

Adabas Delta Save

Adabas Utility Functions for Delta Save

Version 8.1.2

June 2008

This document applies to Adabas Delta Save Version 8.1.2 and to all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © Software AG 2008. All rights reserved.

The name Software AG, webMethods and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. Other company and product names mentioned herein may be trademarks of their respective owners.

Table of Contents

1 Adabas Utility Functions for Delta Save	1
2 Adabas Utility Functions for Delta Save	3
Utility Considerations for Delta Save Operation	4
VSE/ESA Job Control for Delta Save	6
3 ADAFRM Utility	7
DSIMFRM: Format DSIM Dataset	8
DSIMRESET: Reset the DSIM Dataset	9
4 ADAFRM JCL/JCS Requirements and Examples	11
BS2000/OSD	12
z/OS	14
z/VM	15
VSE/ESA and z/VSE	17
5 ADAPRI Utility	19
ADAPRI DSIMPRI: Printing the DSIM Dataset	20
JCL/JCS Requirements and Examples	21
6 ADAREP Utility	25
Delta Save Report Format	26
Delta Save Checkpoints	28
JCL/JCS Requirements and Examples	28
7 ADARES Utility	33
COPY: Copy Sequential Protection Log	34
PLCOPY: Copy PLOG to Sequential Log	38
Rebuilding the DSIM Dataset	41
8 ADARES JCL/JCS Requirements and Examples	43
BS2000/OSD	44
z/OS	47
z/VM	49
VSE/ESA and z/VSE	51
9 ADASAV Utility	53
10 ADASAV MERGE	55
Consolidating Delta Saves or Updating Full Saves	56
Merging Online Saves and DSIM Datasets	57
Unloading DSIM Datasets	57
Using Unloaded DSIM Datasets as "Online" MERGE Input	58
Concatenated Datasets in MERGE Input	59
Syntax	60
Optional Parameters	60
Examples	62
11 ADASAV RESTORE DELTA: Restore Without Prior Merging	65
Online Save and DSIM Inputs	66
Concatenated Datasets in RESTORE DELTA Input	67
Output Database Requirements	68
Subsequent Operations	68

Syntax	68
Optional Parameters	70
Examples	78
12 ADASAV SAVE: Save Database	81
Syntax	83
Optional Parameters	83
Example	84
13 ADASAV SAVE DELTA: Save Changed Database Blocks	87
Syntax	89
Optional Parameters	89
Example	90
14 Restarting an Interrupted Save Operation	91
User ABEND 34 or 35	92
System ABEND or Other User ABEND	92
Resetting the DSIM Dataset	93
15 ADASAV JCL/JCS Requirements and Examples	95
BS2000/OSD	96
z/OS	104
z/VM	110
VSE/ESA and z/VSE	116
16 ADAULD Utility	123
UNLOAD FILE: Unload Specified File from a Save Tape	125
Save Tape Input Processing	128
ADAULD Output Processing	130
ADAULD User Exit 9	131
17 ADAULD JCL/JCS Requirements and Examples	133
BS2000/OSD	134
z/OS	135
z/VM	136
VSE/ESA and z/VSE	138
Index	141

1 Adabas Utility Functions for Delta Save

The Adabas utilities manage the resources of an Adabas database. General Adabas utility operation and use are described in the basic Adabas documentation library. The information in this documentation applies only to the new and/or expanded utility functions required for an Adabas database running with the Delta Save Facility.

This document is organized as follows:

• <i>Adabas Utility Functions for Delta Save</i>	Provides an overview of the expanded Adabas utility functions required for an Adabas database running with the Delta Save Facility.
• <i>ADAFRM Utility</i>	Describes the functions of the ADAFRM utility specific to the Delta Save Facility.
• <i>ADAFRM JCL/JCS Requirements and Examples</i>	Describes the JCL required to run the Delta Save functions of ADAFRM with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSE systems.
• <i>ADAPRI Utility</i>	Describes the functions of the ADAPRI utility specific to the Delta Save Facility.
• <i>ADAREP Utility</i>	Describes the functions of the ADAREP utility specific to the Delta Save Facility.
• <i>ADARES Utility</i>	Describes the functions of the ADARES utility specific to the Delta Save Facility.
• <i>ADARES JCL/JCS Requirements and Examples</i>	Describes the JCL required to run the Delta Save functions of ADARES with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSE systems.
• <i>ADASAV Utility</i>	Provides an overview of the functions of the ADASAV utility specific to the Delta Save Facility.
• <i>ADASAV MERGE</i>	Describes details of the MERGE function of the ADASAV utility specific to the Delta Save Facility.

• <i>ADASAV RESTORE DELTA: Restore Without Prior Merging</i>	Describes details of the RESTORE DELTA function of the ADASAV utility specific to the Delta Save Facility.
• <i>ADASAV SAVE: Save Database</i>	Describes details of the SAVE function of the ADASAV utility specific to the Delta Save Facility.
• <i>ADASAV SAVE DELTA: Save Changed Database Blocks</i>	Describes details of the SAVE DELTA function of the ADASAV utility specific to the Delta Save Facility.
• <i>Restarting an Interrupted Save Operation</i>	Explains how an ADASAV SAVE database or SAVE DELTA execution can be restarted after failure.
• <i>ADASAV JCL/JCS Requirements and Examples</i>	Describes the JCL required to run the Delta Save functions of ADASAV with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSE systems.
• <i>ADAULD Utility</i>	Describes the functions of the ADAULD utility specific to the Delta Save Facility.
• <i>ADAULD JCL/JCS Requirements and Examples</i>	Describes the JCL required to run the Delta Save functions of ADAULD with BS2000/OSD, z/OS, z/VM, VSE/ESA and /z/VSE systems.

2 **Adabas Utility Functions for Delta Save**

■ Utility Considerations for Delta Save Operation	4
■ VSE/ESA Job Control for Delta Save	6

The Adabas utilities manage the resources of an Adabas database. General Adabas utility operation and use are described in the basic Adabas documentation library. The information in this documentation applies only to the new and/or expanded utility functions required for an Adabas database running with the Delta Save Facility.

The following utility functions are described in this documentation for the Delta Save Facility:

Utility	Function	Action
ADAFRM (formatting)	DSIMFRM	Format the Delta Save Images (DSIM) dataset.
	DSIMRESET	Reset an incorrect DSIM dataset for reuse.
ADAPRI (printing)	DSIMPRI	Print DSIM dataset blocks.
ADAREP (reporting)	ADAREP	Functions expanded for Delta Save operation where applicable.
ADARES (recovery)	COPY	Copy a sequential log dataset.
	PLCOPY	Copy dual or multiple PLOG datasets to a single sequential dataset.
ADASAV (save/restore)	MERGE	Combine delta save outputs with each other, and/or with the last full database save output.
	RESTORE DELTA	Restore files/database from separate full and delta save outputs.
	SAVE	Save the complete database to a full save dataset.
	SAVE DELTA	Save the changed portions of the database to a delta save dataset.
ADAULD (unloading)	UNLOAD with SAVETAPE	Unload from a full save tape, 1-8 delta save tapes, and optionally the DSIM dataset.

The ADASAV RESTORE function is unchanged for conventional full save tapes, and can also be used to restore consolidated full save tapes that were created with ADASAV MERGE.

This chapter covers the following topics:

Utility Considerations for Delta Save Operation

In addition to the changes made to the ADASAV and ADARES utilities to allow delta save/merge/restore operations, changes have also been made to the ADAFRM, ADAPRI and ADAREP utilities.

The following points should be considered when running Adabas utilities with DSF:

- Changes made to database files by utilities (ADAINV, ADALOD, ADAORD and ADASAV RESTORE) are marked in the file control blocks (FCBs). The next Delta Save operation saves the changed parts (that is, the Data Storage, address converter and/or index) of the affected files;

- Delta Save operations are *not recommended* following ADAFRM utility runs that reset parts of the Associator or of Data Storage, since such changes are *not* logged. This means that those blocks are not *necessarily* saved by a later Delta Save operation. Therefore, any save following these operations should be a full database save.

Identifying New AC Extents Allocated During an Online Save

Every time the Adabas nucleus allocates a new address converter (AC) extent, it clears (fills with binary zeros) all blocks of that extent.

If a new AC extent is allocated during an online save, the Adabas nucleus writes a *special type of protection record* to the PLOG identifying the new extent and the blocks the nucleus has cleared. Like any online dump block, this protection record is written to the PLOG only, not to Work part 1.

The special protection record type is short and contains in its header only the file number and the first and last RABN of the newly allocated and cleared AC extent. The protection record has no data part.

ADARES PLCOPY (or COPY) recognizes the special protection record type when extracting online dump blocks from the PLOG and writing them to the DSIM dataset. For each such protection record encountered, it creates special DSIM directory entries that contain the file number and one RABN of the new AC extent. ADARES creates one directory entry for each new AC block. There are no associated DSIM detail records containing the block images, since the blocks are known to be empty (binary zeros).

When the DSIM dataset is processed for merging or unloading, the sort of the DSIM directory correctly positions the entries for new AC blocks within the sequence of all blocks and removes new AC entries made obsolete by subsequent block images written later in time.

New AC entries resulting from the merge process are substituted by empty AC blocks of the proper length. For an unload of the DSIM dataset, the correct block size cannot be determined because no GCB is present, so any block deriving from a new AC entry is constructed with zero block length. The correct block length is determined in the next ADASAV MERGE or ADASAV RESTORE DELTA operation.



Note: The ADASAV RESTONL and RESTPLOG functions ignore the special AC extent record type on the protection log and continue comparing FCBs to determine any newly allocated and formatted address converter extents.

VSE/ESA Job Control for Delta Save

The VSE/ESA sample jobs used by Adabas utilities operations assume that the procedures for defining Adabas libraries (ADAVvLIB) and Adabas files (ADAVvFIL) have been cataloged into an accessible procedure library.

For information about cataloging these procedures, see the section *Cataloging Procedures for Defining the Delta Save Libraries and Files*.

3 ADAFRM Utility

■ DSIMFRM: Format DSIM Dataset	8
■ DSIMRESET: Reset the DSIM Dataset	9

The Delta Save Facility introduces two new functions to the database formatting utility ADAFRM:

- **DSIMFRM (Format the DSIM dataset);**
- **DSIMRESET (Reset DSIM dataset blocks).**
- **JCL/JCS Requirements and Examples**

DSIMFRM: Format DSIM Dataset

Before a Delta Save Images (DSIM) dataset can be used in an online save/merge operation, it must be formatted using the DSIMFRM function.

```
ADAFRM DSIMFRM  SIZE = size  
                  [ DEVICE = device-type | ADARUN-device ]  
                  [ FROMRABN = starting-rabn ]  
                  [ NOUSERABEND ]
```

Essential Parameter

SIZE: Size of Area to be Formatted

SIZE specifies the size of the area to be formatted. Blocks (a decimal value followed by "B") or cylinders may be specified.

Optional Parameters

DEVICE: Device Type

DEVICE is the physical or logical device type to be assigned to the DSIM dataset. If FROMRABN is also specified, DEVICE must specify the device type of the existing DSIM dataset. If DEVICE is not specified, the device type specified by the ADARUN DEVICE parameter is used.

FROMRABN: Starting RABN

FROMRABN specifies the RABN at which formatting is to begin. This parameter may only be used for an existing dataset.

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Example

```
ADAFRM DSIMFRM SIZE=20,DEVICE=8381
```

A DSIM dataset is to be formatted. It is 20 cylinders large. Its device type is to be 8381.

DSIMRESET: Reset the DSIM Dataset

The DSIMRESET function resets the specified blocks of the Delta Save Images (DSIM) dataset to binary zeros. DSIMRESET can be used to reinitialize the DSIM dataset for further use. If some exceptional condition left a DSIM dataset ineligible for use by an online save operation, the dataset can be made available again using the DSIMRESET function. This can be achieved by resetting only the first block (RABN 1).

Software AG recommends that you specify the DSIM dataset for exclusive use by the ADAFRM utility to avoid accidentally destroying information in a DSIM dataset currently in use by another utility.

```
ADAFRM DSIMRESET FROMRABN = starting-rabn  
                SIZE = size  
                [ DEVICE = device-type | ADARUN-device ]  
                [ NOUSERABEND ]
```

Essential Parameters

FROMRABN: Starting RABN

FROMRABN specifies the RABN at which the resetting is to begin. This parameter must be specified.

SIZE: Size of Area to be Reset

SIZE specifies the size of the area to be reset. Either cylinders (a numerical value alone) or blocks (a value followed by "B") can be specified. The SIZE parameter must be specified.

Optional Parameters

DEVICE: Device Type

DEVICE is the physical or logical device type of the DSIM dataset. If DEVICE is not specified, the device type specified by the ADARUN DEVICE parameter is used.

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Example

```
ADAFRM DSIMRESET SIZE=1B, FROMRABN=1
```

A DSIM dataset is to be reset, making it again usable for online save/merge operation.

4 ADAFRM JCL/JCS Requirements and Examples

▪ BS2000/OSD	12
▪ z/OS	14
▪ z/VM	15
▪ VSE/ESA and z/VSE	17

This section describes the job control information required to run the Delta Save functions of ADAFRM with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSE systems, and shows examples of each of the job streams.

This chapter covers the following topics:

BS2000/OSD

Dataset	Link Name	Storage	More Information
Delta Save images (DSIM)	DDDSIMR1	disk	Required for DSIM... functions
ADAFRM parameters	SYSDTA/ DDKARTE		
ADARUN parameters	SYSDTA/ DDCARD		<i>Adabas Operations</i>
ADAFRM messages	SYSLST/ DDDRUCK		<i>Adabas Messages and Codes</i>
ADARUN messages	SYSOUT/ DDPRINT		<i>Adabas Messages and Codes</i>

Format DSIM Dataset (ADAFRM DSIMFRM)

```
/ . FRM                LOGON

/ MOD - JOB - OPTIONS   LISTING=YES

/ MOD - TEST            DUMP=YES

/ ASS - SYSLST          DO . PRI . LST

/ ASS - SYSOUT          DO . PRI . OUT

/ SET - JOB - STEP

/ ASS - SYSDTA          *SYSCMD

/ SET - FILE - LINK     DDDSIMR1, ADABAS . DB010 . DSIM
, SUP=DISK( SHARE - UPD=NO)

/ SET - FILE - LINK     DDLIB      , ADABAS . MOD

/ REMARK

/ START - PROGRAM       FROM - FILE=*MOD( ADABAS . MOD, ADARUN )

ADARUN    PROG=ADAFRM, ...
```



```
ADAFRM DSIMFRM SIZE=20,...
```

```
/ASS-SYSLST      *PRIM
/ASS-SYSOUT      *PRIM
/ASS-SYSDTA      *PRIM
/LOGOFF          SYS-OUTPUT=DEL
```

Reset DSIM Dataset (ADAFRM DSIMRESET)

```
/.RESET          LOGON
/MOD-JOB-OPTIONS LISTING=YES
/MOD-TEST        DUMP=YES
/ASS-SYSLST      DO.PRI.LST
/ASS-SYSOUT      DO.PRI.OUT
/SET-JOB-STEP
/ASS-SYSDTA      *SYSCMD
/SET-FILE-LINK   DDDSIMR1, ADABAS.DB010.DSIM
, SUP=DISK(SHARE-UPD=NO)
/SET-FILE-LINK   DDLIB    , ADABAS.MOD
/REMARK
/START-PROGRAM   FROM-FILE=*MOD(ADABAS.MOD, ADARUN)
ADARUN  PROG=ADAFRM, ...
ADAFRM DSIMRESET FROMRABN=1,SIZE=1B
/ASS-SYSLST      *PRIM
/ASS-SYSOUT      *PRIM
/ASS-SYSDTA      *PRIM
/ASS-SYSIPT      *PRIM
```

```
/LOGOFF          SYS-OUTPUT=DEL
```

z/OS

Dataset	DD Name	Storage	More Information
Delta Save images (DSIM)	DDDSIMR1	disk	Required for DSIM... functions
ADAFRM parameters	DDKARTE		
ADARUN parameters	DDCARD		<i>Adabas Operations</i>
ADAFRM messages	DDDRUCK		<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT		<i>Adabas Messages and Codes</i>

Format DSIM Dataset (ADAFRM DSIMFRM)

```
//FORMAT    EXEC   PGM=ADARUN

//DDDSIMR1 DD   DSN=ADABAS.DB010.DSIMR1,DISP=(NEW,CATLG),
//
//          VOL=SER=...,UNIT=...,SPACE=(CYL,20)

//DDCARD    DD   *

ADARUN      PROG=ADAFRM,...

//DDKARTE   DD   *

ADAFRM DSIMFRM SIZE=20,...

//DDPRINT   DD   SYSOUT=*
```

```
//DDDRUCK DD SYSOUT=*
```

Reset DSIM Dataset (ADAFRM DSIMRESET)

```
//FORMAT EXEC PGM=ADARUN

//DDDSIMR1 DD DSN=ADABAS.DB010.DSIMR1, DISP=OLD

//DDCARD DD *

ADARUN PROG=ADAFRM, ...

//DDKARTE DD *

ADAFRM DSIMRESET FROMRABN=1, SIZE=1B

//DDPRINT DD SYSOUT=*

//DDDRUCK DD SYSOUT=*
```

z/VM

Dataset	DD Name	Storage	More Information
Delta Save images (DSIM)	DDDSIMR1	disk	Required for DSIM... functions
ADAFRM parameters	DDKARTE	disk/ terminal/ reader	
ADARUN parameters	DDCARD	disk/ terminal/ reader	<i>Adabas Operations</i>
ADAFRM messages	DDDRUCK	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>

Format DSIM Dataset (ADAFRM DSIMFRM)

```
DATADEF DDDSIMR1, DSN=ADABASVV.DSIM, VOL=DSIM

DATADEF DDPRINT, DSN=ADAFRM.DDPRINT, MODE=A

DATADEF DDDRUCK, DSN=ADAFRM.DDDRUCK, MODE=A

DATADEF DDCARD, DSN=RUNFRM.CONTROL, MODE=A
```

```
DATADEF DDKARTE, DSN=ADAFRM.CONTROL, MODE=A  
ADARUN
```

Contents of RUNFRM CONTROL A1

```
ADARUN  PROG=ADAFRM, ...
```

Contents of ADAFRM CONTROL A1

```
ADAFRM  DSIMFRM SIZE=20, ...
```

Reset DSIM Dataset (ADAFRM DSIMRESET)

```
DATADEF DDDSIMR1, DSN=ADABASVv.DSIM, VOL=DSIM  
DATADEF DDPRINT, DSN=ADAFRM.DDPRINT, MODE=A  
DATADEF DDDRUCK, DSN=ADAFRM.DDDRUCK, MODE=A  
DATADEF DDCARD, DSN=RUNFRM.CONTROL, MODE=A  
DATADEF DDKARTE, DSN=ADAFRM.CONTROL, MODE=A  
ADARUN
```

Contents of RUNFRM CONTROL A1

```
ADARUN  PROG=ADAFRM, ...
```

Contents of ADAFRM CONTROL A1

```
ADAFRM DSIMRESET FROMRABN=1, SIZE=1B
```

VSE/ESA and z/VSE

Dataset	Symbolic	Storage	Logical Unit	More Information
Delta Save images (DSIM)	DSIMR1	disk		Required for DSIM... functions
ADAFRM parameters		reader	SYSIPT	
ADARUN parameters	CARD CARD	reader tape disk	SYSRDR SYS000 *	<i>Adabas Operations</i>
ADAFRM messages		printer	SYS009	<i>Adabas Messages and Codes</i>
ADARUN messages		printer	SYSLST	<i>Adabas Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information in section [Adabas Utility Functions for Delta Save](#) .

Format DSIM Dataset (ADAFRM DSIMFRM)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// EXEC ADARUN, SIZE=ADARUN
ADARUN PROG=ADAFRM, ...
/*
ADAFRM DSIMFRM SIZE=...
/*
```

Reset DSIM Dataset (ADAFRM DSIMRESET)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// EXEC ADARUN, SIZE=ADARUN
ADARUN PROG=ADAFRM, . . .
/*
ADAFRM DSIMRESET FROMRABN=1, SIZE=1B
/*
```

5

ADAPRI Utility

■ ADAPRI DSIMPRI: Printing the DSIM Dataset	20
■ JCL/JCS Requirements and Examples	21

This chapter covers the following topics:

ADAPRI DSIMPRI: Printing the DSIM Dataset

The ADAPRI DSIMPRI function prints one or more specified DSIM dataset blocks.

```
ADAPRI  DSIMPRI  FROMRABN = block-number
              TORABN = block-number
              [ BATCH ]
              [ DEVICE = device-type | ADARUN-device ]
              [ NOUSERABEND ]
```

Essential Parameters

FROMRABN / TORABN: Range of Blocks to be Printed

The beginning and ending numbers of the DSIM dataset RABNs to be printed. Both values must be specified; there are no defaults. Printing begins with the block number specified with the FROMRABN parameter and ends with the block number specified with the TORABN parameter. Each block in the range is printed in hexadecimal format.

Optional Parameters

BATCH: Output Format

Controls the line length of the printed output. If BATCH is not specified, the default line size is 80 characters. If BATCH is specified, the output line size is 120 characters.

DEVICE: Device Type

The device type on which the DSIM dataset is contained. This parameter is required only if the device type is different from the standard device type assigned by the ADARUN DEVICE parameter.

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Example

```
ADAPRI DSIMPRI FROMRABN=1,TORABN=1
```

Block 1 only of the DSIM dataset is printed.

JCL/JCS Requirements and Examples

This section describes the job control information required to run the Delta Save functions of ADAPRI with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSE systems, and shows examples of each of the job streams.

