. Then, the WFL compiles the resulting database and runs the FileXtract Initialize utility to prepare it for Databridge.

- 8 Update the DBServer parameter file with the generated DBServer parameter file fragment, as follows:

- a. Transmit the following command:

```
GET DATA/SERVER/CONTROL
```

- b. Transmit the following command:

```
INSERT DATA/SERVER/BANKFILE/CONTROL AT END
```

The DBServer parameter file is appended with the BANKFILE DBServer parameter file fragment, which defines the SOURCE, database, and data file title to use to replicate the BANKFILE flat file.

- 9 Run the Databridge Client with the DEFINE command to define the COBOLBankFile source. Then run it with the GENERATE command and finally with the PROCESS command. The Databridge Client contacts DBServer to replicate the BANKFILE flat file. Refer to the *Databridge Clients Administrator's Guide* for instructions on running the Databridge Client.

Generating Database Layout from COBOL 01-Level FileRecord Descriptions

COBOLTODASDL/SAMPLE/BANKFILE/CONTROL) contains sample entries based on a flat file called DATA/LOAD/SAMPLE/DATABASE. This file illustrates how to declare a DBServer SOURCE and generate the associated layout database from COBOL 01-level file record descriptions. As you complete the following steps, remove, modify, or comment out the sample entries.

To replicate flat files that use COBOL 01-level file record descriptions.

- 1 Copy the COBOL-to-DASDL parameter file as follows:

```
Get DATA/COBOLTODASDL/SAMPLE/BANKFILE/CONTROL AS DATA/
COBOLTODASDL/dbname/CONTROL
```

where *dbname* is the name you want to use for the layout database that will be generated. (See the table below.)

- 2 Define the DBServer SOURCE declaration as follows:

```
DEFINE SOURCE sourcename USING "filename"
FROM FD IN "COBOLsourcefile";
```

Where	Is
<i>sourcename</i>	the name of the logical database that the COBOL-to-DASDL utility creates. DBSpan and DBSnapshot access flat file data using the logical database will be used .
<i>dbname</i>	different than <i>sourcename</i> and <i>flatfiledirectory</i> . If you enter the same name for SOURCE and database, an error message appears.
<i>filename</i>	a file or a directory. Note that the DISKFILE Reader library treats a file as both a file and a directory. Therefore, that file and all files under that file node will be replicated.
<i>COBOLsourcefile</i>	the file that contains the 01-level file record layout(s). It cannot contain declarations other than the FD and the 01-level file record layouts (and it does not have to contain the actual FD). If the FD is in the main COBOL program source file, you must create a new COBOL file containing just the disk file layout and use that file's title for COBOLsourcefile.

In addition to using this information to create the DASDL, the COBOL-to-DASDL utility uses the SOURCE declaration to create a DBServer parameter file fragment (DATA/SERVER/dbname/CONTROL) which you insert into the DBServer control file.

- 3 Declare the data sets to be generated from the various flat files using the following syntax.

Note: When you run the COBOL-to-DASDL utility, it automatically encloses the SOURCE and logical database names in "quotation marks" in the generated parameter file fragment for DBServer. This feature allows you to use hyphens (-) in SOURCE names, and the SOURCE names can be the same as a parameter file keyword.

```
DATASET datasetname USES identifier_list [WHERE boolean_expression]
```

For example:

```
DATASET BANK USES INPUT-BANK WHERE BANK-TYPE and (BANK-ID not greater than
5000 or BANK-NAME ^= SPACES);
```

The USES clause tells the Reader library what data items to use when there are REDEFINES. Note that successive 01s under an FD are implicit REDEFINES.

The WHERE clause indicates the condition that must be true of all records that belong in that data set. The WHERE clause can be omitted if all records belong in a single data set. The WHERE clause must evaluate to a Boolean expression (true or false).

The Boolean expression can consist of relational expressions, connected by AND or OR, using the normal COBOL relational operators: GREATER THAN, >, GTR, LESS THAN, <, LSS, EQUAL TO, =, EQL, GREATER THAN OR EQUAL TO, >=, GEQ, LESS THAN OR EQUAL TO, <=, LEQ, ^=, !=, <>, NEQ. The left side of a relational expression must be a data item name and the right side must be either an integer, a double-quoted string, a hex string, SPACE, or SPACES. An 88-level condition name can also be used as a relational expression.

- 4 Save the COBOL-to-DASDL parameter file.
- 5 Run the COBOL-to-DASDL utility using the following command:

```
START WFL/FILEBRIDGE/COBOLTODASDL ("dbname")
```

where *dbname* is the name of the layout database you assigned in step 1.