This section covers the following topics:

- [BS2000/OSD](#)
- [z/OS](#)
- [z/VM](#)
- [VSE/ESA and z/VSE](#)

BS2000/OSD

Dataset	Link Name	Storage	More Information
Delta Save images (DSIM)	DDDSIMR1	disk	Required for the DSIMPRI function
ADAPRI parameters	SYSDTA/DDKARTE		
ADARUN parameters	SYSDTA/DDCARD		<i>Adabas Operations</i>
ADAPRI messages	SYSLST/DDDRUCK		<i>Adabas Messages and Codes</i>
ADARUN messages	SYSOUT/DDPRINT		<i>Adabas Messages and Codes</i>

Example

```

/ . PRI          LOGON

/MOD - JOB - OPTIONS  LISTING=YES

/MOD - TEST          DUMP=YES

/ASS - SYSLST        DO . PRI . LST

/ASS - SYSOUT        DO . PRI . OUT

```

```
/ASS-SYSDTA          *SYSCMD

/SET-FILE-LINK       DDSIMR1, ADABAS.DB010.DSIM
, SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK       DDLIB      , ADABAS.MOD

/REMARK

/START-PROGRAM       FROM-FILE=*MOD(ADABAS.MOD, ADARUN)

ADARUN   PROG=ADAPRI, ...

ADAPRI DSIMPRI FROMRABN=...,TORABN=...

/ASS-SYSLST          *PRIM

/ASS-SYSOUT           *PRIM

/ASS-SYSDTA          *PRIM

/LOGOFF              SYS-OUTPUT=DEL
```

z/OS

Dataset	DD Name	Storage	More Information
Delta Save images (DSIM)	DDDSIMR1	disk	Required for the DSIMPRI function
ADAPRI parameters	DDKARTE		
ADARUN parameters	DDCARD		<i>Adabas Operations</i>
ADAPRI messages	DDDRUCK		<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT		<i>Adabas Messages and Codes</i>

Example

```
//PRINT      EXEC   PGM=ADARUN

//DDDSIMR1   DD      DSN=ADABAS.DB010.DSIMR1, DISP=SHR

//DDCARD     DD      *

ADARUN   PROG=ADAPRI, ...

//DDKARTE    DD      *
```

```

ADAPRI  DSIMPRI FROMRABN=... ,TORABN=...

//DDPRINT  DD      SYSOUT=*

//DDDRUCK  DD      SYSOUT=*

```

z/VM

Dataset	DD Name	Storage Type	More Information
Delta Save images (DSIM)	DDDSIMR1	disk	Required for the DSIMPRI function
ADAPRI parameters	DDKARTE	disk/ terminal/ reader	
ADARUN parameters	DDCARD	disk/ terminal/ reader	<i>Adabas Operations</i>
ADAPRI messages	DDDRUCK	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>

Example

```

DATADEF  DDDSIMR1, DSN=ADABASVv.DSIM, VOL=DSIM

DATADEF  DDPRINT, DSN=ADAPRI.DDPRINT, MODE=A

DATADEF  DDDRUCK, DSN=ADAPRI.DDDRUCK, MODE=A

DATADEF  DDCARD, DSN=RUNPRI.CONTROL, MODE=A

DATADEF  DDKARTE, DSN=ADAPRI.CONTROL, MODE=A

ADARUN

```

Contents of RUNPRI CONTROL A1:

```


```

```
ADARUN  PROG=ADAPRI, ...
```

Contents of ADAPRI CONTROL A1:

```
ADAPRI  DSIMPRI FROMRABN=...,TORABN=...
```

VSE/ESA and z/VSE

Dataset	Symbolic	Storage	Logical Unit	More Information
Delta Save images (DSIM)	DSIMR1	disk		Required for the DSIMPRI function
ADAPRI parameters		reader	SYSIPT	
ADARUN parameters	- CARD CARD	reader tape disk	SYSRDR SYS000 *	<i>Adabas Operations</i>
ADAPRI messages		printer	SYS009	<i>Adabas Messages and Codes</i>
ADARUN messages		printer	SYSLST	<i>Adabas Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information in section [Adabas Utility Functions for Delta Save](#).

Example

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// EXEC ADARUN,SIZE=ADARUN
ADARUN  PROG=ADAPRI, ...

/*

ADAPRI  DSIMPRI FROMRABN=...,TORABN=...

/*
```

6

ADAREP Utility

▪ Delta Save Report Format	26
▪ Delta Save Checkpoints	28
▪ JCL/JCS Requirements and Examples	28

If a Delta Save logging (DLOG) area is defined in the database, the database report utility ADAREP additionally reports the Delta Save status. After the section about alternate RABNs, a Delta Save section is displayed with the following information:

- Delta Save status (disabled or enabled);
- percentage of the DLOG area used;
- number of full save operations since installation of the DLOG area;
- date and time of the last full save;
- number of Delta Save operations since the last full save;
- date and time of the last delta save;
- estimate of the number of database blocks that would be saved in a current Delta Save operation. The estimate may be higher or lower than the actual number of blocks.
- location, device type, block size, and number of unused blocks of the DLOG area.

If no DLOG area is defined, the Delta Save section is omitted.

This chapter covers the following topics:

Delta Save Report Format

The following is an example of the Delta Save section in an ADAREP database report:

```
*****
```

```
* DELTA SAVE FACILITY *
```

```
yyyy-mm-dd hh:mm:ss
```

```
*****
```

```
DELTA SAVE STATUS = ENABLED
```

```
DLOG AREA USAGE   = 2%
```

```
LAST FULL SAVE NUMBER      = 2
```

DATE/TIME OF LAST FULL SAVE = 1997-01-19 04:03:36

LAST DELTA SAVE NUMBER = 5

DATE/TIME OF LAST DELTA SAVE = 1997-01-24 22:39:45

ESTIMATED NUMBER OF CHANGED BLOCKS = CA. 1,700 BLOCKS

DSF LOGGING AREA

LIST	I	DEV	BLOCK	I	SPACE	ALLOC.	I	FROM	TO	I	UNUSED	SPACE	I
TYPE	I	TYPE	LNTH	I	BLOCKS	CYL	I	RABN	RABN	I	BLOCKS	CYL	I
	I			I			I			I			I
-----	I	-----	I	-----	I	-----	I	-----	I	-----	I	-----	I
	I			I			I			I			I
DSF	I	3390	2004	I	100	0	I	1340	1439	I	74	0	I
	I			I			I			I			I
-----		-----		-----		-----		-----		-----		-----	

Delta Save Checkpoints

The following additional checkpoints can be written by the Adabas nucleus or utilities and included in the ADAREP output when running with the Delta Save Facility:

Checkpoint		Originated	
Type	Name	By	Description
0B	SYNP	ADASAV	End of SAVE DELTA operation
0C	SYNP	ADASAV	End of RESTORE DELTA operation
0D	SYNP	ADASAV	End of MERGE operation
0E	SYNV	ADASAV	VOLSER entry for SAVE DELTA operation
0F	SYNV	ADASAV	VOLSER entry for MERGE operation (output volumes)
6A	SYNS	SYSAOS	DSF logging (DLOG) area installed
6B	SYNS	SYSAOS	DLOG area changed
6C	SYNS	SYSAOS	DLOG area removed

JCL/JCS Requirements and Examples

This section describes the job control information required to run the ADAREP utility with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSE systems, and shows examples of each of the job streams.

This section covers the following topics:

- [BS2000/OSD](#)
- [z/OS](#)
- [z/VM](#)
- [VSE/ESA and z/VSE](#)

BS2000/OSD

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
ADAREP parameters	SYSDTA/DDKARTE		
ADARUN parameters	SYSDTA/DDCARD		<i>Adabas Operations</i>
ADAREP messages	SYSLST/DDDRUCK		<i>Adabas Messages and Codes</i>

Dataset	Link Name	Storage	More Information
ADARUN messages	SYSOUT/DDPRINT		<i>Adabas Messages and Codes</i>

Example

```

/.ADAREP LOGON

/OPTION MSG=FH,DUMP=YES

/REMARK *

/REMARK *      A D A R E P      ALL FUNCTIONS

/REMARK *

/SYSFILE SYSLST=L.REP

/FILE      ADA.MOD, LINK=DDLIB

/FILE      ADA99.ASSO  , LINK=DDASSOR1, SHARUPD=YES

/FILE      ADA99.DATA  , LINK=DDDATAR1, SHARUPD=YES

/EXEC (ADARUN, ADA.MOD)

ADARUN PROG=ADAREP, DB=99

ADAREP CPLIST

/LOGOFF NOSPOOL

```

z/OS

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
ADAREP parameters	DDKARTE		
ADARUN parameters	DDCARD		<i>Adabas Operations</i>
ADAREP messages	DDDRUCK		<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT		<i>Adabas Messages and Codes</i>

Example

```
//REP          EXEC  PGM=ADARUN

//STEPLIB      DD    DISP=SHR,DSN=ADABAS.Vvrs.LOADLIB

//*

//DDASSOR1     DD    DISP=SHR,DSN=EXAMPLE.ADA99.ASSOR1

//DDDATAR1     DD    DISP=SHR,DSN=EXAMPLE.ADA99.DATAR1

//SYSUDUMP      DD    SYSOUT=A

//DDDRUCK      DD    SYSOUT=A

//DDPRINT      DD    SYSOUT=A

//DDCARD       DD    *

ADARUN  PROG=ADAREP,SVC=249,DEVICE=3390,DB=99

//DDKARTE      DD    *

ADAREP    CPLIST
```

z/VM

Dataset	DD Name	Storage Type	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
ADAREP parameters	DDKARTE	disk/ terminal/ reader	
ADARUN parameters	DDCARD	disk/ terminal/ reader	<i>Adabas Operations</i>
ADAREP messages	DDDRUCK	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>

Example

```

DATADEF DDASSOR1, DSN=ADABASVv . ASSO, VOL=ASSOV1

DATADEF DDDATAR1, DSN=ADABASVv . DATA, VOL=DATAV1

DATADEF DDPRINT, DSN=ADAREP . DDPRINT, MODE=A

DATADEF DUMP, DUMMY

DATADEF DDDRUCK, DSN=ADAREP . DDDRUCK, MODE=A

DATADEF DDCARD, DSN=RUNREP . CONTROL, MODE=A

DATADEF DDKARTE, DSN=ADAREP . CONTROL, MODE=A

ADARUN

```

Contents of RUNREP CONTROL A1:

```
ADARUN PROG=ADAREP, DEVICE=3390, DB=99
```

Contents of ADAREP CONTROL A1:

```
ADAREP CPLIST
```

VSE/ESA and z/VSE

Dataset	Symbolic	Storage	Logical Unit	More Information
Associator	ASSORn	disk		
Data Storage	DATARn	disk		
ADAREP parameters		reader	SYSIPT	
ADARUN parameters	SYSRDR CARD CARD	reader tape disk	SYSRDR SYS000 *	<i>Adabas Operations</i>
ADAREP messages		printer	SYS009	<i>Adabas Messages and Codes</i>
ADARUN messages		printer	SYSLST	<i>Adabas Messages and Codes</i>

** Any programmer logical unit can be used.*

See the VSE/ESA-related job information in section [Adabas Utility Functions for Delta Save](#).

Example

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// EXEC ADARUN, SIZE=ADARUN
ADARUN  PROG=ADAREP, SVC=xxx, DEVICE=dddd, DB=yyyy
/*
ADAREP CPLIST
/*
```

7

ADARES Utility

■ COPY: Copy Sequential Protection Log	34
■ PLCOPY: Copy PLOG to Sequential Log	38
■ Rebuilding the DSIM Dataset	41

This chapter describes the additional functionality and new parameters of ADARES that are introduced by the Delta Save Facility.

The ADARES COPY and PLCOPY functions include additional operations and parameters for Delta Save operation. These additional Delta Save operations of ADARES are available only when the ADARUN parameter DSF=YES is specified in the DDCARD/CARD input.

This chapter covers the following topics:

- **COPY: Copy Sequential Protection Log**
- **PLCOPY: Copy PLOG to Sequential Log**
- **Rebuilding the DSIM Dataset**
- **JCL/JCS Requirements and Examples**

COPY: Copy Sequential Protection Log

The COPY function copies one *sequential* protection log to another *sequential* protection log dataset. It must be used to copy the sequential protection log of an Adabas nucleus session that has terminated abnormally; only then can the protection log be used for other purposes. Protection logs from several consecutive Adabas sessions may be copied in a single ADARES COPY execution.

The COPY function may be used to copy a sequential protection log from disk to tape before it can be used as input to the ADARES BACKOUT function. ADARES COPY may be used even if subsequent Adabas sessions have created other protection log datasets. ADARES COPY may be executed any number of times for any given input dataset.

The COPY function also copies save tapes created by the ADASAV utility. Only one save dataset can be copied in a single COPY execution.

When the Delta Save Facility is installed and ADARUN DSF=YES, the COPY function

- can also be used to rebuild a DSIM dataset that could not be successfully built by PLCOPY executions because an error occurred. For example, the DSIM dataset might have been too small to hold all online save information extracted from the protection log. Building a complete DSIM dataset is a prerequisite for using the online save dataset in subsequent merge operations.
- additionally extracts all protection log information relating to an online save operation and writes them to the DSIM dataset. This dataset can then be supplied to an ADASAV MERGE or RESTORE DELTA function.

The protection log to be processed must cover a single, complete online save operation. If it covers more than one online save, only the protection log information of the last one is extracted. Any other online save operation may be designated by specifying the block number of the save operation's start checkpoint in the SYN1 parameter.

When copying the protection log for the current nucleus session, the NONUC parameter must be specified.

If the building of the DSIM dataset is the only purpose of the COPY operation, the sequential output may be directed to a dummy dataset so that no data is actually written. The online save information will nevertheless be extracted from the protection log and written to the DSIM dataset.

For ADARES, the DSIM dataset must be specified for shared-update use.

Procedures for rebuilding a DSIM dataset are outlined in the section [Rebuilding the DSIM Dataset](#).

Syntax

```
ADARES COPY  DSIMSIZE = DSIM-dataset-size
               { PLOGNUM = protection-log-num [ , SYN1 = chkpt-block-num ] }
               { FROMPLOG = start-session [ , TOPLOG = stop-session ] }
               [ DSIMDEV = device-type | ADARUN-device ]
               [ NONUC ]
               [ NOUSERABEND ]
               [ OPENOUT ]
               [ TEST ]
               [ TWOCOPIES ]
               [ UTICPLIST ]
```

Essential Parameters

DSIMSIZE: DSIM Dataset Size

The size of the DSIM dataset. The size can be specified in cylinders, or in blocks (by appending "B" to the number). This parameter is mandatory if ADARUN parameter DSF=YES is specified. Otherwise, do not specify DSIMSIZE.

PLOGNUM: Protection Log Number

The Adabas protection log number of the dataset to be copied. This number may be obtained from the database status report produced by the ADAREP utility. The output of the COPY function will be assigned the same log number.

If ADARUN parameter DSF=YES is specified, the PLOGNUM parameter also identifies the protection log number of the SYN1 checkpoint where extraction of online save information is to start.

FROMPLOG: Beginning Session for Backout

FROMPLOG specifies the session number at which the ADARES COPY function is to start. ADARES searches the input (DDSIIN/SIIN) file for the correct starting session.

Optional Parameters

DSIMDEV: DSIM Dataset Device Type

The DSIMDEV parameter specifies the device type of the DSIM dataset. This parameter is required only if the DSIM device type is different from that specified by the ADARUN DEVICE parameter (which is the default). It is allowed only if the ADARUN parameter DSF=YES is specified.

NONUC: Ignore Nucleus Response

In the event that the Adabas nucleus is unable to perform an autorestart, the NONUC parameter may be used to cause the nucleus response code to be ignored. ADARES will then attempt to physically copy the input tape as far as possible.

This parameter should only be specified after all other corrective actions have failed; however, NONUC *must* be specified when copying the PLOG for the current nucleus session. The output dataset will have a valid EOF mark, even if the input dataset does not.

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

OPENOUT: Open DDSIAUS 1/2 or SIAUS 1/2 Output Datasets

The OPENOUT parameter indicates that the DD/SIAUS1/2 output datasets are to be opened by ADARES, even if no data is actually to be copied. Without OPENOUT, the sequential output datasets are not opened if ADARES detects an end-of-file condition while attempting to read the first input record; this may cause problems in some operating system environments. With OPENOUT, the output datasets are opened before the first input record is read.

SYN1: Checkpoint Block Number

The block number of the SYN1 checkpoint on the protection log (PLOG) to be copied where extraction of online save information is to start. This is the start checkpoint of the online save operation. The SYN1 block number may be obtained from the database status report produced by the ADAREP utility.

If the SYN1 parameter is omitted, extraction of online save information starts at the first SYN1 checkpoint encountered. If the PLOG covers several online save operations, extracting the online save information is started anew at every SYN1 checkpoint that occurs. Therefore, if the SYN1 parameter is not specified, the online save information of the last online save operation recorded in the PLOG is extracted.

The SYN1 parameter is only allowed if ADARUN parameter DSF=YES is specified, and can be specified only with the PLOGNUM parameter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values and variables *cannot* be tested; only the syntax of the specified parameters can be tested.

TOPLOG: Ending PLOG Session for Backout

TOPLOG specifies the last session to be processed by the specified ADARES function. If ADARES finds a session on the PLOG input (DDSIIN/SIIN) file that is greater than the specified TOPLOG session, that session is excluded from ADARES processing. If TOPLOG is not specified, the FROMPLOG session becomes the default.

TWOCOPIES: Create Two Output Copies

TWOCOPIES causes two copies of the output to be created.

UTICPLIST: Print All Utility Checkpoints

The UTICPLIST parameter causes ADARES to select and print all utility checkpoints found on the data protection log during the COPY function.

Example

```
ADARES COPY PLOGNUM=1310
ADARES      DSIMSIZE=10
ADARES      DSIMDEV=8381
ADARES      NONUC
```

A sequential protection log of the current active session is to be copied. The session number is 1310. Any protection log information relating to an online save operation is to be extracted and written to the DSIM dataset. The DSIM dataset size is 10 cylinders and the device type is 8381.

PLCOPY: Copy PLOG to Sequential Log

The PLCOPY function copies the *dual* or *multiple* protection log to a *sequential* protection log dataset. The dual or multiple protection log dataset that has the earlier timestamp is copied. Once the PLCOPY function is successfully completed, the copied dataset is reset to an empty status. This function can therefore be used only once for any given portion of the protection log of an entire Adabas nucleus session. The PLCOPY function is only applicable if the Adabas nucleus is run with dual or multiple protection logging.

When the Delta Save Facility is installed and ADARUN DSF=YES, the PLCOPY function also extracts all protection log information relating to an online save operation and writes it to the Delta Save images (DSIM) dataset. This process is called "building the DSIM dataset". The dataset can then be supplied to an ADASAV MERGE or RESTORE DELTA function.

Depending on the number of protection log switches during an online save operation, several PLCOPY executions may be necessary to extract all online save information. The DSIM dataset remains incomplete between two such PLCOPY operations. It is marked ready for merge only after the SYN2 checkpoint indicating the end of an online save operation is found on the protection log.

If the "build" process is interrupted due to an error condition, subsequent merging and/or restore operations cannot take place; this renders the online save tape almost useless. In this case it is necessary to "rebuild" the DSIM dataset using the COPY function.

Procedures for rebuilding a DSIM dataset are outlined in the section [Rebuilding the DSIM Dataset](#).

Syntax

```
ADARES  PLCOPY  DSIMSIZE = DSIM-dataset-size
                [ DSIMDEV = device-type | ADARUN-device ]
                [ DUALPLD = device-type | ADARUN-device ]
                [ PLOGDEV = device-type | ADARUN-device ]
                [ NOUSERABEND ]
                [ OPENOUT ]
                [ TEST ]
                [ TWOCOPIES ]
                [ UTICPLIST ]
```

Essential Parameter

DSIMSIZE: DSIM Dataset Size

The size of the DSIM dataset. The size can be specified in cylinders, or in blocks (by appending "B" to the number). This parameter is mandatory if ADARUN parameter DSF=YES is specified. Otherwise, do not specify DSIMSIZE.

Optional Parameters

DSIMDEV: DSIM Dataset Device Type

The DSIMDEV parameter specifies the device type of the DSIM dataset. This parameter is required only if the DSIM device type is different from that specified by the ADARUN DEVICE parameter (which is the default). It is allowed only if ADARUN parameter DSF=YES is specified.

DUALPLD | PLOGDEV: PLOG Device Type

DUALPLD specifies the device type used for dual protection log datasets; PLOGDEV specifies the device type used for multiple protection log datasets. This parameter is required if the device type used for the dual or multiple protection log datasets is different from that specified with the ADARUN DEVICE parameter.

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

OPENOUT: Open DDSIAUS 1/2 or SIAUS 1/2 Output Datasets

The OPENOUT parameter indicates that the DD/SIAUS1/2 output datasets are to be opened by ADARES, even if no data is actually to be copied. Without OPENOUT, the sequential output

datasets are not opened if ADARES detects an end-of-file condition while attempting to read the first input record; this may cause problems in some operating system environments. With OPENOUT, the output datasets are opened before the first input record is read.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Note that the validity of values *cannot* be tested; only the syntax of the specified parameters can be tested.

TWOCOPIES: Create Two Copies of Output

TWOCOPIES causes two copies of the output to be created. If TWOCOPIES is not specified, the default is one copy.

UTICPLIST: Print All Utility Checkpoints

The UTICPLIST parameter causes ADARES to select and print all utility checkpoints found on the data protection log during the PLCOPY function.

Example

```
ADARES  PLCOPY
ADARES      DSIMSIZE=10
ADARES      DSIMDEV=8381
```

The dual or multiple protection log is to be copied. Any protection log information relating to an online save operation is to be extracted and written to the DSIM dataset. The DSIM dataset size is 10 cylinders and the device type is 8381.

Rebuilding the DSIM Dataset

Using the Delta Save Facility, you can perform full or delta save operations online and automatically combine the resulting save dataset with all changes to the database that took place during the online save. The result is a quasi-offline save dataset, which is equivalent to an offline save taken at the end of the online save operation.

Accumulating database changes during the online save operation depends on the ability of the ADARES PLCOPY function to extract images of changed database blocks from the protection log (PLOG) and write them to the DSIM dataset. This process is called "building the DSIM dataset". If this process is interrupted due to an error condition, subsequent merging and/or restore operations cannot take place; this renders the online save tape almost useless.

Possible error conditions on the DSIM dataset include the following:

- The DSIM dataset is full: that is, it is too small to hold all images of all database blocks changed during the online save operation;
- The status of the DSIM dataset is incorrect; the sequence of operations being performed does not conform to the save/copy/merge cycle;
- An I/O error occurs on the DSIM dataset;
- The DSIM dataset is accidentally used (altered) by another utility.

► **If such an error condition occurs, the correct DSIM dataset can be "rebuilt" using the ADARES COPY function. This is called "recover mode" and requires the following steps:**

- 1 Determine and correct the cause of the error; for example, allocate and format a new, larger DSIM dataset if the current one is too small.
- 2 Identify the copied, sequential PLOG datasets that contain all of the online save operation in question; the checkpoint list from the ADAREP database report may be used to determine the save operation's SYN1 checkpoint block number (the SYN1 checkpoint marks the beginning of an online full or delta save operation).
- 3 Using the chosen sequential PLOG datasets as input, execute the ADARES COPY function to rebuild the DSIM dataset. If sequential PLOG output is not needed, specify a dummy output dataset. If the supplied PLOG input covers multiple online saves, identify the correct one by specifying the SYN1-parameter with the checkpoint block number determined in step 2.
- 4 After the ADARES COPY function completes successfully, the DSIM dataset can be merged with the online save dataset and, optionally, other full or delta save datasets.

8

ADARES JCL/JCS Requirements and Examples

■ BS2000/OSD	44
■ z/OS	47
■ z/VM	49
■ VSE/ESA and z/VSE	51

This section describes the job control information required to run the Delta Save functions of ADARES with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSEsystems, and shows examples of each of the job streams.



Note: When running with the optional Recovery Aid (RLOG), all temporary datasets must also be cataloged in the job control.

This chapter covers the following topics:

BS2000/OSD

Dataset	Link Name	Storage	More Information
Associator	DDASSORn	disk	
Delta Save images (DSIM)	DDDSIMR1	disk	required when ADARUN DSF=YES
Sequential protection log (PLOG)	DDSIIN	tape/disk	input log for COPY
Dual/multiple PLOG	DDPLOGRn	disk	input log for PLCOPY
Copied log	DDSIAUS1	tape/disk	output of COPY, CLCOPY, PLCOPY
Extra copied log	DDSIAUS2	tape/disk	required if a copy function is used with TWOCOPIES
Recovery log (RLOG)	DDRLOGR1	disk	required for RLOG function
ADARES parameters	SYSDTA/DDKARTE		
ADARUN parameters	SYSDTA/DDCARD		<i>Adabas Operations</i>
ADARES messages	SYSLST/DDDRUCK		<i>Adabas Messages and Codes</i>
ADARUN messages	SYSOUT/DDPRINT		<i>Adabas Messages and Codes</i>

Rebuild DSIM Dataset from Sequential Protection Log (ADARES COPY)

```

/ . COPY          LOGON
/ MOD - JOB - OPTIONS  LISTING=YES
/ MOD - TEST         DUMP=YES
/ ASS - SYSLST       DO . PLCOPY . LST
/ ASS - SYSOUT       DO . PLCOPY . OUT
/ SET - JOB - STEP
/ ASS - SYSDTA       *SYSCMD

```



```

/SET-FILE-LINK      DDASSOR1,
ADABAS.DB010.ASSOR1,SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK      DDSIMR1, ADABAS.DB010.DSIM
,SUP=DISK(SHARE-UPD=NO)

/SET-FILE-LINK      DDSIIN  , ADABAS.DB010.PLOG0310

/SET-FILE-LINK      DDSIAUS1, *DUMMY

/SET-FILE-LINK      DDLIB   , ADABAS.MOD

/REMARK

/START-PROGRAM      FROM-FILE=*MOD(ADABAS.MOD,ADARUN)

ADARUN   PROG=ADARES,DSF=YES...

ADARES COPY ...

ADARES      DSIMSIZE=...

/ASS-SYSLST        *PRIM

/ASS-SYSOUT        *PRIM

/ASS-SYSDTA        *PRIM

/LOGOFF            SYS-OUTPUT=DEL

```

Copy Dual/Multiple Protection Log/Build DSIM Dataset (ADARES PLCOPY)

```

/.PLCOPY           LOGON

/MOD-JOB-OPTIONS   LISTING=YES

/MOD-TEST          DUMP=YES

/ASS-SYSLST        D0.PLCOPY.LST

/ASS-SYSOUT        D0.PLCOPY.OUT

/CREATE-FILE       ADABAS.DB010,PLOG.COPY,

/                  (SPACE=RELA(PRIM-ALLOC=960,SEC-ALLOC=96))

```

```
/SET - JOB-STEP

/ASS - SYSDTA          *SYSCMD

/SET - FILE - LINK      DDASS0R1,

ADABAS.DB010.ASS0R1, SUP=DISK(SHARE-UPD=YES)

/SET - FILE - LINK      DDSIMR1, ADABAS.DB010.DSIM

, SUP=DISK(SHARE-UPD=YES)

/SET - FILE - LINK      DDPLOGR1, ADABAS.DB010.PLOGR1

/SET - FILE - LINK      DDPLOGR2, ADABAS.DB010.PLOGR2

/SET - FILE - LINK      DDSIIN  , ADABAS.DB010.PLOG.COPY, OPEN-MODE=OUTIN,

/                          BUFFER-LENGTH=STD(16)

/SET - FILE - LINK      DDLIB   , ADABAS.MOD

/REMARK

/START - PROGRAM        FROM-FILE=*MOD(ADABAS.MOD, ADARUN)

ADARUN   PROG=ADARES, DSF=YES...

ADARES  PLCOPY ...

ADARES      DSIMSIZE=...

/ASS - SYSLST          *PRIM

/ASS - SYSOUT          *PRIM

/ASS - SYSDTA          *PRIM
```

```
/LOGOFF          SYS-OUTPUT=DEL
```

z/OS

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Delta Save images (DSIM)	DDDSIMR1	disk	required when ADARUN DSF=YES
Sequential protection log (PLOG)	DDSIIN	tape/disk	input log for COPY
Dual/multiple PLOG	DDPLOGRn	disk	input log for PLCOPY
Copied log	DDSIAUS1	tape/disk	output of COPY and PLCOPY
Extra copied log	DDSIAUS2	tape/disk	required if a copy function is used with TWOCOPIES
Recovery log (RLOG)	DDRLOGR1	disk	required for RLOG option
ADARES parameters	DDKARTE		
ADARUN parameters	DDCARD		<i>Adabas Operations</i>
ADARES messages	DDDRUCK		<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT		<i>Adabas Messages and Codes</i>

Rebuild DSIM Dataset from Sequential Protection Log (ADARES COPY)

```
//COPY      EXEC  PGM=ADARUN

//DDASSOR1 DD   DSN=ADABAS.DB010.ASSOR1, DISP=SHR

//DDDSIMR1 DD   DSN=ADABAS.DB010.DSIMR1, DISP=OLD

//DDSIIN    DD   DSN=ADABAS.DB010.PLOG0310, DISP=SHR

//DDSIAUS1 DD   DUMMY

//DDCARD     DD   *

ADARUN PROG=ADARES, DSF=YES, ...

//DDKARTE   DD   *

ADARES COPY ...

ADARES      DSIMSIZE=...
```

```
//DDPRINT DD SYSOUT=*  
//DDDRUCK DD SYSOUT=*
```

Copy Dual/Multiple Protection Log/Build DSIM Dataset (ADARES PLCOPY)

```
//PLCOPY EXEC PGM=ADARUN  
  
//DDASSOR1 DD DSN=ADABAS.DB010.ASSOR1, DISP=SHR  
  
//DDPLOGR1 DD DSN=ADABAS.DB010.PLOGR1, DISP=SHR  
  
//DDPLOGR2 DD DSN=ADABAS.DB010.PLOGR2, DISP=SHR  
  
//DDDSIMR1 DD DSN=ADABAS.DB010.DSIMR1, DISP=SHR  
  
//DDSIAUS1 DD DSN=ADABAS.DB010.PLOG1310, DISP=(NEW, CATLG),  
// UNIT=CASS, ...  
  
//DDCARD DD *  
  
ADARUN PROG=ADARES, DSF=YES, ...  
  
//DDKARTE DD *  
  
ADARES PLCOPY ...  
  
ADARES DSIMSIZE=...  
  
//DDPRINT DD SYSOUT=*  
  
//DDDRUCK DD SYSOUT=*
```

z/VM

Dataset	DD Name	Storage Type	More Information
Associator	DDASSORn	disk	
Delta Save images (DSIM)	DDDSIMR1	disk	required when ADARUN DSF=YES
Sequential protection log (PLOG)	DDSIIN	tape/disk	input log for COPY
Dual/multiple PLOG	DDPLOGRn	disk	input log for PLCOPY
Copied log	DDSIAUS1	tape/disk	output of COPY and PLCOPY
Extra copied log	DDSIAUS2	tape/disk	required if a copy function is used with TWOCOPIES
Recovery log (RLOG)	DDRLOGR1	disk	required for RLOG option
ADARES parameters	DDKARTE	disk/ terminal/ reader	
ADARUN parameters	DDCARD	disk/ terminal/ reader	<i>Adabas Operations</i>
ADARES messages	DDDRUCK	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>

Rebuild DSIM Dataset from Sequential Protection Log (ADARES COPY)

```

DATADEF DDASSOR1, DSN=ADABASVv . ASS0, VOL=ASSOV1

DATADEF DDDSIMR1, DSN=ADABASVv . DSIM, VOL=DSIM

DATADEF DDSIIN, DSN=ADABASVv . SIBA, MODE=A

DATADEF DDSIAUS1, DSN=ADABASVv . SIAUS1, UNIT=181, VOL=SIBA01

DATADEF DDPRINT, DSN=ADARES . DDPRINT, MODE=A

DATADEF DDDRUCK, DSN=ADARES . DDDRUCK, MODE=A

DATADEF DDCARD, DSN=RUNRES . CONTROL, MODE=A

DATADEF DDKARTE, DSN=ADARES . CONTROL, MODE=A

```

ADARUN

Contents of RUNRES CONTROL A1:

```
ADARUN  PROG=ADARES, DSF=YES . . .
```

Contents of ADARES CONTROL A1:

```
ADARES  COPY  . . .  
ADARES           DSIMSIZE= . . .
```

Copy Dual/Multiple Protection Log/Build DSIM Dataset (ADARES PLCOPY)

```
DATADef DDASSOR1, DSN=ADABASVv . ASS0, VOL=ASSOV1  
DATADef DDPLOGR1, DSN=ADABASVv . PLOGR1, VOL=PLOGR1  
DATADef DDPLOGR2, DSN=ADABASVv . PLOGR2, VOL=PLOGR2  
DATADef DDDSIMR1, DSN=ADABASVv . DSIM, VOL=DSIM  
DATADef DDSIAUS1, DSN=ADABASVv . SIAUS1, UNIT=181, VOL=SIBA01  
DATADef DDPRINT, DSN=ADARES . DDPRINT, MODE=A  
DATADef DDDRUCK, DSN=ADARES . DDDRUCK, MODE=A  
DATADef DDCARD, DSN=RUNRES . CONTROL, MODE=A  
DATADef DDKARTE, DSN=ADARES . CONTROL, MODE=A
```

ADARUN

Contents of RUNRES CONTROL A1:

```
ADARUN  PROG=ADARES, DSF=YES...
```

Contents of ADARES CONTROL A1:

```
ADARES  PLCOPY ...
ADARES          DSIMSIZE=...
```

VSE/ESA and z/VSE

Dataset	Symbolic	Storage	Logical Unit	More Information
Associator	ASSORn	disk		see note
Delta Save images (DSIM)	DSIMR1	disk		required when ADARUN DSF=YES
Sequential protection log (PLOG)	SIIN	tape/disk		input log for COPY
Dual/multiple PLOG	PLOGRn	disk	*	input log for PLCOPY
Copied log	SIAUS1	tape disk	SYS021 *	output of COPY and PLCOPY
Extra copied log	SIAUS2	tape disk	SYS022 *	required if a copy function is used with TWOCOPIES
Recovery log (RLOG)	RLOGR1	disk		required for RLOG option
ADARES parameters		reader	SYSIPT	
ADARUN parameters	- CARD CARD	reader tape disk	SYSRDR SYS000 *	<i>Adabas Operations</i>
ADARES messages		printer	SYS009	<i>Adabas Messages and Codes</i>
ADARUN messages		printer	SYSLST	<i>Adabas Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information in section [Adabas Utility Functions for Delta Save](#) .

Rebuild DSIM Dataset from Sequential Protection Log (ADARES COPY)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// EXEC ADARUN, SIZE=ADARUN
ADARUN PROG=ADARES, DSF=YES, ...
/*
ADARES COPY
ADARES DSIMSIZE=...
/*
```

Copy Dual/Multiple Protection Log/Build DSIM Dataset (ADARES PLCOPY)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// EXEC ADARUN, SIZE=ADARUN
ADARUN PROG=ADARES, DSF=YES, ...
/*
ADARES PLCOPY
ADARES DSIMSIZE=...
/*
```


9 ADASAV Utility

This chapter describes the ADASAV utility functions added specifically for use with the Delta Save Facility. The functions are:

- **MERGE** for
 - combining "offline" delta save output, either alone or with the last full save output;
 - combining "online" full or delta save output with data from the corresponding DSIM dataset;
 - unloading the DSIM dataset to a sequential delta save dataset.
- **RESTORE DELTA** for restoring a full save and delta save output without a prior merging;
- **SAVE** and **SAVE DELTA** for performing either a full or "changes only" database save operation.

With the exception of 'SAVE database', these functions are only available when the ADARUN parameter DSF=YES is specified. The 'SAVE database' function includes special processing when DSF=YES is specified.

- **Restarting an Interrupted Save Operation**
- **JCL/JCS Requirements and Examples**

10

ADASAV MERGE

■ Consolidating Delta Saves or Updating Full Saves	56
■ Merging Online Saves and DSIM Datasets	57
■ Unloading DSIM Datasets	57
■ Using Unloaded DSIM Datasets as "Online" MERGE Input	58
■ Concatenated Datasets in MERGE Input	59
■ Syntax	60
■ Optional Parameters	60
■ Examples	62

This chapter covers the following topics:

Consolidating Delta Saves or Updating Full Saves

The ADASAV MERGE function takes as input zero or one full save dataset and zero to eight delta save datasets:

DD/DEL1...DD/DELn	up to eight consecutive delta save tapes covering a contiguous time span with no gaps or overlaps. Do not omit a delta save tape.
DD/FULL	the optional full save input tape.

The input datasets may originate from SAVE database, SAVE DELTA, or MERGE functions. They must cover a continuous sequence of full and delta save operations.

ADASAV MERGE combines the input datasets into a single dataset: either a consolidated delta save tape or an updated full save tape:

DD/SAVE1	the new full or delta save output tape.
----------	---

The output dataset may subsequently be used as input for MERGE, RESTORE, or RESTORE DELTA functions.

If the DRIVES parameter is specified, ADASAV MERGE cuts the *full* save output into as many pieces as specified by DRIVES:

DD/SAVE1...DD/SAVEn	up to eight consecutive partial save tapes comprising together the new full save output tape.
---------------------	---

The output datasets created can be used as input to a RESTORE function with the same value of DRIVES.

If only delta save input datasets are specified, the MERGE function creates a new delta save dataset that contains every database block from each input save dataset. If a block is present on more than one input save dataset, its latest occurrence is taken. The result is a "consolidated" delta save dataset that contains all changes since the last full save or since the last delta save not included in the MERGE function.

If a full save dataset is specified as input, the MERGE function creates a new full save dataset that contains the latest image of every used block of the database. This full save dataset is equivalent to the database at the time of the last SAVE DELTA execution.

Merging Online Saves and DSIM Datasets

The "online" variation of the MERGE function takes as input a full or delta save dataset that originates from an online save operation *and* the accompanying DSIM dataset that was specified for the online save operation:

DD/DSIMR1	interim dataset from an online SAVE operation.
-----------	--

An online full or delta save dataset with its corresponding DSIM dataset can also be specified in a merge with "offline" save datasets. The output save dataset from the "online" merge operation is always like an "offline" save dataset and may be supplied together with another online save dataset for a subsequent MERGE operation.

The DSIM dataset is automatically built while ADARES PLCOPY copies the dual or multiple protection logs (PLOGs). Building the DSIM dataset is completed when the PLCOPY function encounters the online save operation's SYN2 checkpoint on the PLOG.

If the MERGE function is performed directly after the online save operation, the DSIM dataset will normally not be completely built when the MERGE function starts. The DSIM dataset will not be ready for merge before the next PLCOPY operation has been performed. The DSIMWAIT parameter tells the MERGE function how long to wait for the DSIM dataset to become ready for merge.

Upon completion of the MERGE function, the DSIM dataset is reset for use in another online save/merge cycle.

Unloading DSIM Datasets

The "unload" variation of the MERGE function transfers the contents of the DSIM dataset

DD/DSIMR1	interim dataset from the last online SAVE operation.
-----------	--

-to a sequential delta save dataset

DD/SAVE1	delta save dataset.
----------	---------------------

-that can be used in MERGE or RESTORE DELTA operations later on as needed.

The "unload" frees the DSIM dataset for the next online SAVE operation.

Unlike the "merge" of the DSIM dataset, the "unload" does not process the associated online full or delta save dataset again immediately after it is created. Instead, more than one online save dataset can be processed later on in a single MERGE or RESTORE DELTA operation by sequencing for each the delta save dataset created from its associated DSIM dataset.

The DSIM dataset is unloaded by running ADASAV MERGE with only the DSIM dataset as input. The PATTERN parameter must be omitted or blank, and the DRIVES parameter must not be set to a value greater than one. The DSIMWAIT parameter functions normally.

Whereas the DSIM dataset is direct access and contains changed blocks from the database in no particular order, the output dataset is sequential and contains the blocks in ascending RABN sequence without duplicates, like any delta save dataset.

An unloaded DSIM dataset is identified by a flag in the save dataset header. The header specifies

- the delta save ID (DSID) of the associated online save; and
- the positions of the online save's SYN1 and SYN2 checkpoints on the PLOG.

The fields mentioned in the header record are filled from the corresponding fields in the DSIM header block.

If the DSIM dataset was (re)built by an ADARES COPY execution, the DSID is not known and will be zero. The associated online save is still correctly identified by the positions of its SYN1 and SYN2 checkpoints on the PLOG.

When a DSIM dataset is unloaded, the MERGE function still generates a SYN0-0D checkpoint but with an indication that the created save dataset is an unloaded DSIM dataset.

Using Unloaded DSIM Datasets as "Online" MERGE Input

When an unloaded DSIM dataset is supplied as input for a MERGE operation, it must be specified

- as one of the DD/DELn input datasets. More than one unloaded DSIM dataset may be supplied for a single MERGE.
- directly after its associated online delta save input; that is, with the next higher DD/DELn index number. For example, if an online delta save input is supplied as DD/DEL2, its associated un-

loaded DSIM dataset must be given as DD/DEL3. If an unloaded DSIM dataset is associated with an online full save, it must be specified as DD/DEL1.

These rules for specifying unloaded DSIM datasets ensure the correct output: each block is taken from the last input in sequence that contains a block image with the same RABN.

The MERGE function checks whether the online save fits the contents of the DSIM dataset. If so, the online save is associated with the DSIM dataset. If not, the next input delta save in logical sequence must be a matching unloaded DSIM dataset.

Concatenated Datasets in MERGE Input

The full save input to the ADASAV MERGE function must be supplied as a single entity. Full saves that are spread over several datasets (by means of the DRIVES parameter in ADASAV), must be concatenated as follows:

- For BS2000/OSD systems, by using the DDFULL01, DDFULL02, ... link names for the second, third, ... datasets;
- For z/OS, and z/VM systems, by using operating system concatenation capabilities;
- For VSE/ESA and z/VSE systems, by using the FULL01, FULL02, ... symbolic names for the second, third, ... datasets.

For a MERGE operation, a delta save dataset may not be concatenated to another full or delta save dataset. If the MERGE operation detects a concatenated dataset at the end of any other dataset in the logical sequence, it terminates with an error message.



Note: In cases where the MERGE function can be completed without reading all input save datasets to their end, wrongly concatenated input save datasets may still go undetected and be ignored.

See the JCL examples for the ADASAV MERGE function in section [JCL/JCS Requirements and Examples](#). for more information on dataset concatenation on BS2000 and VSE/ESA systems, see the *Adabas Utilities* documentation.

Syntax

```
ADASAV  MERGE  [ DRIVES = count | 1 ]  
               [ DSIMDEV = device-type | ADARUN-device ]  
               [ DSIMWAIT = seconds | 0 ]  
               [ NOUSERABEND ]  
               [ PATTERN = merge-pattern ]  
               [ PERDRIVE = disks-per-tape ]  
               [ TEST ]  
               [ TWOCOPIES ]
```

Optional Parameters

DRIVES: Number of Output Datasets

DRIVES can only be specified if the output save dataset is a full save.