The COBOL-to-DASDL utility processes the COBOL file description, generates the DASDL called SOURCE/FILEBRIDGE/COBOLTODASDL/*dbname*, and generates a DBServer parameter file fragment called DATA/SERVER/*dbname*/CONTROL. Then, the WFL compiles the resulting database and prepares it for use with FileXtract.

- 6 Do one of the following based on which Databridge accessory you are going to use:

For	Do this
DBServer	<p>Insert the DBServer parameter file fragment (DATA/SERVER/<i>dbname</i>/CONTROL) into the DATA/SERVER/CONTROL file using the following steps:</p> <ol style="list-style-type: none"> 1. Get DATA/SERVER/CONTROL. 2. Transmit the following command: <pre>INSERT DATA/SERVER/<i>dbname</i>/CONTROL AT END</pre> 3. Save the file. <p>The DBServer parameter file is appended with the DBServer parameter file fragment, which defines the SOURCE, database, and Reader library to use to replicate the flat files. After you have updated the DBServer parameter file, run the Client with the DEFINE, GENERATE, and PROCESS commands. Refer to the <i>Databridge Clients Administrator's Guide</i> for instructions.</p>
Other accessories	<p>The flat file is ready to be cloned. For DBSpan and DBSnapshot, start the associated WFL with the layout database and logical database (SOURCE) names to create the accessory parameter file. For example:</p> <pre>START WFL/DATABRIDGE/SPAN ("BANKFILE", "COBOLBANKFILE")</pre> <p>Define the READER options in the accessory parameter file and then run the Accessory. Refer to the <i>Databridge Host Administrator's Guide</i> for instructions.</p>

Generating Database Layout from LINC Activity Log Files

To generate the LINCLOG database so that Databridge can replicate the log records to the client database, follow these steps:

- 1 Edit the DATA/COBOLTODASDL/SAMPLE/LINCLOG/FD file, which describes the layout of the LINC Activity Log records, to reflect your particular LINC system. (Different LINC implementations sometimes have new fields added to the log record layout.)
- 2 Save the file as DATA/COBOLTODASDL/LINCLOG/FD,(that is, without the SAMPLE node).

```
SAVE AS DATA/COBOLTODASDL/LINCLOG/FD
```

- 3 Edit the DATA/COBOLTODASDL/SAMPLE/LINCLOG/CONTROL file so that it has the proper location for the LINC log files.

```
DEFINE SOURCE sourcename USING "directoryname"  
FROM FD IN "COBOLsourcefile"
```

Where	Is
<i>sourcename</i>	the name of the logical database that the COBOL-to-DASDL utility creates. DBSpan and DBSnapshot access flat file data using the logical database will be used .
<i>COBOLsourcefile</i>	the file that contains the 01-level file record layout(s). It cannot contain declarations other than the FD and the 01-level file record layouts (and it does not have to contain the actual FD). If the FD is in the main COBOL program source file, you must create a new COBOL file containing just the disk file layout and use that file's title for <i>COBOLsourcefile</i> .

- 4 Save the file as DATA/COBOLTODASDL/LINCLOG/CONTROL, again without the SAMPLE node.

```
SAVE AS DATA/COBOLTODASDL/LINCLOG/CONTROL
```

Use the COBOL-to-DASDL utility to generate the database and compile the Reader:

```
START WFL/FILEBRIDGE/COMP ("LINCLOGS")
```

In addition to generating the database (DASDL called SOURCE/FILEBRIDGE/COBOLTODASDL/LINCLOG), the COBOL-to-DASDL utility also generates a file called DATA/SERVER/LINCLOG/CONTROL that you can insert into DATA/SERVER/CONTROL to define the LINCLOG SOURCE.

If you want to filter the log records, declare a filter and generate a tailored DBSupport library using DBGenFormat. (See the *Databridge Host*

Administrator's Guide for instructions.) Do one of the following based on which Databridge accessory you are going to use:

For	Do this
DBServer	<p>Insert the DBServer parameter file fragment (DATA/SERVER/<i>dbname</i>/CONTROL) into the DATA/SERVER/CONTROL file using the following steps:</p> <ol style="list-style-type: none"> 1. Get DATA/SERVER/CONTROL. 2. Transmit the following command: <pre data-bbox="548 447 1089 468">INSERT DATA/SERVER/<i>dbname</i>/CONTROL AT END</pre> 3. Save the file. <p>The DBServer parameter file is appended with the DBServer parameter file fragment, which defines the SOURCE, database, and Reader library to use to replicate the flat files. After you have updated the DBServer parameter file, run the Client with the DEFINE, GENERATE, and PROCESS commands. Refer to the <i>Databridge Clients Administrator's Guide</i> for instructions.</p>
Other accessories	<p>The flat file is ready to be cloned. For DBSpan and DBSnapshot, start the associated WFL with the layout database and logical database (SOURCE) names to create the accessory parameter file. For example:</p> <pre data-bbox="521 850 1235 871">START WFL/DATABRIDGE/SPAN ("LINCLOG", "LINCLOGFILES")</pre> <p>Define the READER options in the accessory parameter file and then run the accessory. Refer to the <i>Databridge Host Administrator's Guide</i> for instructions.</p>

Troubleshooting

If you have problems using FileXtract, complete the following steps.

To troubleshoot FileXtract

- 1 Make sure that your system meets the requirements necessary to use the product. See "System Requirements" in the *Databridge Installation Guide*.
- 2 Make sure that the READER option in the parameter file has the correct Reader library and flat file location. If you are using DBServer, verify that the SOURCE declaration is pointing to the correct logical database. If you are using DBSpan or DBSnapshot, make sure that you start the WFL with the correct logical database name as the second parameter. For example, START WFL/DATABRIDGE/SPAN ("SYSFILESDB", "SUMLOG"). For information about the Reader libraries, see one of the following:
 - Sample Reader Libraries (page [19](#))
 - Generating Database Layout from COBOL 01-Level File Record Descriptions (page [27](#))
 - Generating Database Layout from LINC Activity Log Files (page [29](#))
 - Creating a Custom Reader Library (page [19](#))
- 3 Check your setup:
 - Check the mix to make sure that the Databridge accessory is running. For instructions, see the *Databridge Host Administrator's Guide*.
 - Make sure that the READER declaration for the Databridge accessory is correct. Refer to the *Databridge Host Administrator's Guide* for instructions.
 - Make sure that the source name for the DEFINE command for the Databridge Client matches the SOURCE name for DBServer.
- 4 Resolve any errors you receive.

If you are receiving error messages you don't understand, see the *Databridge Error Guide* for help resolving error these messages.
- 5 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.
- 6 Contact Attachmate Technical Support or troubleshoot the problem using information available from the Support site.

Out-of-Order Processing

Occasionally you need to load a large amount of flat file data that has been archived, and the archived data is not available in the proper sequence. For example, you might have a set of backup tapes with SUMLOGs for several days and one set of tapes in the middle of the sequence is currently “on loan.”

The SUMLOG Reader library (and other Reader libraries) rely on the timestamp from StateInfo to locate the current file and the ABSN concatenated with INX to determine the record number.

In general you want to keep two audit locations—one for the “real” (relatively current) location, and another for the archive-catch-up location. The following is one approach to handling out-of-order processing. This procedure is for processing archived log files sequentially. Processing archived log files discontinuously requires additional techniques, which are not explained in this guide.

In this section

Before You Process	35
Process Archived SUMLOGs	35
Process Current SUMLOGs	36

Before You Process

On the host, define a duplicate SOURCE in DBServer (call it SUMLOGARCHIVE) and add a STOP BEFORE *timestamp* condition to it, where *timestamp* is the timestamp of the oldest record that has already been loaded using the regular SUMLOG source. (This will prevent picking up the same files twice, including the active SUMLOG.) The SUMLOGs for this SOURCE could be in a different directory to avoid confusion.

On the client, you need the two audit locations and a way to switch them back and forth. Create a new table (AudLocSwitch) to hold the variable parts of the audit location. The variable parts are: AFN, ABSN, INX, and timestamp. Also define a text column in AudLocSwitch called “*source_name*.”

Create a row for each data set in AudLocSwitch with all zeros for the audit location and *source_name* = “SUMLOG”. Create a similar row for each data set having *source_name* = “SUMLOGARCHIVE”.

Process Archived SUMLOGs

To process archived SUMLOGs

- 1 Copy the audit locations from the DATASETS table into the AudLocSwitch rows for *source_name* = “SUMLOG”.
- 2 Copy the AudLocSwitch rows for *source_name* = “SUMLOGARCHIVE” into the data sets table.