The DRIVES parameter specifies the number of output datasets to be created, all of which together build the full save output. The MERGE function cuts the output full save into as many pieces as specified by DRIVES.



Note: The full save input must always be specified using a single DD/link name (DD/FULL). Concatenation can be used, if necessary. See [Concatenated Datasets in MERGE Input](#).

The output datasets created can be used as input to a RESTORE function with the same value of DRIVES.

DSIMDEV: DSIM Device Type

The DSIMDEV parameter specifies the device type of the DSIM dataset. This parameter is required only if the DSIM device type is different from that specified by the ADARUN DEVICE parameter (which is the default).

DSIMWAIT: DSIM Dataset Wait Time in Seconds

The DSIMWAIT parameter specifies how long the MERGE function should wait for the DSIM dataset to become ready for merge.

It can be used if a merge step is to be executed directly after an online save operation but must wait for the ADARES PLCOPY function to copy the nucleus' dual or multiple protection log.

The maximum time to wait is specified in seconds. If the DSIM dataset does not become ready for merge during this interval, and one of the input save datasets is an online save dataset, the MERGE function fails.

If DSIMWAIT is not specified, the MERGE function will not wait for the DSIM dataset to become ready for merge, but instead will begin operation immediately or fail if the DSIM dataset is not ready for merge (the default).

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

PATTERN: Merge Input Pattern

PATTERN is an optional parameter that provides a cross-check between the intended merge input and the actual datasets specified by the job control. PATTERN specifies the count and type of input datasets to the merge operation as single letters "F" (full save tape) or "D" (delta save tape). Only one "F" can be specified in the pattern field, but up to eight "D"s can be entered—one for each existing delta save tape to be merged (additional delta save tapes can be merged in subsequent merge operations with the result of this merge operation).

The pattern starts with an "F" if a full save input is specified, and continues (or starts) with a "D" for every delta save input. For example, PATTERN=FDD specifies that the input comprises one full save dataset and two delta save datasets. No special indicator is given for a save dataset being online. The pattern string must exactly match the input save datasets specified in the job control.

PERDRIVE: Disk Drives Per Tape Drive

PERDRIVE specifies the number of disk drives to be assigned to a single output tape drive. For example, if the database is contained on seven disk drives and three tape drives are available for MERGE processing, PERDRIVE=3,2,2 would cause the first three disk drives to be written to tape drive 1, the next two disk drives to be written to tape drive 2, and the next two disk drives to be written to tape drive 3. The drive sequence corresponds to the DD/SAVEN or DD/DUALn job control specifications, as described in the section [JCL/JCS Requirements and Examples](#).

The total number of drives specified by PERDRIVE must equal the sum of all Associator (DD/ASSORn) and Data Storage (DD/DATARn) disks; if both Associator and Data Storage are on a single disk, this counts as two separate disks. If the DRIVES parameter is used and the PERDRIVE parameter is omitted, ADASAV determines the most efficient utilization of the tape drives.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the merge operation. Note that the validity of values *cannot* be tested; only the syntax of the specified parameters can be tested.

TWOCOPIES: Create Two Copies of Output

TWOCOPIES creates two physical copies of the ADASAV output.

Examples

Example 1:

```
ADASAV MERGE

DSIMDEV=8381, DSIMWAIT=300, PATTERN=DD
```

Two delta save tapes, one created during an online delta save, are to be merged into a single delta save tape. The merge operation will wait up to five minutes (300 seconds) for the DSIM dataset to become available. The DSIM dataset's device type is 8381.

Example 2:

```
ADASAV MERGE PATTERN=FD

ADASAV          DRIVES=4

ADASAV          TWOCOPIES
```

One full save and one delta save dataset are to be merged. The full save output is to be distributed over four datasets, making them suitable for a restore using four drives. Two copies of each output dataset are to be made.

Example 3:

```
ADASAV MERGE PATTERN=F
```

Only one (online) full save and its corresponding DSIM dataset are to be merged. The output is equivalent to an offline full save taken at the end of the online save operation.

Example 4:

```
ADASAV MERGE PATTERN='FDDDDDDDD'
```

A full save and eight delta saves (the maximum number allowed) are to be merged.

11

ADASAV RESTORE DELTA: Restore Without Prior Merging

■ Online Save and DSIM Inputs	66
■ Concatenated Datasets in RESTORE DELTA Input	67
■ Output Database Requirements	68
■ Subsequent Operations	68
■ Syntax	68
■ Optional Parameters	70
■ Examples	78

The RESTORE DELTA function restores a database or file(s) from a full save and zero, one, or more delta save datasets without first merging the save inputs. The input datasets may originate from SAVE database (with DSF=YES), SAVE DELTA, or MERGE functions, and must cover a continuous sequence of full and delta save operations.

For RESTORE DELTA database operations where more input save tapes are present than tape units are available, the excess save tapes may be concatenated to the delta save tape assigned to the last tape unit; however, the logical sequence of delta saves *must* be retained. If concatenation is not possible or not enough tape units are available, the number of input save datasets must be reduced to eight or fewer by doing consecutive merge operations.

An interrupted RESTORE operation must be executed again from the beginning. For RESTORE DELTA database, the restore operation can be restarted without the full save tape if the full save tape was completely restored and only some of the supplied delta save tapes remain to be restored. Message DSF048 in the restore operation's protocol indicates the input save datasets that have been completely restored. The other delta save datasets must be supplied for the restarted restore operation.

This chapter covers the following topics:

Online Save and DSIM Inputs

A full or delta save input dataset may originate from an "online" save operation. In this case, the original or unloaded DSIM dataset that was specified for the online save operation must also be specified as an input dataset. It is possible to specify a single online save input and its corresponding DSIM dataset without specifying any other input save dataset.

The DSIM dataset is automatically built while ADARES PLCOPY copies the dual or multiple PLOGs. Building the DSIM dataset is completed when the PLCOPY function encounters the online save operation's SYN2 checkpoint on the PLOG.

If the RESTORE DELTA function is to be performed directly after the online save operation, the DSIM dataset will normally not be completely built when the RESTORE DELTA function starts. The DSIM dataset will not be ready for merge before the next PLCOPY operation has been performed. The DSIMWAIT parameter tells the RESTORE DELTA function how long to wait for the DSIM dataset to become ready for merge.

When an "unloaded" DSIM dataset is supplied as input for a RESTORE DELTA operation, it must be specified

- as one of the DD/DELn input datasets. More than one unloaded DSIM dataset may be supplied for a single RESTORE.
- directly after its associated online delta save input; that is, with the next higher DD/DELn index number. For example, if an online delta save input is supplied as DD/DEL2, its associated un-

loaded DSIM dataset must be given as DD/DEL3. If an unloaded DSIM dataset is associated with an online full save, it must be specified as DD/DEL1.

These rules for specifying unloaded DSIM datasets ensure the correct output: each block is taken from the last input in sequence that contains a block image with the same RABN.

The RESTORE DELTA function checks whether the online save fits the contents of the DSIM dataset. If so, the online save is associated with the DSIM dataset. If not, the next input delta save in logical sequence must be a matching unloaded DSIM dataset.

A RESTORE DELTA function may postpone processing the expected unloaded DSIM dataset until a subsequent phase of the restore. For example, an online full save is first restored on multiple drives and then the associated DSIM dataset is handled, which must be either the original or an unloaded DSIM dataset; or, if the last delta save input in logical sequence contains an online save, the associated unloaded DSIM must be concatenated to that online save. If the expected unloaded DSIM dataset is missing, the restore terminates with an error message, leaving the database intact for a proper RESTORE DELTA from the most recent restart point on, but with all files in restore-status.

During a RESTORE DELTA operation, ADASAV prints the DSF048 partial completion messages providing restart points in case of subsequent failure only when the full and delta saves already restored really form a consistent state of the database. A DSF048 message is suppressed if an online save is restored but the associated DSIM dataset will be processed in a subsequent phase of the restore.

Upon completion of the RESTORE DELTA, the DSIM dataset is *not* reset; it is still available for a MERGE operation.

Concatenated Datasets in RESTORE DELTA Input

Consecutive delta save results can be combined (concatenated) as input to the ADASAV RESTORE DELTA function. The last delta save input (the one with the most recent time stamp) can be a concatenation of several consecutive delta save tapes, as follows:

- For BS2000/OSD systems, by using the DDDELn01, DDDELn02, ... link names, where the last delta save input has the serial number "n";
- For z/OS, and z/VM systems, by using the available operating systems capabilities;
- For VSE/ESA and z/VSE, by using the DELn01, DELn02, ... symbolic names, where the last delta save input has serial number "n".

See the JCL examples for the ADASAV RESTORE DELTA function in section [JCL/JCS Requirements and Examples](#). For more information on dataset concatenation on BS2000/OSD, z/VSE and VSE/ESA systems, see the *Adabas Utilities* documentation.

Output Database Requirements

For RESTORE DELTA database or RESTORE DELTA GCB, the output database must have the same physical layout (device types, extent sizes) as the original database. The Associator and Data Storage datasets must be present and must have been formatted at least once. The Adabas nucleus must not be active. If the dual or multiple protection log (PLOG) and/or dual or multiple command log (CLOG) datasets are specified in the job control, these datasets are reset to an empty status.

For RESTORE DELTA FILES or RESTORE DELTA FMOVE, an existing database must be present. The files to be restored may come from the same or a different database. The Adabas nucleus may be active. If the nucleus is active when checkpoint or security files are restored, the ADASAV utility requires exclusive database control; that is, no user may be active on the database.

Subsequent Operations

After a RESTORE database or RESTORE DELTA database operation, another RESTORE DELTA database function may be executed without the full save input dataset, provided that

- a DSF logging (DLOG) area is defined in the database;
- no Adabas nucleus session has been started since the last restore operation;
- no utility that makes changes to the database (ADALOD, ADASAV RESTORE FILE) has been run since the last restore operation; and
- the delta save tape to be restored is the next in the logical sequence of delta save tapes already restored.

This way of executing the RESTORE DELTA function can be used to keep a shadow database up to date.

Syntax

The minimum syntax is ADASAV RESTORE DELTA. A database restore is assumed unless the FILES or FMOVE parameter is specified.

Any or all of the parameters DSIMDEV, DSIMWAIT, PATTERN, and TEST may be optionally specified independently. The parameters BUFNO, CLOGDEV, DRIVES, EXCLUDE, NEWDBID, NEWDBNAME, OVERWRITE, and PLOGDEV have special dependencies on other parameters as indicated in the individual parameter discussions.

The FILES, FMOVE, and GCB parameters are optional. However, if FILES or GCB is specified, FMOVE may not be specified and vice versa. FILES may be specified with or without GCB. The parameters available with FILES, FMOVE, and GCB are not depicted in the following syntax diagram; they are depicted instead with their detail descriptions instead.

ALLOCATION, NEWFILES, and PASSWORD are allowed with FILES or FMOVE only.

If FMOVE is specified, any or all of the parameters ACRABN, ALLOCATION, ASSOVLUM, DATAVOLUME, DRIVES, DSRABN, DSSIZE, MAXISN, NEWFILES, NIRABN, NISIZE, PASSWORD, UIRABN, and UISIZE may be specified.

If GCB is specified, any or all of the parameters CLOGDEV, FILES, NEWDBID, NEWDBNAME, and PLOGDEV may be specified.

```

ADASAV  RESTORE  DELTA
  [ DSIMDEV = { device-type | ADARUN-device } ]
  [ DSIMWAIT = { seconds | 0 } ]
  [ NOUSERABEND ]
  [ PATTERN = merge-pattern ]
  [ TEST ]
  [ BUFNO = { number-of-buffers | 1 } ]
  [ CLOGDEV = { clog-device-type | ADARUN-device } ]
  [ DRIVES = { count | 1 } ]
  [ EXCLUDE = file-list ]
  [ NEWDBID = new-database-ID ]
  [ NEWDBNAME = new-databas-name ]
  [ OVERWRITE ]
  [ PLOGDEV = { plog-device-type | ADARUN-device } ]

  {
    [ FILES = file-list [ parameters ] ] [ GCB [ parameters ] ]
    [ FMOVE = file-list [ parameters ] ]
  } **

```

**** = Parameters for FILES, FMOVE, and GCB are shown with their descriptions.**

Optional Parameters

ACRABN: Starting AC RABN

ACRABN specifies the starting address converter RABN for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

If FMOVE is specified and ACRABN omitted, the location of the address converter is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the ACRABN parameter must correspond to the list of files in the FMOVE parameter. If no ACRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the [Example](#).

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

ALLOCATION can only be specified if FILES or FMOVE is specified. If specified with FILES, ALLOCATION pertains to the implicit RABN specifications derived from the files on the save dataset.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

ASSOVOLUME: Associator Extent Volume



Note: The value for ASSOVOLUME must be enclosed in apostrophes.

ASSOVOLUME identifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) is to be allocated. If the requested number of blocks cannot be found on the specified volume, ADASAV retries the allocation while disregarding the ASSOVOLUME parameter.

If ACRABN, UIRABN, or NIRABN is specified, ADASAV ignores the ASSOVOLUME value when allocating the corresponding extent type. If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADASAV's default allocation rules.

If several files are to be restored, the list of volumes in the ASSOVOLUME parameter must correspond to the list of files in the FMOVE parameter. If no ASSOVOLUME value is to be given for a file, its entry in the list of volumes must be left empty. See the [Example](#).

BUFNO: Count of Buffers Per Drive

The BUFNO value, multiplied by the DRIVES parameter value, allocates fixed buffers for the RESTORE DELTA operation. A value of 2 or 3 usually provides optimum performance; a value up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

See also the DRIVES parameter .

CLOGDEV: Command Log Device Type

CLOGDEV is the device type to be assigned to the dual or multiple command log (CLOG). This parameter is required only if the device type to be used for the CLOG is different from that specified by the ADARUN DEVICE parameter. CLOGDEV is required only for the RESTORE DELTA database and RESTORE DELTA GCB functions if the dual or multiple CLOG is specified in the job control.

DATAVOLUME: Data Storage Extent Volume

Note: The value for DATAVOLUME must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) is to be allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADASAV retries the allocation while disregarding the DATAVOLUME value.

If DSRABN is specified, DATAVOLUME is ignored for the related file. If DATAVOLUME is not specified, the Data Storage space is allocated according to ADASAV's default allocation rules.

If several files are to be restored, the list of volumes in the DATAVOLUME parameter must correspond to the list of files in the FMOVE parameter. If no DATAVOLUME value is to be given for a file, its entry in the list of volumes must be left empty. See the [Example](#).

DRIVES: Tape Drives for Parallel Restore

ADASAV is able to restore files from multiple full SAVE dataset volumes in parallel to RABNs that are different from their original RABNs in the database. DRIVES is the number (1-8 inclusive; default 1) of tape drives to be used for parallel restore processing.

The corresponding number of input datasets must be specified in the job control. These input datasets, combined, must represent a full save dataset.

The default value (DRIVES=1) is required when GCB, FILES, or FMOVE is specified in conjunction with DELTA.

DSIMDEV: DSIM Device Type

The DSIMDEV parameter specifies the device type of the DSIM dataset. This parameter is required only if the DSIM device type is different from that specified by the ADARUN DEVICE parameter (which is the default).

DSIMWAIT: DSIM Dataset Wait Time in Seconds

The DSIMWAIT parameter specifies how long the restore function should wait for the DSIM dataset to become ready for merge. DSIMWAIT can be used if a restore step is to be executed directly after an online save operation but must wait for the ADARES PLCOPY function to copy the nucleus' dual or multiple protection log (PLOG).

The maximum time to wait is specified in seconds. If the DSIM dataset does not become ready for restore operation within this time and one of the input save datasets is an online save dataset, the RESTORE DELTA function fails.

If DSIMWAIT is not specified, the RESTORE DELTA function will not wait for the DSIM dataset to become ready for restore operation, but instead will begin operation immediately or fail if the DSIM dataset is not ready for restore operation (the default).

DSRABN: Starting Data Storage RABN/RABN List

DSRABN specifies the starting Data Storage RABN for each file specified by FMOVE. DSRABN can only be used in conjunction with the FMOVE parameter.

If FMOVE is specified and DSRABN omitted, the location of the file's Data Storage is chosen by ADASAV from the free areas in Data Storage that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the DSRABN parameter must correspond to the list of files in the FMOVE parameter. If no DSRABN value is specified for a file, its entry in the RABN list must be specified as zero. See the [Example](#).

DSSIZE: New Data Storage Size

DSSIZE is the new size to be allocated for Data Storage for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter. If DSSIZE is omitted, the original Data Storage size is used.

The size can be specified in cylinders, or in blocks (by appending "B" to the number). It must be at least as large as the used area of the original Data Storage.

If several files are to be restored, the list of sizes in the DSSIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the [Example](#).

EXCLUDE: Exclude Specified Files from Restore

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI). EXCLUDE lists the file numbers to be excluded from the restore operation; that is, the files that are not to be restored. Files specified in the EXCLUDE parameter will not exist in the restored database.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

For a database restore, all files specified in the EXCLUDE parameter must exist on the save dataset.

For a RESTORE DELTA *without* a full save input, the EXCLUDE file list must be the same as that specified for the preceding RESTORE DELTA *with* a full save input; otherwise, the EXCLUDE parameter is rejected.

For a file restore

- without the NEWFILES parameter, all files specified in the EXCLUDE parameter must also be specified in the FILES or FMOVE parameter.
- with the NEWFILES parameter, all files specified in the EXCLUDE parameter must also be specified in the NEWFILES parameter. In this case, the file numbers specified in the EXCLUDE parameter refer to the new file numbers in the NEWFILES parameter, not the old file numbers in the FILES or FMOVE parameter.

FILES: Files to be Restored



Note: When FILES is specified, a RESTORE DELTA requires a full save input dataset.

The file or list of files to be restored. For an Adabas expanded file, all component files of the expanded file including the anchor file must be specified. If a specified file is coupled to other files, the coupled files must also be specified.

The specified files are restored at the same place where they were originally located. The Adabas nucleus may be active during the restore operation unless the GCB parameter is also specified, or the Adabas checkpoint or security file is to be restored.

The FILES parameter cannot be specified if the FMOVE parameter is specified.

The syntax of the FILES parameter is shown next. The subparameters for the FILES parameter are described elsewhere in this section.

```
FILES = file-list
      [ ALLOCATION = { FORCE | NOFORCE } ]
      [ NEWFILES = file-list ]
      [ PASSWORD = password-list ]
```

FMOVE: Files to Be Restored to a New Location



Note: When FMOVE is specified, a RESTORE DELTA requires a full save input dataset.

The file or list of files to be restored. For an Adabas expanded file, all component files of the expanded file including the anchor file must be specified. If a specified file is coupled to other files, the coupled files must also be specified.

The specified files are restored as designated by the ACRABN, ASSOVLUM, DATA-VOLUME, DSRABN, DSSIZE, MAXISN, NIRABN, NISIZE, UIRABN, and UISIZE parameters. If any of these parameters is not specified, ADASAV chooses values for the locations of the corresponding parts of the file(s), and/or retains the existing sizes. The Adabas nucleus may be active during the restore operation unless the Adabas checkpoint or security file is to be restored.

FMOVE cannot be specified if the GCB or FILE parameter is specified.

The syntax of the FMOVE parameter is shown next. The subparameters for the FMOVE parameter are described elsewhere in this section.

```
FMOVE = file-list  
    [ ACRABN = ac-start-rabn-list ]  
    [ ALLOCATION = { FORCE | NOFORCE } ]  
    [ ASSOVOLUME = 'asso-extend-volume-list' ]  
    [ DATAVOLUME = 'data-extend-volume-list' ]  
    [ DSRABN = ds-start-rabn-list ]  
    [ DSSIZE = ds-size-list ]  
    [ MAXISN = isn-count-list ]  
    [ NEWFILES = file-list ]  
    [ NIRABN = ni-start-rabn-list ]  
    [ NISIZE = ni-size-list ]  
    [ PASSWORD = password-list ]  
    [ UIRABN = ui-start-rabn-list ]  
    [ UISIZE = ui-size-list ]
```

GCB: Restore General Control Block

The GCB keyword parameter specifies that the GCB and the other administrative blocks are to be restored from the input save dataset(s), as well as any files specified by the FILES parameter. That is, the entire database is restored except for the files not specified by FILES. The Adabas checkpoint and security files (if present) are always restored.

The Adabas nucleus must *not* be active when RESTORE DELTA GCB is being executed. The GCB parameter cannot be specified if the FMOVE parameter is specified.



Important: Any existing database in the target Associator and Data Storage datasets is completely overwritten, and any files in that database are lost.

The syntax of the GCB parameter is shown next. The subparameters for the GCB parameter are described elsewhere in this section.

GCB

```
[ CLOGDEV = { clog-device-type | ADARUN-device } ]
[ FILES = file-list ]
    [ PASSWORD = password-list ]
[ NEWDBID = new-database-ID ]
[ NEWDBNAME = new-database-name ]
[ PLOGDEV = { plog-device-type | ADARUN-device } ]
```

MAXISN: New Maximum ISN

MAXISN is the new number of ISNs to be allocated for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

The value must be at least as large as the original highest used ISN.

If MAXISN is omitted, the original ISN count is used.

If several files are to be restored, the list of ISN counts in the MAXISN parameter must correspond to the list of files in the FMOVE parameter. If no ISN count is to be given for a file, its entry in the ISN count list must be specified as zero. See the [Example](#).

If the database consists of several Associator extents with different device types, error-171 may occur if MAXISN is specified. If this happens, remove the MAXISN parameter for the file indicated in the error message and rerun RESTORE DELTA FMOVE.

NEWDBID: New Database ID

NEWDBID may be used to assign a different database ID to the restored database. The ID can be in the range 1-65535. NEWDBID can only be specified for ADASAV RESTORE DELTA database and RESTORE DELTA GCB operations.

- If NEWDBID is specified, the ADARUN DBID parameter must specify the ID of the database on the save tape.
- If NEWDBID is not specified, the restored database keeps its old ID.

NEWDBNAME: New Database Name

NEWDBNAME assigns a new name to the restored database. NEWDBNAME can only be specified for ADASAV RESTORE DELTA database and RESTORE DELTA GCB operations. If NEWDBNAME is not specified, the restored database keeps its old name.

NEWFILES: New File Numbers

The NEWFILES parameter specifies the new file number to be assigned to each file specified by FILES or FMOVE.

The parameter is optional: if no new file number is assigned to a file, the file retains its original number.

NEWFILES may not be specified for expanded files, physically coupled files, or replicated files.

If a file with a number specified by NEWFILES already exists in the database, the corresponding file will not be restored unless the OVERWRITE parameter is also specified. If the file to be overwritten is password-protected, the corresponding PASSWORD parameter must also be specified.

If several files are to be restored, the list of file numbers in the NEWFILES parameter must correspond to the list of files in the FILES or FMOVE parameter. If no new file number is to be assigned to a file, its entry in the file number list of NEWFILES must be specified as zero. See the [Example](#).

NIRABN: Starting Normal Index RABN/RABN List

NIRABN specifies the starting normal index RABN for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

If FMOVE is specified and NIRABN omitted, the location of the normal index is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the NIRABN parameter must correspond to the list of files in the FMOVE parameter. If no NIRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the [Example](#).

NISIZE: New Size for Normal Index

NISIZE is the new size to be allocated for the normal index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

The size can be specified in cylinders, or in blocks (by appending "B" to the number). It must be at least as large as the used area of the original normal index.

If NISIZE is omitted, the original normal index size is used.

If several files are to be restored, the list of sizes in the NISIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the [Example](#).

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

OVERWRITE: Overwrite an Existing Database or File

The OVERWRITE parameter specifies that an existing database or file is to be overwritten by the RESTORE DELTA function.

If the FILES or FMOVE parameter is not specified, OVERWRITE specifies that an existing database is to be overwritten. If OVERWRITE is not specified and a GCB is found in Associator RABN 1, the RESTORE function is rejected.

If the FILES or FMOVE parameter is specified, OVERWRITE specifies that any existing file specified by this parameter is to be overwritten. If OVERWRITE is not specified and an FCB is found for any file to be restored, that file will not be restored.

See also the **PASSWORD** parameter.

PASSWORD: Adabas Security File Password

PASSWORD specifies one password or a list of passwords if one or more files in the FILES or FMOVE file list are password-protected. This also applies to files already in the database that are to be overwritten. If the NEWFILES parameter is specified, the PASSWORD parameter must specify the passwords related to the new file numbers.

When restoring more than one password-protected file, the correct passwords must be specified as positional values corresponding to the protected file numbers' positions in the FILES or FMOVE list. See the **Example** for more information about the PASSWORD parameter. When overwriting password-protected files, the Adabas nucleus must be active.

PATTERN: Merge Input Pattern

PATTERN is an optional parameter that provides a cross-check between the intended save input and the actual datasets specified by the job control. PATTERN specifies the count and type of input datasets to the save operation as single letters "F" (full save tape) or "D" (delta save tape). Only one "F" can be specified in the pattern field, but up to eight "D"s can be entered—one for each existing delta save tape to be included in the restore operation.

The pattern starts with an "F" if a full save input is specified, and continues (or starts) with a "D" for every delta save input. For example, PATTERN=FDD specifies that the input comprises one full save dataset and two delta save datasets. No special indicator is given for a save dataset being online. The pattern string must exactly match the input save datasets specified in the job control.

Concatenated delta save input does not take part in the matching of the PATTERN parameter.

PLOGDEV: Protection Log Device Type

The device type of the dual or multiple protection log (PLOG). This parameter is required only if the device type of the PLOG is different from that specified by the ADARUN DEVICE parameter. PLOGDEV is required only for the RESTORE DELTA database and RESTORE DELTA GCB functions if the dual or multiple protection log is specified in the job control.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the restore operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

UIRABN: Starting Upper Index RABN/RABN List

UIRABN specifies the starting upper index RABN for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

If FMOVE is specified and UIRABN omitted, the location of the upper index is chosen by ADASAV from the free areas in the Associator that have the same device type as used originally.

If several files are to be restored, the list of RABNs in the UIRABN parameter must correspond to the list of files in the FMOVE parameter. If no UIRABN value is to be given for a file, its entry in the RABN list must be specified as zero. See the [Example](#).

UISIZE: New Upper Index Size

UISIZE is the new size to be allocated for the upper index for each file specified by FMOVE. It can only be used in conjunction with the FMOVE parameter.

The size can be specified in cylinders, or in blocks (by appending "B" to the number). It must be at least as large as the used area of the original upper index.

If UISIZE is omitted, the original upper index size is used.

If several files are to be restored, the list of sizes in the UISIZE parameter must correspond to the list of files in the FMOVE parameter. If no size is to be given for a file, its entry in the size list must be specified as zero. See the [Example](#).

Examples

Example 1:

```
ADASAV RESTORE DELTA,OVERWRITE  
  
ADASAV      PATTERN=FDD  
  
ADASAV      DSIMDEV=8381  
  
ADASAV      DRIVES=4
```

```
ADASAV      PLOGDEV=3390,CLOGDEV=3390
```

An entire database is to be restored, possibly overwriting an already existing database. One full save and two delta save datasets are supplied as input. One of the input save datasets may have been created online; its accompanying DSIM dataset is on a device type 8381.

The full save input is supplied as four datasets and is to be restored from four tape drives in parallel. The dual or multiple protection log and dual or multiple command log datasets are specified in the job control; they both have device type 3390. They are reset to an empty status as part of the restore operation.

Example 2:

```
ADASAV RESTORE DELTA,EXCLUDE=1000,255
```

Files 255 and 1000 are excluded from the restore of the database from full and delta save datasets.

Example 3:

```
ADASAV RESTORE DELTA,OVERWRITE
ADASAV      FMOVE=10,20,30
ADASAV      ACRABN=31415
ADASAV      MAXISN=1000000,0,20000
ADASAV      DATAVOLUME='ADADS1,,ADADS1'
ADASAV      DSSIZE=0,0,500B
ADASAV      NIRABN=0,10001
ADASAV      NISIZE=0,50
ADASAV      UIRABN=0,9901
ADASAV      UISIZE=0,100B
ADASAV      PASSWORD='PSW10,,PSW30'
```

Files 10, 20, and 30 are to be restored to new locations in the database, possibly overwriting already existing files. The number and type of input save datasets is not specified and is to be determined from the job control.

File 10's address converter is to be placed beginning at RABN 31,415, with a new highest allocated ISN of 1,000,000. The normal and upper index of file 20 are to be placed beginning at RABNs 10,001 and 9,901, with sizes of 50 cylinders and 100 blocks, respectively. The new highest allocated ISN of file 30 is to be 20,000. The Data Storage of file 10 is to be placed on volume ADADS1, if possible. The Data Storage of file 30 is to be placed on volume ADADS1 as well, if possible, with a size of 500 blocks. No placement instructions are provided for the Data Storage of file 20. File 10 has password "PSW10", file 30 has password "PSW30".

Example 4:

```
ADASAV RESTORE DELTA
```

```
ADASAV      FILES=11,12,13, 14,OVERWRITE
```

```
ADASAV      NEWFILES=16,0,17
```

Files 11, 12, 13, and 14 are to be restored. Files 11 and 13 are to be restored as files 16 and 17, respectively. The file numbers of files 12 and 14 will not be changed because the corresponding NEWFILES parameter values were specified as zero or omitted. Files 12, 14, 16, and 17 are to be overwritten, if already present in the database.

12

ADASAV SAVE: Save Database

▪ Syntax	83
▪ Optional Parameters	83
▪ Example	84

The SAVE database function saves all blocks of the database that are in use to create a full save dataset. If a Delta Save (DLOG) area is defined at the time of the save, the full save dataset can be specified as input for subsequent MERGE or RESTORE DELTA functions. In any case, the dataset can be specified as input for RESTORE or RESTONL functions.

The SAVE database function enables Delta Save logging at the end of the save if a DLOG area is defined in the database and the ADARUN parameter DSF=YES is specified. The next save operation may then be SAVE DELTA.

SAVE database may be executed with the Adabas nucleus active or inactive.

- If the Adabas nucleus is *inactive*, it cannot be started while the SAVE database function is executing, and no utility that makes changes to the database (e.g. ADALOD) can be run during this time. SAVE database cannot be executed offline if a nucleus session autorestart is pending, or if another offline utility (ADALOD or ADASAV) is currently running.
- If the Adabas nucleus is *active*, users have full access to the database. They can perform read, find, update, insert, and delete commands. However, utilities that make changes to the database (ADALOD, ADAINV, ADADBS DELETE, etc.) may not be running and cannot be started while the SAVE database function is performed. An online save operation is also not possible if the nucleus is running without protection logging.

If the nucleus is active during the SAVE database function, an ET-synchronization is performed at the end of the save operation to bring all user transactions to ET status. During ET synchronization, transactions already begun are allowed to continue while the start of new transactions is delayed until the end of the ET synchronization. The maximum time required for this synchronization can be limited by the TTSYN parameter.

For an online SAVE database function using the Delta Save Facility, a DSIM dataset must be supplied with DD name/link name DD/DSIMR1. This dataset receives all database blocks changed by the nucleus during the execution of SAVE database. The DSIM dataset must be supplied together with the created online save dataset for a MERGE or RESTORE DELTA function. Until the DSIM dataset has been specified for a subsequent MERGE operation, it cannot be reused for another online SAVE or SAVE DELTA operation (unless it is specifically reset by the ADAFRM DSIMRESET function).

If the execution of the SAVE database function is interrupted, it can be restarted using procedures outlined in the section [Restarting an Interrupted Save Operation](#).

This chapter covers the following topics:

Syntax

```

ADASAV SAVE  [ BUFNO = number-of-buffers | 1 ]
                [ DRIVES = count | 1 ]
                [ DSIMDEV = device-type | ADARUN-device ]
                [ NOUSERABEND ]
                [ PERDRIVE = disks-per-tape ]
                [ TEST ]
                [ TTSYN = seconds | ADARUN-tt ]
                [ TWOCOPIES ]

```

Optional Parameters

BUFNO: Count of Buffers Per Drive

The BUFNO value, multiplied by the DRIVES parameter value, allocates fixed buffers for the SAVE operation. A value of 2 or 3 usually provides optimum performance; a value up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

See also the DRIVES parameter .

DRIVES: Tape Drives for Parallel Save Processing

DRIVES is the number of tape drives to be used for parallel SAVE operations. A maximum of eight drives may be specified. The default is 1.

DSIMDEV: DSIM Device Type

The DSIMDEV parameter specifies the device type of the DSIM dataset. This parameter is required only if the DSIM device type is different from that specified by the ADARUN DEVICE parameter (which is the default).

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

PERDRIVE: Disk Drives Per Tape Drive

PERDRIVE specifies the number of disk drives to be assigned to a single output tape drive.

For example, if the database is contained on seven disk drives and three tape drives are available

for SAVE processing, PERDRIVE=3,2,2 would cause the first three disk drives to be written to tape drive 1, the next two disk drives to be written to tape drive 2, and the next two disk drives to be written to tape drive 3. The drive sequence corresponds to the DDSAVEn/ DDDUALn or SAVEn/ DUALn job control specifications, as described in the section [JCL/JCS Requirements and Examples](#).

The total number of drives specified by PERDRIVE must equal the sum of all Associator (ASSO) and Data Storage (DATA) disks; if both ASSO and DATA are on a single disk, this counts as two separate disks. If the DRIVES parameter is used and the PERDRIVE parameter is omitted, ADASAV determines the most efficient utilization of the tape drives.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TTSYN: SYN2 Checkpoint Control

TTSYN allows the user to decrease the nucleus' ADARUN TT (maximum transaction time) during the synchronized checkpoint processing of the current ADASAV operation. The value specified is the approximate time in seconds (TT + 1.05 seconds), and must be less than the current ADARUN TT value. If TTSYN is not specified or if TTSYN is greater than the nucleus' TT value, that TT value becomes the default.

If the Adabas nucleus is active while ADASAV SAVE is running, a synchronized SYN2 checkpoint is taken at the end of the SAVE operation. This ensures that there is a point in time where all users are at ET status. If a user is not at ET status, no new transactions can be started for other users; they must wait until the SYN2 checkpoint can be taken.

The ADARUN TT value controls the maximum elapsed time permitted for a logical transaction. This is the maximum wait time until the SYN2 checkpoint can be processed. The ADASAV SAVE TTSYN parameter allows the user to decrease the TT value only during the synchronized checkpoint processing. The original TT value becomes effective again when ADASAV ends the SAVE operation.

TWOCOPIES: Create Two Copies of Output

TWOCOPIES creates two physical copies of the ADASAV output.

Example

ADASAV	SAVE
ADASAV	DRIVES=4

ADASAV TTSYN=10

The complete database is to be saved using four tape drives in parallel. If running online, the ET synchronization at the end of the save operation should last at most 10 seconds.

13

ADASAV SAVE DELTA: Save Changed Database Blocks

▪ Syntax	89
▪ Optional Parameters	89
▪ Example	90

The SAVE DELTA function is only available if ADARUN parameter DSF=YES is specified. It can only be executed if Delta Save logging is enabled.

The SAVE DELTA function saves all blocks of the database that have been changed since the execution of the last SAVE database or SAVE DELTA function. It creates a delta save dataset. This dataset can be specified as input for a subsequent MERGE or RESTORE DELTA function.

SAVE DELTA may be executed with the Adabas nucleus active or inactive.

- If the Adabas nucleus is *inactive*, it cannot be started while the SAVE DELTA function is executing, and no utility that makes changes to the database (e.g. ADALOD) can be run during this time. SAVE DELTA cannot be executed offline if a nucleus session Autorestart is pending, or if another offline utility (ADALOD or ADASAV) is currently running.
- If the Adabas nucleus is *active*, users have full access to the database. They can perform read, find, update, insert, and delete commands. However, utilities that make changes to the database (ADALOD, ADAINV, ADADBS DELETE, etc.) may not be running and cannot be started while the SAVE DELTA function is performed. For an online delta save operation the nucleus must be running with dual or multiple protection logging.

If the Adabas nucleus is active during the SAVE DELTA function, an ET synchronization is performed at the end of the save operation to bring all user transactions to ET status. During ET synchronization, transactions already begun are allowed to continue while the start of new transactions is delayed until the end of the ET synchronization. The maximum time required for this synchronization can be limited by the TT SYN parameter.

For an online SAVE DELTA function, a DSIM dataset must be supplied with DD name/link name DD/DSIMR1. This dataset receives all database blocks changed by the nucleus during the execution of SAVE DELTA. The DSIM dataset must be supplied together with the created online save dataset for a MERGE or RESTORE DELTA function. Until the DSIM dataset has been specified for a subsequent MERGE operation, it cannot be reused for another online SAVE or SAVE DELTA operation (unless it is specifically reset by the ADAFRM DSIMRESET function).

If the execution of the SAVE DELTA function is interrupted, it can be restarted using procedures outlined in the section [Restarting an Interrupted Save Operation](#).

This chapter covers the following topics:

Syntax

```
ADASAV  SAVE  DELTA [ BUFNO = number-of-buffers | 1 ]  
                  [ DSIMDEV = device-type | ADARUN-device ]  
                  [ NOUSERABEND ]  
                  [ TEST ]  
                  [ TTSYN = seconds | ADARUN-tt ]  
                  [ TWOCOPIES ]
```

Optional Parameters

BUFNO: Count of Buffers Per Drive

The BUFNO value allocates fixed buffers for a SAVE DELTA operation. A value of 2 or 3 usually provides optimum performance; a value up to 255 is possible. A value greater than 5, however, provides little advantage and allocates a lot of space. The default is 1 (one buffer per drive).

DSIMDEV: DSIM Device Type

The DSIMDEV parameter specifies the device type of the DSIM dataset. This parameter is required only if the DSIM device type is different from that specified by the ADARUN DEVICE parameter (which is the default).

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

TTSYN: SYN2 Checkpoint Control

TTSYN allows the user to decrease the nucleus' ADARUN TT (maximum transaction time) during the synchronized checkpoint processing of the current ADASAV operation. The value specified is the approximate time in seconds (TT + 1.05 seconds), and must be less than the current ADARUN TT value. If TTSYN is not specified or if TTSYN is greater than the nucleus' TT value, that TT value becomes the default.

If the Adabas nucleus is active while ADASAV SAVE DELTA is running, a synchronized SYN2 checkpoint is taken at the end of the SAVE DELTA operation. This ensures that there is a point in time where all users are at ET status. If a user is not at ET status, no new transactions can be started for other users; they must wait until the SYN2 checkpoint can be taken.

The ADARUN TT value controls the maximum elapsed time permitted for a logical transaction. This is the maximum wait time until the SYN2 checkpoint can be processed. The ADASAV SAVE DELTA TTSYN parameter allows the user to decrease the TT value only during the synchronized checkpoint processing. The original TT value becomes effective again when ADASAV ends the SAVE DELTA operation.

TWOCOPIES: Create Two Copies of Output

TWOCOPIES creates two physical copies of the ADASAV output.

Example

```
ADASAV SAVE DELTA
ADASAV      TTSYN=10
ADASAV      DSIMDEV=8381
```

A Delta Save operation is to be performed, using a DSIM device type 8381. If running online, the ET synchronization at the end of the SAVE DELTA operation should last at most 10 seconds.

14

Restarting an Interrupted Save Operation

■ User ABEND 34 or 35	92
■ System ABEND or Other User ABEND	92
■ Resetting the DSIM Dataset	93

This section describes how an ADASAV SAVE database or SAVE DELTA execution can be restarted after failure. The specific actions to be performed depend on the type of failure.

This chapter covers the following topics:

User ABEND 34 or 35

► **If the ADASAV utility terminates with an error message and user abend 34 or 35:**

- 1 Determine and correct the cause of the error.
- 2 If the save operation was performed online, reset the DSIM dataset (see below).
- 3 Resubmit the save job.

System ABEND or Other User ABEND

If the ADASAV utility terminates abnormally with a system ABEND or a user ABEND other than 34 or 35:

► **For Offline Save Operation**

- 1 Determine and correct the cause of the error.
- 2 Remove the DIB entry of the ADASAV job, using ADADBS functions OPERCOM DDIB and RESETDIB.
- 3 Resubmit the save job.

► **For Online Save Operation**

- 1 Determine and correct the cause of the error.
- 2 Reset the nucleus online dump status, using the ADADBS OPERCOM RDUMPST command or the corresponding Adabas Online System function.
- 3 Delete the user queue element (UQE) of ADASAV in the nucleus, using the ADADBS functions OPERCOM DUQ and OPERCOM STOPU or the corresponding Adabas Online System functions.
- 4 Remove the DIB entry of the ADASAV job, using the ADADBS OPERCOM DDIB and RESETDIB commands, or the corresponding Adabas Online System functions.
- 5 Reset the DSIM dataset (see below).
- 6 Resubmit the save job.

Resetting the DSIM Dataset

After an interrupted online save operation, the DSIM dataset must be reset to prepare it for another save/copy/merge cycle.

This can be done using the ADAFRM function `DSIMRESET FROMRABN=1,SIZE=1B`. See [DSIM-RESET: Reset the DSIM Dataset](#).

Software AG recommends that you specify the DSIM dataset for exclusive use by the ADAFRM utility to avoid accidentally destroying information in a DSIM dataset currently in use by another utility. For the other utilities (ADASAV and ADARES), the DSIM dataset must be specified for shared-update use.

15

ADASAV JCL/JCS Requirements and Examples

▪ BS2000/OSD	96
▪ z/OS	104
▪ z/VM	110
▪ VSE/ESA and z/VSE	116

This section describes the job control information required to run the ADASAV functions for the Delta Save Facility with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSE systems, and shows examples of each of the job streams.



Note: When running with the optional Recovery Aid (RLOG), all temporary datasets must also be cataloged in the job control.