3 Change `data_source` in the DATASOURCES table and the DATATABLES table to "SUMLOGARCHIVE".

4 Run the following command:

```
dbutility process SUMLOGARCHIVE
```

Process Current SUMLOGs

To process current SUMLOGs

1 Copy the audit locations from the DATASETS table into the AudLocSwitch rows for `source_name = "SUMLOGARCHIVE"`.

2 Copy the AudLocSwitch rows for `source_name = "SUMLOG"` into the DATASETS table.

3 Change `data_source` in the DATASOURCES table and the DATATABLES table to "SUMLOG".

4 Run the following command:

```
dbutility process SUMLOG
```

By maintaining separate audit locations and using the STOP BEFORE condition, you can continue to copy in old SUMLOGs, process them, and alternately switch back to processing the current SUMLOGs using the other SOURCE. If the old files are loaded out of order, before you can do the process command, you must manually remove the processed files and reset the audit location in the AudLocSwitch rows for "SUMLOGARCHIVE" to 0.

Databridge Products and their Components

This section lists the Databridge product family and the components included with each product.

In this section

Databridge Host	37
Databridge Enterprise Server	38
Databridge Client	39
Databridge DMSII Client	40
Databridge FileXtract	40
DataBridge Plus	40
DataBridge Twin	40

Databridge Host

The Databridge Host is installed on the mainframe. This product is required by all of the other Databridge products.

Component	Description
Databridge 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 passes the information to Databridge Server.
Databridge 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 reading the audit files on the host and sends 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.
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.
Databridge Span Accessory	Produces a replication of one or more data sets into flat sequential disk files that can be extended when more audit becomes available. Databridge Span 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 Accessory	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). Databridge Snapshot clones the selected data sets each time you run it.
Tanker Accessory	Provides filtered audit files for Databridge Span and Databridge Server.
Lister Accessory	Produces a report of the layout of the structures in your DMSII database, including structure numbers and key sets.
Info Utility	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.
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 Databridge Span each time an audit file becomes available.
Audit Close Utility	Enables you to close the current audit file.
Audit Remove Utility	Enables you to remove processed audit files that you copied to a separate audit pack.
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 <i>Databridge Programmer's Reference</i> .
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 and the Databridge COBOL API, see the <i>Databridge Programmer's Reference</i> .

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
DBEnterprise	The executable file for Enterprise Server, which is often used interchangeably with Enterprise Server.
Databridge Director (DBDirector)	A Windows Service that starts Enterprise Server whenever a data request is received.

Databridge Client

The Client initiates a connection with DBServer on the host or DBEnterprise on a Windows PC 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 dbutility, except that it runs as a background run and uses the Client Console to display its output and interact with the operator.
DBCIntCfgServer	A command-line program that implements the import and export commands for the service configuration file. It converts the service configuration file (dbcontrol.cfg) from its native binary format to text format (.ini) and back using the export and import commands.
dbutility	A program that runs the Databridge Client from a command line.
Batch Console (bconsole)	A program that allows command files, such as UNIX shell scripts, issue process-related requests to the Client Service. The Batch Console interprets (and runs) VB-like language scripts.
Client Console (installed separately)	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 DMSII Client

A mainframe program that clones and updates a DMSII database as another DMSII database. With the DMSII Client, you can selectively replicate by filtering on both row and column.

Databridge FileXtract

Databridge FileXtract is a product which allows you to clone and update non-DMSII data (that is, flat files) residing on Unisys ClearPath NX, LX, or A Series mainframes. FileXtract is bundled with the Databridge Host software and includes several Reader libraries and other associated files.

You can use FileXtract in conjunction with the Databridge Client, DBSpan, or DBSnapshot. When using FileXtract with the Databridge Client, FileXtract data sources resemble any other DMSII data source.

DataBridge Plus

A system library program that provides Databridge 5.0 and later the ability to access current DMSII audit files.

Databridge Plus applies to all Databridge accessories. That is, after Databridge Plus is established as an SL (System Library), all Accessories automatically use it, even if the accessories access different databases.

DataBridge Twin

A mainframe program that replicates a DMSII database as another DMSII database. Twin is installed on the primary DMSII database and the cloned data typically resides in a secondary database on a development or departmental mainframe. As DMSII audit becomes available, updates to the primary database are applied to the secondary database.

Glossary of Terms

A

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.
- DBTanker, which provides filtered audit files for the DBSpan and DBServer accessories.
- 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.

C

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.

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
- INDEPENDENTTRANS 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

D

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 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.

E

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).

R

replication

Replication is the ongoing process of cloning and tracking changes to a DMSII database.

S

set

An index into a data set. A set has an entry (key + pointer) for every record in the data set.

structure

A data set, set, subset, access, or remap. Each structure has a unique number called the structure number.

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.

T**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.