This chapter covers the following topics:

BS2000/OSD

Dataset	Link Name	Storage	More Information
Full save input(s)	DDRESTn	tape/disk	for RESTORE DELTA function
Full save input	DDFULL	tape/disk	for MERGE function
Delta save input(s)	DDDELn	tape/disk	for MERGE/RESTORE DELTA
Delta save images (DSIM)	DDDSIMR1	disk	required for online saves
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	required for SAVE/RESTORE
Work	DDWORKR1	disk	required if nucleus is inactive
Dual/multiple PLOG	DDPLOGRn	disk	optional for RESTORE DELTA
Dual/multiple CLOG	DDCLOGRn	disk	optional for RESTORE DELTA
Recovery log	DDRLOGR1	disk	required for recovery log
Full or delta save outputs	DDSAVEn	tape/disk	required for SAVE/SAVE DELTA /MERGE functions
Dual full or delta save outputs	DDDUALn	tape/disk	required for TWOCOPIES
ADARUN parameters	SYSDTA/DDCARD		<i>Adabas Operations</i> documentation
ADASAV parameters	SYSDTA/DDKARTE		
ADARUN messages	SYSOUT/DDPRINT		<i>Adabas Messages and Codes</i>
ADASAV messages	SYSLST/DDDRUCK		<i>Adabas Messages and Codes</i>

Merge Delta Save Output (ADASAV MERGE)

```

/ .MERGE          LOGON

/ MOD - JOB - OPTIONS  LISTING=YES

/ MOD - TEST         DUMP=YES

/ ASS - SYSLST       DO . SAV . MERGE . LST

/ ASS - SYSOUT       DO . SAV . MERGE . OUT

/ SET - JOB - STEP

/ ASS - SYSDTA       *SYSCMD

/ SET - FILE - LINK   DDASSOR1,
ADABAS . DB010 . ASSOR1, SUP=DISK( SHARE - UPD=YES)

/ SET - FILE - LINK   DDDSIMR1, ADABAS . DB010 . DSIM
, SUP=DISK( SHARE - UPD=YES)

/ SET - FILE - LINK   DDFULL  , ADABAS . DB010 . SAVE039 . PART1

/ SET - FILE - LINK   DDFULL01, ADABAS . DB010 . SAVE039 . PART2

/ SET - FILE - LINK   DDFULL02, ADABAS . DB010 . SAVE039 . PART3

/ SET - FILE - LINK   DDFULL03, ADABAS . DB010 . SAVE039 . PART4

/ SET - FILE - LINK   DDDEL1  , ADABAS . DB010 . MASTER . DELTA . OLD

/ SET - FILE - LINK   DDSAVE1 , ADABAS . DB010 . SAVE040 . PART1

/ SET - FILE - LINK   DDSAVE2 , ADABAS . DB010 . SAVE040 . PART2

/ SET - FILE - LINK   DDSAVE3 , ADABAS . DB010 . SAVE040 . PART3

/ SET - FILE - LINK   DDSAVE4 , ADABAS . DB010 . SAVE040 . PART4

/ SET - FILE - LINK   DDDUAL1 , ADABAS . DB010 . COPY040 . PART1

/ SET - FILE - LINK   DDDUAL2 , ADABAS . DB010 . COPY040 . PART2

/ SET - FILE - LINK   DDDUAL3 , ADABAS . DB010 . COPY040 . PART3

/ SET - FILE - LINK   DDDUAL4 , ADABAS . DB010 . COPY040 . PART4

```

```

/SET-FILE-LINK      DDLIB      , ADABAS.MOD

/REMARK

/START-PROGRAM      FROM-FILE=*MOD(ADABAS.MOD,ADARUN)

ADARUN      PROG=ADASAV,DSF=YES...

ADASAV MERGE PATTERN=FD

ADASAV      DRIVES=4

ADASAV      TWOCOPIES

/ASS-SYSLST        *PRIM

/ASS-SYSOUT        *PRIM

/ASS-SYSDTA        *PRIM

/LOGOFF            SYS-OUTPUT=DEL

```

Merge and Restore Database (ADASAV RESTORE DELTA)

```

/.REST            LOGON

/MOD-JOB-OPTIONS  LISTING=YES

/MOD-TEST         DUMP=YES

/ASS-SYSLST       DO.SAV.REST.LST

/ASS-SYSOUT       DO.SAV.REST.OUT

/SET-JOB-STEP

/ASS-SYSDTA       *SYSCMD

/SET-FILE-LINK    DDASS0R1,

ADABAS.DB010.ASS0R1,SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK    DDDATAR1,

ADABAS.DB010.DATAR1,SUP=DISK(SHARE-UPD=YES)

```

```

/SET-FILE-LINK      DDWORKR1, ADABAS.DB010.WORK, SUP=DISK( SHARE-UPD=YES)

/SET-FILE-LINK      DDPLOGR1,
ADABAS.DB010.PLOGR1, SUP=DISK( SHARE-UPD=YES)

/SET-FILE-LINK      DDPLOGR2,
ADABAS.DB010.PLOGR2, SUP=DISK( SHARE-UPD=YES)

/SET-FILE-LINK      DDCLOGR1,
ADABAS.DB010.CLOGR1, SUP=DISK( SHARE-UPD=YES)

/SET-FILE-LINK      DDCLOGR2,
ADABAS.DB010.CLOGR2, SUP=DISK( SHARE-UPD=YES)

/SET-FILE-LINK      DDDSIMR1, ADABAS.DB010.DSIM
, SUP=DISK( SHARE-UPD=YES)

/SET-FILE-LINK      DDREST1 , ADABAS.DB010.SAVE040.PART1

/SET-FILE-LINK      DDREST2 , ADABAS.DB010.SAVE040.PART2

/SET-FILE-LINK      DDREST3 , ADABAS.DB010.SAVE040.PART3

/SET-FILE-LINK      DDREST4 , ADABAS.DB010.SAVE040.PART4

/SET-FILE-LINK      DDDEL1  , ADABAS.DB010.MASTER.DELTA.OLD

/SET-FILE-LINK      DDDEL2  , ADABAS.DB010.MASTER.DELTA1

/SET-FILE-LINK      DDDEL201, ADABAS.DB010.MASTER.DELTA2

/SET-FILE-LINK      DDDEL202, ADABAS.DB010.MASTER.DELTA3

/SET-FILE-LINK      DDLIB   , ADABAS.MOD

/REMARK

/START-PROGRAM      FROM-FILE=*MOD( ADABAS.MOD, ADARUN)

ADARUN    PROG=ADASAV, DSF=YES...

ADASAV  RESTORE DELTA, OVERWRITE

ADASAV      PATTERN=FDD

ADASAV      DSIMDEV=2000

```

```

ADASAV          DRIVES=4

ADASAV          PLOGDEV=2000, CLOGDEV=2000

/ASS-SYSLST      *PRIM

/ASS-SYSOUT      *PRIM

/ASS-SYSDTA      *PRIM

/LOGOFF          SYS-OUTPUT=DEL

```

Restore Files (ADASAV RESTORE DELTA,FMOVE...)

```

/.REST          LOGON

/MOD-JOB-OPTIONS LISTING=YES

/MOD-TEST       DUMP=YES

/ASS-SYSLST     DO.SAV.REST.LST

/ASS-SYSOUT     DO.SAV.REST.OUT

/SET-JOB-STEP

/ASS-SYSDTA     *SYSCMD

/SET-FILE-LINK  DDASSOR1,

ADABAS.DB010.ASSOR1, SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK  DDDATAR1,

ADABAS.DB010.DATAR1, SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK  DDWORKR1, ADABAS.DB010.WORK

, SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK  DDDSIMR1, ADABAS.DB010.DSIM

, SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK  DDREST1 ,

```



```

ADABAS.DB010.SAVE040.PART1

/SET-FILE-LINK      DDREST01, ADABAS.DB010.SAVE040.PART2
/SET-FILE-LINK      DDREST02, ADABAS.DB010.SAVE040.PART3
/SET-FILE-LINK      DDREST03, ADABAS.DB010.SAVE040.PART4
/SET-FILE-LINK      DDDEL1  , ADABAS.DB010.MASTER.DELTA.OLD
/SET-FILE-LINK      DDDEL2  , ADABAS.DB010.MASTER.DELTA1
/SET-FILE-LINK      DDLIB   , ADABAS.MOD
/REMARK
/START-PROGRAM      FROM-FILE=*MOD(ADABAS.MOD,ADARUN)

ADARUN   PROG=ADASAV,DSF=YES...

ADASAV  RESTORE DELTA,OVERWRITE

ADASAV      FMOVE=10,20,30,...

/ASS-SYSLST      *PRIM
/ASS-SYSOUT      *PRIM
/ASS-SYSDTA      *PRIM
/LOGOFF          SYS-OUTPUT=DEL

```

Save Database (ADASAV SAVE)

```

/.SAVE          LOGON
/MOD-JOB-OPTIONS LISTING=YES
/MOD-TEST       DUMP=YES
/ASS-SYSLST     D0.SAV.LST
/ASS-SYSOUT     D0.SAV.OUT
/SET-JOB-STEP
/ASS-SYSDTA     *SYSCMD

```

```
/SET-FILE-LINK      DDASSOR1,
ADABAS.DB010.ASSOR1,SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK      DDDATAR1,
ADABAS.DB010.DATAR1,SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK      DDWORKR1, ADABAS.DB010.WORK
,SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK      DDDSIMR1, ADABAS.DB010.DSIM
,SUP=DISK(SHARE-UPD=YES)

/SET-FILE-LINK      DDSAVE1 , ADABAS.DB010.SAVE039.PART1
/SET-FILE-LINK      DDSAVE2 , ADABAS.DB010.SAVE039.PART2
/SET-FILE-LINK      DDSAVE3 , ADABAS.DB010.SAVE039.PART3
/SET-FILE-LINK      DDSAVE4 , ADABAS.DB010.SAVE039.PART4
/SET-FILE-LINK      DDLIB   , ADABAS.MOD

/REMARK

/START-PROGRAM      FROM-FILE=*MOD(ADABAS.MOD,ADARUN)

ADARUN   PROG=ADASAV,DSF=YES...

ADASAV  SAVE

ADASAV      TTSYN=10

ADASAV      DRIVES=4

/ASS-SYSLST      *PRIM
/ASS-SYSOUT      *PRIM
/ASS-SYSDTA      *PRIM
```

```
/LOGOFF          SYS-OUTPUT=DEL
```

Save Database (ADASAV SAVE DELTA)

```

/ .DELTA          LOGON
/ MOD-JOB-OPTIONS LISTING=YES
/ MOD-TEST        DUMP=YES
/ ASS-SYSLST       DO.SAV.DELTA.LST
/ ASS-SYSOUT       DO.SAV.DELTA.OUT
/ SET-JOB-STEP
/ ASS-SYSDTA       *SYSCMD
/ SET-FILE-LINK    DDASSOR1,
ADABAS.DB010.ASSOR1, SUP=DISK(SHARE-UPD=YES)
/ SET-FILE-LINK    DDDATAR1,
ADABAS.DB010.DATAR1, SUP=DISK(SHARE-UPD=YES)
/ SET-FILE-LINK    DDWORKR1, ADABAS.DB010.WORK
, SUP=DISK(SHARE-UPD=YES)
/ SET-FILE-LINK    DDDSIMR1, ADABAS.DB010.DSIM
, SUP=DISK(SHARE-UPD=YES)
/ SET-FILE-LINK    DDSAVE1 , ADABAS.DB010.SAVE
/ SET-FILE-LINK    DDLIB  , ADABAS.MOD
/ REMARK
/ START-PROGRAM    FROM-FILE=*MOD(ADABAS.MOD, ADARUN)
ADARUN  PROG=ADASAV, DSF=YES...
ADASAV  SAVE  DELTA
ADASAV   TTSYN=10

```

```
ADASAV      DSIMDEV=2000

/ASS-SYSLST      *PRIM

/ASS-SYSOUT      *PRIM

/ASS-SYSDTA      *PRIM

/LOGOFF          SYS-OUTPUT=DEL
```

z/OS

Dataset	DD Name	Storage	More Information
Full save input(s)	DDRESTn	tape/disk	for RESTORE DELTA
Full save input	DDFULL	tape/disk	for MERGE
Delta save input(s)	DDDELn	tape/disk	for MERGE/RESTORE DELTA
Delta Save images (DSIM)	DDDSIMR1	disk	required for online saves
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	required for SAVE/RESTORE
Work	DDWORKR1	disk	required for SAVE/RESTORE if nucleus is inactive
Dual/multiple PLOG	DDPLOGRn	disk	optional for RESTORE DELTA
Dual/multiple CLOG	DDCLOGRn	disk	optional for RESTORE DELTA
Recovery log	DDRLOGR1	disk	required for recovery log
Full or delta save outputs	DDSAVEN	tape/disk	required for SAVE/SAVE DELTA /MERGE functions
Dual full or delta save outputs	DDDUALn	tape/disk	required for TWOCOPIES
ADARUN parameters	DDCARD		<i>Adabas Operations</i> documentation
ADASAV parameters	DDKARTE		
ADARUN messages	DDPRINT		<i>Adabas Messages and Codes</i>
ADASAV messages	DDDRUCK		<i>Adabas Messages and Codes</i>

Merge Delta Save Output (ADASAV MERGE)

```

//MERGE      EXEC   PGM=ADARUN

//DDASSOR1 DD   DSN=ADABAS.DB010.ASSOR1, DISP=SHR

//DDDSIMR1 DD   DSN=ADABAS.DB010.DSIM, DISP=SHR

//DDFULL     DD   DSN=ADABAS.DB010.SAVE039.PART1, DISP=SHR

//           DD   DSN=ADABAS.DB010.SAVE039.PART2, DISP=SHR

//           DD   DSN=ADABAS.DB010.SAVE039.PART3, DISP=SHR

//           DD   DSN=ADABAS.DB010.SAVE039.PART4, DISP=SHR

//DDDEL1     DD   DSN=ADABAS.DB010.MASTER.DELTA.OLD, DISP=SHR


//DDSAVE1    DD

DSN=ADABAS.DB010.SAVE040.PART1, DISP=(NEW, CATLG),

//           UNIT=CASS, ...

//DDSAVE2    DD   DSN=ADABAS.DB010.SAVE040.PART2, DISP=(NEW, CATLG),

//           UNIT=CASS, ...

//DDSAVE3    DD   DSN=ADABAS.DB010.SAVE040.PART3, DISP=(NEW, CATLG),

//           UNIT=CASS, ...

//DDSAVE4    DD   DSN=ADABAS.DB010.SAVE040.PART4, DISP=(NEW, CATLG),

//           UNIT=CASS, ...

//DDDUAL1    DD   DSN=ADABAS.DB010.COPY040.PART1, DISP=(NEW, CATLG),

//           UNIT=CASS, ...

//DDDUAL2    DD   DSN=ADABAS.DB010.COPY040.PART2, DISP=(NEW, CATLG),

//           UNIT=CASS, ...

//DDDUAL3    DD   DSN=ADABAS.DB010.COPY040.PART3, DISP=(NEW, CATLG),

//           UNIT=CASS, ...

```

```
//DDDUAL4 DD DSN=ADABAS.DB010.COPY040.PART4,DISP=(NEW,CATLG),  
  
// UNIT=CASS, ...  
  
//DDCARD DD *  
  
ADARUN PROG=ADASAV,DSF=YES, ...  
  
//DDKARTE DD *  
  
ADASAV MERGE PATTERN=FD  
  
ADASAV DRIVES=4  
  
ADASAV TWOCOPIES  
  
//DDPRINT DD SYSOUT=*  
  
//DDDRUCK DD SYSOUT=*
```

Merge and Restore Database (ADASAV RESTORE DELTA)

```
//RESTORE EXEC PGM=ADARUN  
  
//DDASSOR1 DD DSN=ADABAS.DB010.ASSOR1,DISP=SHR  
  
//DDDATAR1 DD DSN=ADABAS.DB010.DATAR1,DISP=SHR  
  
//DDWORKR1 DD DSN=ADABAS.DB010.WORKR1,DISP=SHR  
  
//DDPLOGR1 DD DSN=ADABAS.DB010.PLOGR1,DISP=SHR  
  
//DDPLOGR2 DD DSN=ADABAS.DB010.PLOGR2,DISP=SHR  
  
//DDCLOGR1 DD DSN=ADABAS.DB010.CLOGR1,DISP=SHR  
  
//DDCLOGR2 DD DSN=ADABAS.DB010.CLOGR2,DISP=SHR  
  
//DDDSIMR1 DD DSN=ADABAS.DB010.DSIM,DISP=SHR  
  
//DDREST1 DD DSN=ADABAS.DB010.SAVE040.PART1,DISP=SHR  
  
//DDREST2 DD DSN=ADABAS.DB010.SAVE040.PART2,DISP=SHR  
  
//DDREST3 DD DSN=ADABAS.DB010.SAVE040.PART3,DISP=SHR  
  
//DDREST4 DD DSN=ADABAS.DB010.SAVE040.PART4,DISP=SHR
```

```
//DDDEL1 DD DSN=ADABAS.DB010.MASTER.DELTA.OLD, DISP=SHR

//DDDEL2 DD DSN=ADABAS.DB010.MASTER.DELTA1, DISP=SHR

// DD DSN=ADABAS.DB010.MASTER.DELTA2, DISP=SHR

// DD DSN=ADABAS.DB010.MASTER.DELTA3, DISP=SHR


//DDCARD DD *

ADARUN PROG=ADASAV, DSF=YES, ...

//DDKARTE DD *

ADASAV RESTORE DELTA, OVERWRITE

ADASAV PATTERN='FDD'

ADASAV DRIVES=4

ADASAV DSIMDEV=8381

ADASAV PLOGDEV=3390, CLOGDEV=3390

//DDPRINT DD SYSOUT=*

//DDDRUCK DD SYSOUT=*
```

Restore Files (ADASAV RESTORE DELTA,FMOVE...)

```
//RESTORE EXEC PGM=ADARUN

//DDASSOR1 DD DSN=ADABAS.DB010.ASSOR1, DISP=SHR

//DDDATAR1 DD DSN=ADABAS.DB010.DATAR1, DISP=SHR

//DDWORKR1 DD DSN=ADABAS.DB010.WORKR1, DISP=SHR

//DDREST1 DD DSN=ADABAS.DB010.SAVE040.PART1, DISP=SHR

// DD DSN=ADABAS.DB010.SAVE040.PART2, DISP=SHR

// DD DSN=ADABAS.DB010.SAVE040.PART3, DISP=SHR

// DD DSN=ADABAS.DB010.SAVE040.PART4, DISP=SHR
```

```
//DDDEL1   DD   DSN=ADABAS.DB010.MASTER.DELTA.OLD, DISP=SHR

//DDDEL2   DD   DSN=ADABAS.DB010.MASTER.DELTA1, DISP=SHR

//DDCARD   DD   *

ADARUN    . . .

//DDKARTE  DD   *

ADASAV  RESTORE DELTA, OVERWRITE

ADASAV          FMOVE=10, 20, 30, . . .

//DDPRINT   DD   SYSOUT=*

//DDDRUCK   DD   SYSOUT=*
```

Save Database (ADASAV SAVE)

```
//SAVE      EXEC   PGM=ADARUN

//DDASS0R1  DD   DSN=ADABAS.DB010.ASS0R1, DISP=SHR

//DDDATAR1  DD   DSN=ADABAS.DB010.DATAR1, DISP=SHR

//DDWORKR1  DD   DSN=ADABAS.DB010.WORKR1, DISP=SHR

//DDDSIMR1  DD   DSN=ADABAS.DB010.DSIM, DISP=SHR

//DDSAVE1   DD   DSN=ADABAS.DB010.SAVE039.PART1, DISP=(NEW, CATLG)

//          UNIT=CASS, . . .

//DDSAVE2   DD   DSN=ADABAS.DB010.SAVE039.PART2, DISP=(NEW, CATLG)

//          UNIT=CASS, . . .

//DDSAVE3   DD   DSN=ADABAS.DB010.SAVE039.PART3, DISP=(NEW, CATLG)

//          UNIT=CASS, . . .

//DDSAVE4   DD   DSN=ADABAS.DB010.SAVE039.PART4, DISP=(NEW, CATLG)

//          UNIT=CASS, . . .

//DDCARD    DD   *
```



```
ADARUN  PROG=ADASAV,DSF=YES, . . .

//DDKARTE  DD  *

ADASAV  SAVE

ADASAV           DRIVES=4

ADASAV           TTSYN=10

//DDPRINT  DD  SYSOUT=*

//DDDRUCK  DD  SYSOUT=*
```

Save Database (ADASAV SAVE DELTA)

```
//SAVE      EXEC  PGM=ADARUN

//DDASSOR1 DD  DSN=ADABAS.DB010.ASSOR1,DISP=SHR

//DDDATAR1 DD  DSN=ADABAS.DB010.DATAR1,DISP=SHR

//DDWORKR1 DD  DSN=ADABAS.DB010.WORKR1,DISP=SHR

//DDDSIMR1 DD  DSN=ADABAS.DB010.DSIM,DISP=SHR

//DDSAVE1  DD  DSN=ADABAS.DB010.DELTA,DISP=OLD

//DDCARD   DD  *

ADARUN  PROG=ADASAV,DSF=YES, . . .

//DDKARTE  DD  *

ADASAV  SAVE  DELTA

ADASAV           TTSYN=10

ADASAV           DSIMDEV=8381

//DDPRINT  DD  SYSOUT=*
```

```
//DDDRUCK DD SYSOUT=*
```

z/VM

Dataset	DD Name	Storage	More Information
Full save input(s)	DDRESTn	tape/disk	for RESTORE DELTA
Full save input	DDFULL	tape/disk	for MERGE
Delta save input(s)	DDDELn	tape/disk	for MERGE/RESTORE DELTA
Delta Save images (DSIM)	DDDSIMR1	disk	required for online saves
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	required for SAVE/RESTORE
Work	DDWORKR1	disk	required if nucleus is inactive
Dual/multiple PLOG	DDPLOGRn	disk	optional for RESTORE DELTA
Dual/multiple CLOG	DDCLOGRn	disk	optional for RESTORE DELTA
Recovery log	DDRLOGR1	disk	required for recovery log
Full or delta save outputs	DDSAVEn	tape/disk	required for SAVE/SAVE DELTA /MERGE functions
Dual full or delta save outputs	DDDUALn	tape/disk	required for TWOCOPIES
ADARUN parameters	DDCARD	disk/ terminal/ reader	<i>Adabas Operations</i> documentation
ADASAV parameters	DDKARTE	disk/ terminal/ reader	
ADARUN messages	DDPRINT	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>
ADASAV messages	DDDRUCK	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>

Merge Delta Save Output (ADASAV MERGE)

```
DATADEF DDASSOR1, DSN=ADABASVv . ASSO, VOL=ASSOV1

DATADEF DDDSIMR1, DSN=ADABASVv . DSIM, VOL=DSIM

DATADEF DDFULL, DSN=ADABASVv . SAVE039 . PART1, MODE=A

DATADEF DDFULL, DSN=ADABASVv . SAVE039 . PART2, MODE=A, CONCAT=1

DATADEF DDFULL, DSN=ADABASVv . SAVE039 . PART2, MODE=A, CONCAT=2

DATADEF DDFULL, DSN=ADABASVv . SAVE039 . PART2, MODE=A, CONCAT=3

DATADEF DDDEL1, DSN=ADABASVv . MASTER . DELTA, MODE=A
```

```

DATADEF DDSAVE1, DSN=ADABASVv . SAVE040 . PART1, MODE=A
DATADEF DDSAVE2, DSN=ADABASVv . SAVE040 . PART2, MODE=A
DATADEF DDSAVE3, DSN=ADABASVv . SAVE040 . PART3, MODE=A
DATADEF DDSAVE4, DSN=ADABASVv . SAVE040 . PART4, MODE=A
DATADEF DDDUAL1, DSN=ADABASVv . COPY040 . PART1, MODE=A
DATADEF DDDUAL2, DSN=ADABASVv . COPY040 . PART2, MODE=A
DATADEF DDDUAL3, DSN=ADABASVv . COPY040 . PART3, MODE=A
DATADEF DDDUAL4, DSN=ADABASVv . COPY040 . PART4, MODE=A
DATADEF DDPRINT, DSN=ADASAV . DDPRINT, MODE=A
DATADEF DDDRUCK, DSN=ADASAV . DDDRUCK, MODE=A
DATADEF DDCARD, DSN=RUNSAV . CONTROL, MODE=A
DATADEF DDKARTE, DSN=ADASAV . CONTROL, MODE=A
ADARUN

```

Contents of RUNSAV CONTROL A1:

```

ADARUN  PROG=ADASAV, DSF=YES . . .

```

Contents of ADASAV CONTROL A1:

```

ADASAV  MERGE  PATTERN='FD'

ADASAV          DRIVES=4

```

ADASAV	TWOCOPIES
--------	-----------

Merge and Restore Database (ADASAV RESTORE DELTA)

```
DATADef DDASS0R1,DSN=ADABASVv.ASS0,VOL=ASS0V1
DATADef DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADef DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1
DATADef DDCLOGR1,DSN=ADABASVv.CLOG1,VOL=CLOGV1
DATADef DDCLOGR2,DSN=ADABASVv.CLOG2,VOL=CLOGV2
DATADef DDDSIMR1,DSN=ADABASVv.DSIM,VOL=DSIM
DATADef DDREST1,DSN=ADABASVv.SAVE040.PART1,MODE=A
DATADef DDREST2,DSN=ADABASVv.SAVE040.PART2,MODE=A
DATADef DDREST3,DSN=ADABASVv.SAVE040.PART3,MODE=A
DATADef DDREST4,DSN=ADABASVv.SAVE040.PART4,MODE=A
DATADef DDDEL1,DSN=ADABASVv.DELTA1,MODE=A
DATADef DDDEL2,DSN=ADABASVv.DELTA2,MODE=A
DATADef DDDEL2,DSN=ADABASVv.DELTA3,MODE=A,CONCAT=1
DATADef DDDEL2,DSN=ADABASVv.DELTA4,MODE=A,CONCAT=2
DATADef DDPRINT,DSN=ADASAV.DDPRINT,MODE=A
DATADef DDDRUCK,DSN=ADASAV.DDDRUCK,MODE=A
DATADef DDCARD,DSN=RUNSAV.CONTROL,MODE=A
DATADef DDKARTE,DSN=ADASAV.CONTROL,MODE=A
```

ADARUN

Contents of RUNSAV CONTROL A1:

```
ADARUN  PROG=ADASAV,DSF=YES...
```

Contents of ADASAV CONTROL A1:

```
ADASAV  RESTORE  DELTA,OVERWRITE

ADASAV      PATTERN='FDD'

ADASAV      DSIMDEV=2000

ADASAV      DRIVES=4

ADASAV      PLOGDEV=...,CLOGDEV=...
```

Restore Files (ADASAV RESTORE DELTA,FMOVE...)

```
DATADEF  DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1

DATADEF  DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1

DATADEF  DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1

DATADEF  DDDSIMR1,DSN=ADABASVv.DSIM,VOL=DSIM

DATADEF  DDREST1,DSN=ADABASVv.SAVE040.PART1,MODE=A

DATADEF  DDREST1,DSN=ADABASVv.SAVE040.PART2,MODE=A,CONCAT=1

DATADEF  DDREST1,DSN=ADABASVv.SAVE040.PART3,MODE=A,CONCAT=2

DATADEF  DDREST1,DSN=ADABASVv.SAVE040.PART4,MODE=A,CONCAT=3

DATADEF  DDDEL1,DSN=ADABASVv.DELTA1,MODE=A

DATADEF  DDDEL2,DSN=ADABASVv.DELTA2,MODE=A
```

```
DATADEF DDPRINT,DSN=ADASAV.DDPRINT,MODE=A
DATADEF DDDRUCK,DSN=ADASAV.DDDRUCK,MODE=A
DATADEF DDCARD,DSN=RUNSAV.CONTROL,MODE=A
DATADEF DDKARTE,DSN=ADASAV.CONTROL,MODE=A
ADARUN
```

Contents of RUNSAV CONTROL A1:

```
ADARUN PROG=ADASAV,DSF=YES...
```

Contents of ADASAV CONTROL A1:

```
ADASAV RESTORE DELTA,OVERWRITE
ADASAV          FMOVE=10,20,30,...
```

Save Database (ADASAV SAVE)

```
DATADEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADEF DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1
DATADEF DDASSOR1,DSN=ADABASVv.ASS0,VOL=ASSOV1
DATADEF DDDSIMR1,DSN=ADABASVv.DSIM,VOL=DSIM
DATADEF DDSAVE1,DSN=ADABASVv.SAVE040.PART1,MODE=A
DATADEF DDSAVE2,DSN=ADABASVv.SAVE040.PART2,MODE=A
DATADEF DDSAVE3,DSN=ADABASVv.SAVE040.PART3,MODE=A
DATADEF DDSAVE4,DSN=ADABASVv.SAVE040.PART4,MODE=A
DATADEF DDPRINT,DSN=ADASAV.DDPRINT,MODE=A
```

```

DATADEF DDDRUCK, DSN=ADASAV . DDDRUCK, MODE=A

DATADEF DDCARD, DSN=RUNSAV . CONTROL, MODE=A

DATADEF DDKARTE, DSN=ADASAV . CONTROL, MODE=A

ADARUN

```

Contents of RUNSAV CONTROL A1:

```

ADARUN  PROG=ADASAV, DSF=YES . . .

```

Contents of ADASAV CONTROL A1:

```

ADASAV  SAVE

      ADASAV      DRIVES=4

      ADASAV      TTSYN=10

```

Save Database (ADASAV SAVE DELTA)

```

DATADEF DDDATAR1, DSN=ADABASVv . DATA, VOL=DATAV1

DATADEF DDWORKR1, DSN=ADABASVv . WORK, VOL=WORKV1

DATADEF DDASSOR1, DSN=ADABASVv . ASSO, VOL=ASSOV1

DATADEF DDDSIMR1, DSN=ADABASVv . DSIM, VOL=DSIM

DATADEF DDSAVE1, DSN=ADABASVv . SAVE, MODE=A

DATADEF DDPRINT, DSN=ADASAV . DDPRINT, MODE=A

DATADEF DDDRUCK, DSN=ADASAV . DDDRUCK, MODE=A

DATADEF DDCARD, DSN=RUNSAV . CONTROL, MODE=A

DATADEF DDKARTE, DSN=ADASAV . CONTROL, MODE=A

```

ADARUN

Contents of RUNSAV CONTROL A1:

```
ADARUN  PROG=ADASAV, DSF=YES...
```

Contents of ADASAV CONTROL A1:

```
ADASAV  SAVE  DELTA

ADASAV      TTSYN=10

ADASAV      DSIMDEV=8381
```

VSE/ESA and z/VSE

Dataset	Symbolic	Storage	Logical Unit	More Information
Full save input(s)	RESTn	tape/disk		for RESTORE DELTA
Full save input	FULL	tape disk	SYS030 *	for MERGE
Delta save input(s)	DELn	tape disk	SYS031- SYS038 *	for MERGE or RESTORE DELTA
Delta Save images (DSIM)	DSIMR1	disk		required for online saves
Associator	ASSORn	disk	*	
Data Storage	DATARn	disk		required for SAVE or RESTORE
Work	WORKR1	disk		required for SAVE or RESTORE if nucleus is inactive
Dual/multiple PLOG	PLOGRn	disk	*	optional for RESTORE DELTA
Dual/multiple CLOG	CLOGRn	disk	*	optional for RESTORE DELTA
Recovery log	RLOGR1	disk		required for recovery log
Full or delta save outputs	SAVEN	tape disk	SYS011- SYS018 *	required for SAVE, SAVE DELTA or MERGE

Dataset	Symbolic	Storage	Logical Unit	More Information
Dual full or delta save outputs	DUALn	tape disk	SYS021- SYS028 *	required if TWOCOPIES is specified
ADARUN parameters	CARD	disk tape reader	* SYS000 SYSIPT	<i>Adabas Operations</i> documentation
ADASAV parameters		reader	SYSIPT	
ADARUN messages		printer	SYSLST	<i>Adabas Messages and Codes</i>
ADASAV messages		printer	SYS009	<i>Adabas Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information in section [VSE/ESA Job Control for Delta Save](#).

For information about using ZAPs to change VSE/ESA logical units, see the *Adabas Installation* documentation.

Merge Delta Save Output (ADASAV MERGE)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// ASSGN SYS030,TAPE
// TLBL FULL, 'EXAMPLE.DB99.SAVE01'
// TLBL FULL01, 'EXAMPLE.DB99.SAVE02'
// TLBL FULL02, 'EXAMPLE.DB99.SAVE03'
// TLBL FULL03, 'EXAMPLE.DB99.SAVE04'
// ASSGN SYS031,TAPE
// TLBL DEL1, 'EXAMPLE.DB99.DEL1'
// ASSGN SYS011,TAPE
// TLBL SAVE1, 'EXAMPLE.DB99.SAVE1'
// ASSGN SYS012,TAPE
// TLBL SAVE2, 'EXAMPLE.DB99.SAVE2'
// ASSGN SYS013,TAPE
```

```
// TLBL SAVE3, 'EXAMPLE.DB99.SAVE3'

// ASSGN SYS014,TAPE

// TLBL SAVE4, 'EXAMPLE.DB99.SAVE4'

// ASSGN SYS021,TAPE

// TLBL DUAL1, 'EXAMPLE.DB99.DUAL1'

// ASSGN SYS022,TAPE

// TLBL DUAL2, 'EXAMPLE.DB99.DUAL2'

// ASSGN SYS023,TAPE

// TLBL DUAL3, 'EXAMPLE.DB99.DUAL3'

// ASSGN SYS024,TAPE

// TLBL DUAL4, 'EXAMPLE.DB99.DUAL4'

// EXEC ADARUN,SIZE=ADARUN

ADARUN PROG=ADASAV,DSF=YES,...

/*

ADASAV MERGE PATTERN='FD'

ADASAV      DRIVES=4

ADASAV      TWOCOPIES

/*
```

Merge and Restore Database (ADASAV RESTORE DELTA)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// ASSGN SYS031,TAPE
// TLBL DEL1, 'EXAMPLE.DB99.DEL1'
// ASSGN SYS032,TAPE
// TLBL DEL2, 'EXAMPLE.DB99.DEL2'
// TLBL DEL201, 'EXAMPLE.DB99.DEL3'
// TLBL DEL202, 'EXAMPLE.DB99.DEL4'
// ASSGN SYS011,TAPE
// TLBL REST1, 'EXAMPLE.DB99.SAVE1'
// ASSGN SYS012,TAPE
// TLBL REST2, 'EXAMPLE.DB99.SAVE2'
// ASSGN SYS013,TAPE
// TLBL REST3, 'EXAMPLE.DB99.SAVE3'
// ASSGN SYS014,TAPE
// TLBL REST4, 'EXAMPLE.DB99.SAVE4'
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADASAV,DSF=YES,...
/*
ADASAV RESTORE DELTA,OVERWRITE
ADASAV          DRIVES=4
ADASAV          DSIMDEV=8381
ADASAV          PATTERN='FDD'
ADASAV          PLOGDEV=3390,CLOGDEV=3390
```

```
/*
```

Restore Files (ADASAV RESTORE DELTA,FMOVE...)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// ASSGN SYS031,TAPE
// TLBL DEL1, 'EXAMPLE.DB99.DEL1'
// ASSGN SYS032,TAPE
// TLBL DEL2, 'EXAMPLE.DB99.DEL2'
// ASSGN SYS011,TAPE
// TLBL REST1, 'EXAMPLE.DB99.SAVE1'
// TLBL REST101, 'EXAMPLE.DB99.SAVE2'
// TLBL REST102, 'EXAMPLE.DB99.SAVE3'
// TLBL REST103, 'EXAMPLE.DB99.SAVE4'
// EXEC ADARUN, SIZE=ADARUN

ADARUN PROG=ADASAV, DSF=YES, ...

/*

ADASAV RESTORE DELTA, OVERWRITE

ADASAV          FMOVE=...
```

```
/*
```

Save Database (ADASAV SAVE)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// ASSGN SYS011,TAPE
// TLBL SAVE1, 'EXAMPLE.DB99.SAVE1'
// EXEC ADARUN, SIZE=ADARUN
ADARUN PROG=ADASAV, DSF=YES, ...
/*
ADASAV SAVE
ADASAV      DRIVES=4
/*
```

Save Database (ADASAV SAVE DELTA)

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// ASSGN SYS011,TAPE
// TLBL SAVE1, 'EXAMPLE.DB99.SAVE1'
// EXEC ADARUN, SIZE=ADARUN
ADARUN PROG=ADASAV, DSF=YES, ...
/*
ADASAV SAVE DELTA
ADASAV      TTSYN=10
```

```
ADASAV      DSIMDEV=8381, . . .
```

```
/*
```

16

ADAULD Utility

■ UNLOAD FILE: Unload Specified File from a Save Tape	125
■ Save Tape Input Processing	128
■ ADAULD Output Processing	130
■ ADAULD User Exit 9	131

The ADAULD utility can unload an Adabas file from a full save tape together with 1 to 8 delta save tapes and optionally a DSIM dataset. The save tapes may have been created online or offline using any version of the Delta Save Facility. Adabas files are unloaded from a combination of full or delta save tapes to reestablish the files from archive save tapes and load them into a database.



Note: It may not be possible to do this using the ADASAV RESTORE/RESTONL function if the archived database resided on device types that are no longer in use in the data cluster.

The records are unloaded in physical sequence; that is, in the order in which they are physically positioned within Data Storage.

The unloaded record output is in compressed format. The output records have the same format as the records produced by the Adabas ADACMP utility.

When using the MODE=SHORT option, descriptor entries (which are required to create the normal index and upper index for the file) are omitted during the unload process. This reduces the time required for unloading. Note, however, that output created using MODE=SHORT has a different FDT from the same file unloaded without MODE=SHORT, since all descriptor information is removed.

In general, the Adabas nucleus does not need to be active while unloading a file from a save tape.



Note: An interrupted ADAULD UNLOAD FILE run must be reexecuted from the beginning.

This chapter covers the following topics:

- **UNLOAD FILE: Unload Specified File from a Save Tape**
- **Save Tape Input Processing**
- **ADAULD Output Processing**
- **ADAULD User Exit 9**
- **JCL/JCS Requirements and Examples**

UNLOAD FILE: Unload Specified File from a Save Tape

Syntax

```

ADAULD [UNLOAD] FILE = file-number
SAVETAPE
[ CODE = cipher-key ]
[ DDISN ]
[ DSIMDEV = device-type | ADARUN-device ]
[ MODE = SHORT ]
[ NOUSERABEND ]
[ NUMOUT = 1 | 2 ]
[ NUMREC = number ]
[ PATTERN = merge-pattern ]
[TEST]

```

Essential Parameters

FILE

FILE specifies the number of the file to be unloaded. Neither the checkpoint file nor the security file can be unloaded.

SAVETAPE

SAVETAPE is used to unload a file from a full save tape together with 1 to 8 delta save tapes and optionally, a DSIM dataset. This is useful when moving a file from a save tape with one blocksize to a database with another, or when using a file from a save tape in different test environments.

If the file to be unloaded from the save tape is ciphered, the CODE parameter must be specified as usual.

User exit 9 can be used to select records for a particular client of a multiclient file. For more information, see the section [ADAULD User Exit 9](#).

For more information, see the section [Save Tape Input Processing](#).

Optional Parameters

CODE: Cipher Code

If the file to be unloaded is ciphered, CODE *must* supply the appropriate cipher code.

DDISN: Create DD/ISN Output File of Unloaded ISNs

Specifying the DDISN parameter instructs ADAULD to write the list of unloaded ISNs to the sequential output file DD/ISN. DD/ISN is structured so that it can be used as input to ADALOD UPDATE for the purpose of deleting the unloaded records.

If the DDISN keyword is specified but the DD/ISN file is missing in the JCL, ADAULD terminates with error-081.

DSIMDEV: DSIM Device Type

The DSIMDEV parameter specifies the device type of the DSIM dataset. This parameter is required only if the DSIM device type is different from that specified by the ADARUN DEVICE parameter (which is the default).

MODE=SHORT: Exclude Descriptor Information

This parameter indicates whether the descriptor information used to build the normal index and upper index are to be included in the output.

If MODE=SHORT is specified, no descriptor information will be unloaded, and all descriptor information is stripped from the field definition table (FDT) when it is written to the output dataset.

If the output is to be used as direct input to the ADALOD utility, the file will have no descriptors.

NOUSERABEND: Termination without ABEND

When an error is encountered while the function is running, the utility prints an error message and terminates with user ABEND 34 (with a dump) or user ABEND 35 (without a dump).

If NOUSERABEND is specified, the utility will *not* ABEND after printing the error message. Instead, the message "utility TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

NUMOUT: Number of Output Files

NUMOUT specifies the number of output files to be produced. If the number is greater than one, user exit 9 must be used to control DDOUT1 and 2 or OUT1 and 2 output file selection. For additional information, see the *Adabas DBA Tasks* documentation. Permitted values are 1 (default) and 2.

NUMREC: Number of Records to Be Unloaded

NUMREC limits the number of records to be unloaded. No limit will be in effect if the parameter is omitted.

PATTERN: Merge Input Pattern

The PATTERN parameter can be specified if the save tape from which the file is to be unloaded is supplied as a full save tape plus 1 to 8 delta save tapes. The parameter serves as a cross-check between the intended save input and the actual save datasets specified by the job control.

PATTERN specifies the count and type of input datasets to the merge operation as single letters "F" (full save tape) or "D" (delta save tape). Only one "F" can be specified in the pattern field, but up to eight "D"s can be entered-one for each existing delta save tape to be merged.

The pattern starts with an "F" if a full save input is specified, and continues (or starts) with a "D" for every delta save input. For example, PATTERN=FDD specifies that the input comprises one full save dataset and two delta save datasets. No special indicator is given for a save dataset being online.

The pattern string must exactly match the input save datasets specified in the job control.

TEST: Test Syntax

This parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

Examples**Example 1:**

```
ADAULD FILE=6, SAVETAPE
```

File 6 is to be unloaded from a full save tape. The records are to be unloaded in the sequence in which they are physically positioned on the tape.

Example 2:

```
ADAULD FILE=6, SAVETAPE, MODE=SHORT
```

File 6 is to be unloaded in physical sequence from a full save tape. The entries used to create the normal index and upper index are not to be unloaded. All descriptor information is removed from the field definition table (FDT) in the output.

Example 3:

```
ADAULD FILE=6, SAVETAPE, PATTERN=FDD
```

File 6 is to be unloaded in physical sequence from a full save tape and two delta save tapes. The unloaded file will have the same contents as it had at the time of the second delta save operation.

Save Tape Input Processing

ADAULD is used to unload an Adabas file from a full save tape together with 1 to 8 delta save tapes and optionally, a DSIM dataset:

- the full save is expected as a DD/FULL sequential input file;
- the delta saves are expected as DD/DEL1-DEL8 sequential input files; and
- the DSIM dataset as DD/DSIMR1. The DSIM dataset is required if any delta save tape specified is an online tape that is not accompanied by its unloaded DSIM dataset.

Save tapes created using any version of the Delta Save Facility can be used.

The ADARUN DBID specified for the ADAULD run must match the DBID found on the save tapes. The ADARUN parameter DSF=YES must be specified for the ADAULD run.

If the file has hyperdescriptors defined, the corresponding hyperexits must be specified in the ADARUN parameters for ADAULD. If the hyperexit routines are no longer available, the file must be unloaded with MODE=SHORT specified. See the *Adabas DBA Tasks* documentation for more information about hyperexits.

The ADAULD utility protocol on DD/DRUCK displays a short header indicating the kind of save tapes encountered, when each was created, the version of ADASAV used to create it, the database ID found on the save tape, and possibly the delta save ID of the save tapes when merged:

```
A D A U L D  v7.1   SM1   DBID = 00200   STARTED           YYYY-MM-DD  13:33:25
```

```
PARAMETERS:
```

```
-----
```

```
ADAULD UNLOAD FILE=3; SAVETAPE
```

```
*****
```

```

*
* UNLOAD FROM          ONLINE DATABASE SAVE
*
* CREATED AT           1997-04-23
*
* BY ADASAV VERSION    v71
*
* DBID                  200
*
* DSID                  1 / 0 / 1997-04-23 13:32:19
*
* PLOG SESSION NR      17
*
* SYN1/4 BLOCK NR      137
*
*
*****

```

ADAULD first reads the file control block (FCB) and file definition table (FDT) from the save tape. Then:

- for full save tapes and merged full save tapes, ADAULD scans the tape to find the file's Data Storage RABNs, extracts the Data Storage records, and for each Data Storage record, generates the descriptor values according to the FDT.
- for full save tapes plus delta save tapes and optional DSIM dataset, ADAULD calls the Delta Save merge facility to receive the latest version of each affected block. ADARUN DSF=YES must be specified so that the ADADSFS module containing the merge routines is loaded.

DSIM Dataset or the Protection Log

In order to unload a full save tape and one or more delta save tapes, one of which is online, you *must* supply the DSIM dataset associated with the remaining online save. If the DSIM dataset no longer exists, it is necessary to rebuild it from the protection log (PLOG) using ADARES COPY.

If an online full save tape *without* delta save tapes is unloaded, you have the option of providing either the associated DSIM dataset or the protection log (PLOG) as of the time of the online save. If the full save is not the most recent online save, it is necessary to rebuild the DSIM dataset from the PLOG using ADARES COPY. In this case, it is preferable to supply the PLOG directly to the unload. See the document *ADAULD* in the *Adabas Utilities* documentation for more information.

You can ensure that you have the DSIM datasets you need by unloading the original DSIM dataset to a sequential file using the **ADASAV MERGE** function. The unloaded DSIM dataset can be

used wherever the original DSIM would be used. In addition, several unloaded DSIM datasets can be used in a merge, restore, or unload operation, whereas only one original DSIM can be supplied to any of these operations.

ADAULD Output Processing

ADAULD unloads the records in the specified sequence. The unloaded records are written to one of two sequential datasets: DD/OUT1 and DD/OUT2. Writing to these output datasets is controlled by user exit 9.

The records output are identical in format to the output produced by the ADACMP utility unless the MODE=SHORT option is used, in which case the descriptor entries required for the normal index and upper index are omitted and the descriptor information is removed from the Adabas FDT. The ISN of the record immediately precedes the compressed data record, and is provided as a four-byte binary number.

Specifying the DDISN parameter instructs ADAULD to write the list of unloaded ISNs to a sequential output file DD/ISN. Only one DD/ISN file is created, containing the superset of ISNs written to either or both DD/OUT1 and DD/OUT2. ISNs that are rejected by userexit 9 are not written to DD/ISN.

DD/ISN is structured so that it can be used as input to ADALOD UPDATE for the purpose of deleting the unloaded records.

The number of ISNs written to DD/ISN is displayed in the ADAULD statistics on the DD/DRUCK utility protocol:

```
A D A U L D   STATISTICS
-----

NUMBER OF OUTPUT DATA SETS      = 1
NUMBER OF REQUESTED RECORDS     = 16777215
STARTISN                        = 0
OPTIONS                         = DVT
UNLOAD SEQUENCE                 = PHYS SEQ
```

NUMBER OF RECORDS READ	=	1000
NUMBER OF RECORDS WRITTEN	=	1000
RECORDS WRITTEN TO DDOUT1	=	1000
RECORDS WRITTEN TO DDOUT2	=	0
RECORDS REJECTED BY USEREXIT-9	=	0
NUMBER OF ISNS WRITTEN TO DDISN	=	1000

The number of ISNs written to DD/ISN should always be the number of records read minus the number of records rejected by user exit 9.

ADAULD User Exit 9

User exit 9 is called (when present) for each record selected before writing the record to the output dataset. The user exit is supplied with the record address, and returns an action code as follows:

1	write record to DD/OUT1
2	write record to DD/OUT2
3	write record to DD/OUT1 and DD/OUT2
I	ignore this record

The above datasets must have the same blocksize. See the *Adabas DBA Tasks* documentation for more information about user exits.

17

ADAULD JCL/JCS Requirements and Examples

▪ BS2000/OSD	134
▪ z/OS	135
▪ z/VM	136
▪ VSE/ESA and z/VSE	138

This section describes the job control information required to run ADAULD with BS2000/OSD, z/OS, z/VM, VSE/ESA and z/VSE systems and shows examples of each of the job streams.

This chapter covers the following topics:

BS2000/OSD

Dataset	Link Name	Storage	More Information
Unloaded data	DDOUT1	tape/disk	Output by ADAULD (see note)
Unloaded data	DDOUT2	tape/disk	Output by ADAULD (see note)
Unloaded ISNs	DDISN	tape/disk	Required with DDISN
Full save tape	DDFULL	tape/disk	Required for full save tapes
Delta save tape(s)	DDDEL1 - DDDEL8	tape/disk	Required for delta save tape(s)
Delta Save images	DDDSIMR1	disk	Required for DSIM dataset
Recovery log (RLOG)	DDRLOGR1	disk	Required for ADARAI
ADARUN parameters	SYSDTA/DDCARD		<i>Adabas Operations</i>
ADAULD parameters	SYSDTA/DDKARTE		
ADARUN messages	SYSOUT/DDPRINT		<i>Adabas Messages and Codes</i>
ADAULD messages	SYSLST/DDDRUCK		<i>Adabas Messages and Codes</i>



Note: DDOUT1 and DDOUT2 must have the same block size; otherwise, an ADAULD error will occur. DDOUT2 is required only if NUMOUT=2 is specified.

BS2000/OSD Examples

```

/.ADAULD LOGON

/OPTION MSG=FH,DUMP=YES

/REMARK *

/REMARK *      A D A U L D      SAVETAPE FUNCTION

/REMARK *

/SYSFILE SYSLST=L.ULD

/FILE     ADA.MOD      ,LINK=DDLIB

/FILE     ADA99.OUT1    ,LINK=DDOUT1 ,SPACE=(480,48)

```

```

/FILE    ADA99 . FULL      , LINK=DDFULL
/FILE    ADA99 . DEL1      , LINK=DDDEL1
/FILE    ADA99 . DEL2      , LINK=DDDEL2

/EXEC (ADARUN, ADA . MOD)

ADARUN  PROG=ADAULD, DB=99, DSF=YES

ADAULD  FILE=1, SAVETAPE, PATTERN=FDD

/LOGOFF  NOSPOOL

```

z/OS

Dataset	DD Name	Storage	More Information
Unloaded data	DDOUT1	tape/disk	Output by ADAULD (see note)
Unloaded data	DDOUT2	tape/disk	Output by ADAULD (see note)
Unloaded ISNs	DDISN	tape/disk	Required with DDISN
Full save tape	DDFULL	tape/disk	Required for full save tape
Delta save tape(s)	DDDEL1- DDDEL8	tape/disk	Required for delta save tape(s)
Delta Save images	DDDSIMR1	disk	Required for DSIM dataset
Recovery log (RLOG)	DDRLOGR1	disk	Required for ADARAI
ADAULD messages	DDDRUCK	printer	<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT	printer	<i>Adabas Messages and Codes</i>
ADARUN parameters	DDCARD	reader	<i>Adabas Operations</i>
ADAULD parameters	DDKARTE	reader	



Note: DDOUT1 and DDOUT2 must have the same block size; otherwise, an ADAULD error will occur. DDOUT2 is required only if NUMOUT=2 is specified.

z/OS Example

```
//ULD          EXEC  PGM=ADARUN

//STEPLIB      DD    DISP=SHR,DSN=ADABAS.Vvrs.LOADLIB

//*

//DDOUT1       DD    DISP=(,KEEP),DSN=EXAMPLE.ADA99.OUT1,

//              UNIT=SYSDA,VOL=SER=DISK01,SPACE=(TRK,200,RLSE)

//DDFULL       DD    DISP=SHR,DSN=EXAMPLE.ADA99.FULLSAVE

//DDDEL1       DD    DISP=SHR,DSN=EXAMPLE.ADA99.DELTA1

//DDDEL2       DD    DISP=SHR,DSN=EXAMPLE.ADA99.DELTA2

//SYSUDUMP     DD    SYSOUT=A

//DDDRUCK      DD    SYSOUT=A

//DDPRINT      DD    SYSOUT=A

//DDCARD       DD    *

ADARUN  PROG=ADAULD,SVC=249,DEVICE=3390,DB=99,DSF=YES

//DDKARTE      DD    *

ADAULD    FILE=1,SAVETAPE,PATTERN=FDD
```

z/VM

Dataset	DD Name	Storage	More Information
Unloaded data	DDOUT1	tape/disk	Output by ADAULD (see note)
Unloaded data	DDOUT2	tape/disk	Output by ADAULD (see note)
Unloaded ISNs	DDISN	tape/disk	Required with DDISN
Full save tape	DDFULL	tape/disk	Required for full save tape
Delta save tape(s)	DDDEL1- DDDEL8	tape/disk	Required for delta save tape(s)
Delta Save images	DDDSIMR1	disk	Required for DSIM dataset

Dataset	DD Name	Storage	More Information
Recovery log (RLOG)	DDRLOGR1	disk	Required for ADARAI
ADAULD messages	DDDRUCK	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>
ADARUN messages	DDPRINT	disk/ terminal/ printer	<i>Adabas Messages and Codes</i>
ADARUN parameters	DDCARD	disk/ terminal/ reader	<i>Adabas Operations</i>
ADAULD parameters	SYSIPT	disk/ terminal/ reader	



Note: DDOUT1 and DDOUT2 must have the same block size; otherwise, an ADAULD error will occur. DDOUT2 is required only if NUMOUT=2 is specified.

z/VM Example

```

DATADEF DDOUT1, DSN=ADABASVv.ULD1, UNIT=181, VOL=ULDF1
DATADEF DDFULL, DSN=ADABASVv.ULD1, UNIT=181, VOL=ULDF1
DATADEF DDDEL1, DSN=ADABASVv.ULD1, UNIT=181, VOL=ULDF1
DATADEF DDDEL2, DSN=ADABASVv.ULD1, UNIT=181, VOL=ULDF1
DATADEF DDPRINT, DSN=ADAULD.DDPRINT, MODE=A
DATADEF DUMP, DUMMY
DATADEF DDDRUCK, DSN=ADAULD.DDDRUCK, MODE=A
DATADEF DDCARD, DSN=RUNULD.CONTROL, MODE=A
DATADEF DDKARTE, DSN=ADAULD.CONTROL, MODE=A
ADARUN

```

Contents of RUNULD CONTROL A1:

```
ADARUN PROG=ADAULD, DEVICE=3390, DB=111, DSF=YES
```

Contents of ADAULD CONTROL A1 *

ADAULD FILE=1, SAVETAPE, PATTERN=FDD

VSE/ESA and z/VSE

File	Symbolic Name	Storage	Logical Unit	More Information
Unloaded data	OUT1	tape disk	SYS010 see note 1	Output by ADAULD (see note 2)
Unloaded data	OUT2	tape disk	SYS011 see note 1	Output by ADAULD (see note 2)
Unloaded ISNs	ISN	tape disk	SYS012 see note 1	Required with DDISN
Full save tape	FULL	tape disk	SYS030 see note 1	Required for full save tape
Delta save tape(s)	DEL1- DEL8	tape disk	SYS031- SYS038 see note 1	Required for delta save tape(s)
Delta Save images	DSIMR1	disk	see note 1	Required for DSIM dataset
Recovery log (RLOG)	RLOGR1	disk	see note 1	Required for ADARAI
Messages	SYSLST	printer		<i>Adabas Messages and Codes</i>
ADARUN parameters	SYSRDR CARD	reader/ tape/disk		<i>Adabas Operations</i>
ADAULD parameters	SYSIPT	reader		



Notes:

1. Any programmer logical unit can be used.
2. OUT1 and OUT2 must have the same block size; otherwise, an ADAULD error will occur. OUT2 is required only if NUMOUT=2 is specified.

VSE/ESA and z/VSE Example

See the VSE/ESA-related job information in section [Adabas Utility Functions for Delta Save](#).

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// ASSGN SYS004,DISK,VOL=DISK01,SHR
// DLBL OUT1,'EXAMPLE.ADA99.OUT1'
// EXTENT SYS004,DISK01,,,770,200
// ASSGN SYS030,TAPE
// TLBL FULL,`EXAMPLE.ADA199.FULL'
// ASSGN SYS031,TAPE
// TLBL DEL1,'EXAMPLE.ADA99.DEL1'
// ASSGN SYS032,TAPE
// TLBL DEL2,'EXAMPLE.ADA99.DEL2'
// EXEC PROC=ADAvLIBS
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAULD,SVC=xxx,DEVICE=dddd,DB=yyyy,DSF=YES
/*
ADAULD FILE=1,SAVETAPE,PATTERN=FDD
/*
```


Index

A

- ADAFRM utility
 - DSIM dataset format/reset functions
 - for Delta Save Facility, 7
- ADAPRI utility
 - DSIMPRI function
 - for Delta Save Facility, 19
- ADAREP utility
 - description
 - Delta Save Facility, 25
- ADARES
 - utility
 - for Delta Save Facility, 33
 - utility functions for DSF
 - COPY (copy sequential log file), 34
 - PLCOPY (copy PLOG log files), 38
- ADARUN TT parameter, 84, 89
- ADASAV
 - utility functions for DSF
 - MERGE, 55
 - RESTORE DELTA, 65
 - SAVE, 81
 - SAVE DELTA, 87
- ADAULD utility, 123
 - BS2000/OSD JCL, 134
 - input processing, 128
 - LOAD FILE function
 - DDISN parameter, 126
 - output processing, 130
 - user exit 9 processing, 131
 - VSE JCS, 138
 - z/OS JCL, 135
 - z/VM JCL, 136

B

- BS2000/OSD
 - examples
 - Delta Save (DSF) utility functions", "ADAFRM, 12
 - Delta Save (DSF) utility functions", "ADAPRI, 21
 - Delta Save (DSF) utility functions", "ADAREP, 29
 - Delta Save (DSF) utility functions", "ADARES, 44
 - Delta Save (DSF) utility functions", "ADASAV, 97
 - JCL requirements
 - Delta Save (DSF) utility functions", "ADAFRM, 12
 - Delta Save (DSF) utility functions", "ADAPRI, 21
 - Delta Save (DSF) utility functions", "ADAREP, 28
 - Delta Save (DSF) utility functions", "ADARES, 44

Delta Save (DSF) utility functions", "ADASAV, 96

C

- Checkpoints
 - DSF-related nucleus/utility, 28
- COPY function
 - ADARES utility
 - for Delta Save Facility, 34

D

- Databases
 - partially saving a
 - ADASAV SAVE DELTA function, 87
 - restoring from partial save input
 - ADASAV RESTORE DELTA, 65
 - saving complete
 - ADASAV SAVE function, 81
- Delta Save operation
 - restarting after interrupt, 91
- DSIM dataset
 - rebuilding, 41
- DSIMFRM function
 - formatting the Delta Save DSIM dataset
 - ADAFRM utility, 8
- DSIMPRI function
 - printing the Delta Save DSIM dataset
 - ADAPRI utility function, 19
- DSIMRESET function
 - resetting the Delta Save DSIM dataset
 - ADAFRM utility, 9

E

- Examples
 - ADAPRI printing functions, 21
 - BS2000/OSD
 - Delta Save (DSF) utility functions", "ADAFRM, 12
 - Delta Save (DSF) utility functions", "ADAPRI, 21
 - Delta Save (DSF) utility functions", "ADAREP, 29
 - Delta Save (DSF) utility functions", "ADARES, 44
 - Delta Save (DSF) utility functions", "ADASAV, 97
 - COPY function of ADARES, 38
 - MERGE function of ADASAV, 62
 - PLCOPY function of ADARES, 41
 - RESTORE database function of ADASAV, 78
 - SAVE database function of ADASAV, 84, 90
 - VSE

- Delta Save (DSF) utility functions", "ADAFRM, 17
- Delta Save (DSF) utility functions", "ADAPRI, 24
- Delta Save (DSF) utility functions", "ADAREP, 32
- Delta Save (DSF) utility functions", "ADARES, 52
- Delta Save (DSF) utility functions", "ADASAV, 117

z/OS

- Delta Save (DSF) utility functions", "ADAFRM, 14
- Delta Save (DSF) utility functions", "ADAPRI, 22
- Delta Save (DSF) utility functions", "ADAREP, 30
- Delta Save (DSF) utility functions", "ADARES, 47
- Delta Save (DSF) utility functions", "ADASAV, 105

z/VM

- Delta Save (DSF) utility functions", "ADAFRM, 15
- Delta Save (DSF) utility functions", "ADAPRI, 23
- Delta Save (DSF) utility functions", "ADAREP, 31
- Delta Save (DSF) utility functions", "ADARES, 49
- Delta Save (DSF) utility functions", "ADASAV, 110

F

- Formatting the DSIM dataset
 - for Delta Save Facility
 - ADAFRM utility, 7

- Functions
 - utility
 - Delta Save Facility (DSF), 3

I

- ISNs
 - file of unloaded, 126

J

- JCL requirements
 - BS2000/OSD
 - Delta Save (DSF) utility functions", "ADAFRM, 12
 - Delta Save (DSF) utility functions", "ADAPRI, 21
 - Delta Save (DSF) utility functions", "ADAREP, 28
 - Delta Save (DSF) utility functions", "ADARES, 44
 - Delta Save (DSF) utility functions", "ADASAV, 96

z/OS

- Delta Save (DSF) utility functions", "ADAFRM, 14
- Delta Save (DSF) utility functions", "ADAPRI, 22
- Delta Save (DSF) utility functions", "ADAREP, 29
- Delta Save (DSF) utility functions", "ADARES, 47
- Delta Save (DSF) utility functions", "ADASAV, 104

z/VM

- Delta Save (DSF) utility functions", "ADAFRM, 15
- Delta Save (DSF) utility functions", "ADAPRI, 23
- Delta Save (DSF) utility functions", "ADAREP, 30
- Delta Save (DSF) utility functions", "ADARES, 49
- Delta Save (DSF) utility functions", "ADASAV, 110

- JCS requirements
 - Delta Save (DSF) utility functions
 - ADAFRM, 17
 - ADAPRI, 24
 - ADAREP, 31
 - ADARES, 51
 - ADASAV, 116

M

- MERGE function

- ADASAV utility
 - for Delta Save Facility, 55
- Merging delta save output
 - MERGE function, 55

N

- Nucleus
 - checkpoints, 28
 - single SAVE operation restriction if active, 82

P

- PLCOPY function
 - ADARES utility
 - for Delta Save Facility, 38
- Printing database information
 - for DSIM (Delta Save) datasets
 - ADAPRI utility, 19

R

- rebuilding
 - the DSIM dataset, 41
- Reporting database information
 - for Delta Save
 - ADAREP utility, 25
- Requirements and restrictions
 - DSF partial save operations after ADAFRM changes, 5
 - for NONUC parameter when copying current PLOG, 37
- restart
 - after interrupted delta save operation, 91
- RESTORE function
 - ADASAV utility
 - for Delta Save Facility, 65
- Restoring delta save output
 - RESTORE function, 65

S

- SAVE DELTA partial save function
 - ADASAV utility
 - for Delta Save Facility, 87
- SAVE function
 - ADASAV utility
 - for Delta Save Facility, 81
- SAVE function of ADASAV
 - active nucleus restriction to one operation, 82
- Save tape
 - unload
 - using utility, 123
- Saving the database
 - partial
 - SAVE DELTA function", "with Delta Save Facility, 87
 - SAVE function
 - for Delta Save Facility, 81
- Shadow database
 - maintaining, 68

T

- TTSYN parameter, 82, 84, 88, 89

U

User exits

9

used with ADAULD utility, 131

Utility

checkpoints

Delta Save Facility-originated, 28

Delta Save Facility functions

ADAFRM, 7

ADAPRI, 19

ADAREP, 25

ADARES, 33

ADASAV, 33, 53

general information, 3

V

VSE

examples

Delta Save (DSF) utility functions", "ADAFRM, 17

Delta Save (DSF) utility functions", "ADAPRI, 24

Delta Save (DSF) utility functions", "ADAREP, 32

Delta Save (DSF) utility functions", "ADARES, 52

Delta Save (DSF) utility functions", "ADASAV, 117

JCS requirements

Delta Save (DSF) utility functions", "ADAFRM, 17

Delta Save (DSF) utility functions", "ADAPRI, 24

Delta Save (DSF) utility functions", "ADAREP, 31

Delta Save (DSF) utility functions", "ADARES, 51

Delta Save (DSF) utility functions", "ADASAV, 116

VSE/ESA

job control for DSF, 6

Z

z/OS

examples

Delta Save (DSF) utility functions", "ADAFRM, 14

Delta Save (DSF) utility functions", "ADAPRI, 22

Delta Save (DSF) utility functions", "ADAREP, 30

Delta Save (DSF) utility functions", "ADARES, 47

Delta Save (DSF) utility functions", "ADASAV, 105

JCL requirements

Delta Save (DSF) utility functions", "ADAFRM, 14

Delta Save (DSF) utility functions", "ADAPRI, 22

Delta Save (DSF) utility functions", "ADAREP, 29

Delta Save (DSF) utility functions", "ADARES, 47

Delta Save (DSF) utility functions", "ADASAV, 104

z/VM

examples

Delta Save (DSF) utility functions", "ADAFRM, 15

Delta Save (DSF) utility functions", "ADAPRI, 23

Delta Save (DSF) utility functions", "ADAREP, 31

Delta Save (DSF) utility functions", "ADARES, 49

Delta Save (DSF) utility functions", "ADASAV, 110

JCL requirements

Delta Save (DSF) utility functions", "ADAFRM, 15

Delta Save (DSF) utility functions", "ADAPRI, 23

Delta Save (DSF) utility functions", "ADAREP, 30

Delta Save (DSF) utility functions", "ADARES, 49

Delta Save (DSF) utility functions", "ADASAV, 110

