

Adabas Delta Save Facility Manual

Manual Order Number: ADE741-030IBB

This document applies to Adabas Delta Save Facility Version 7.4 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.

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover or to the following e-mail address:

Documentation@softwareag.com

© December 2002, Software AG

All rights reserved

Printed in the Federal Republic of Germany

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other products and company names mentioned herein may be the trademarks of their respective owners.

TABLE OF CONTENTS

ABOUT THIS MANUAL	1
Related Manuals	2
1. CONCEPTS AND FACILITIES	3
How Delta Save Works	4
Conventional Database Save Operation	5
Save Operation with Delta Save	7
Merging the SAVE Output	9
Protection Log (PLOG) and DSIM Dataset Processing	15
Example	16
Sample Schedule for Save and Merge Operations	20
Restoring Delta Save Save Output	20
Operating a Shadow Database with Delta Save	23
Delta Save Functions	25
ADASAV	25
ADARES	26
ADAREP	26
ADAFRM	26
ADAPRI	26
ADAULD	26
2. INSTALLATION	27
Supported Operating Systems	27
Prerequisites for Installing Delta Save	27
Delta Save Installation Procedure	28
VSE/ESA Considerations for Installing Delta Save	29
Sequential File Table	31
Applying Fixes to Delta Save	32
Applying Fixes to Delta Save Using MSHP PATCH	32
Applying Fixes to Delta Save Using MSHP CORRECT	33

Adabas Delta Save Facility Manual

User ZAPs to Change Dataset Options	34
VSE/ESA ZAP Table for Delta Save	35
BS2000 ZAP Table for Delta Save	35
3. ADMINISTRATION	37
ADARUN Parameter DSF	38
ADARUN Parameter DSFEX1	40
Automatic Delta Save Job Submission	40
Activating the Sample User Exit	40
User Exit Interface	44
Special Online Functions	45
Logging On to the Adabas Delta Save Facility	46
Installing Delta Save Logging	47
Enabling Delta Save Logging	48
Displaying Current Allocation Information	49
Displaying Files Modified by Utilities	50
Estimating the Size of a Delta Save Operation	51
Disabling Delta Save Logging	52
Changing the DLOG Area	53
Removing Delta Save Logging	54
Calculating the DLOG Area Size	55
Calculating the DSIM Dataset Size	57
4. ADABAS UTILITY FUNCTIONS FOR DELTA SAVE	59
Utility Considerations for Delta Save Operation	60
Identifying New AC Extents Allocated During an Online Save	60
VSE/ESA Job Control for Delta Save	61
5. ADAFRM UTILITY	63
DSIMFRM : Format DSIM Dataset	63
Essential Parameter	63
Optional Parameters	64
Example	64
DSIMRESET : Reset the DSIM Dataset	65

Table of Contents

Essential Parameters	65
Optional Parameters	66
Example	66
JCL/JCS Requirements and Examples	67
BS2000	67
OS/390 or z/OS	69
VM/ESA or z/VM	70
VSE/ESA	71
6. ADAPRI UTILITY	73
ADAPRI DSIMPRI : Printing the DSIM Dataset	73
Essential Parameters	73
Optional Parameters	73
Example	74
JCL/JCS Requirements and Examples	75
BS2000	75
OS/390 or z/OS	76
VM/ESA or z/VM	77
VSE/ESA	78
7. ADAREP UTILITY	79
Delta Save Report Format	80
Delta Save Checkpoints	81
JCL/JCS Requirements and Examples	82
BS2000	82
OS/390 or z/OS	83
VM/ESA or z/VM	84
VSE/ESA	85
8. ADARES UTILITY	87
COPY : Copy Sequential Protection Log	87
Syntax	88
Essential Parameters	89
Optional Parameters	89

Adabas Delta Save Facility Manual

Example	91
PLCOPY : Copy PLOG to Sequential Log	92
Syntax	92
Essential Parameter	93
Optional Parameters	93
Example	94
Rebuilding the DSIM Dataset	95
JCL/JCS Requirements and Examples	96
BS2000	96
OS/390 or z/OS	98
VM/ESA or z/VM	99
VSE/ESA	101
9. ADASAV UTILITY	103
MERGE	103
Consolidating Delta Saves or Updating Full Saves	103
Merging Online Saves and DSIM Datasets	104
Unloading DSIM Datasets	105
Using Unloaded DSIM Datasets as “Online” MERGE Input	106
Concatenated Datasets in MERGE Input	107
Syntax	107
Optional Parameters	108
Examples	110
RESTORE DELTA : Restore Without Prior Merging	111
Online Save and DSIM Inputs	111
Concatenated Datasets in RESTORE DELTA Input	113
Output Database Requirements	113
Subsequent Operations	114
Syntax	114
Optional Parameters	116
Examples	125
SAVE : Save Database	127
Syntax	128
Optional Parameters	128
Example	130
SAVE DELTA : Save Changed Database Blocks	131

Table of Contents

Syntax	132
Optional Parameters	132
Example	133
Restarting an Interrupted Save Operation	134
User ABEND 34 or 35	134
System ABEND or Other User ABEND	134
Resetting the DSIM Dataset	135
JCL/JCS Requirements and Examples	136
BS2000	136
OS/390 or z/OS	141
VM/ESA or z/VM	145
VSE/ESA	149
10. ADAULD UTILITY	153
UNLOAD FILE : Unload Specified File from a Save Tape	154
Syntax	154
Essential Parameters	154
Optional Parameters	155
Examples	157
Save Tape Input Processing	158
DSIM Dataset or the Protection Log	159
ADAULD Output Processing	160
ADAULD User Exit 9	161
JCL/JCS Requirements and Examples	162
BS2000	162
OS/390 or z/OS	163
VM/ESA or z/VM	164
VSE/ESA	165
APPENDIX A : GLOSSARY OF TERMS	167
INDEX	173

ABOUT THIS MANUAL

Note:

Dataset names starting with DD are referred to in Adabas Delta Save Facility manuals with a slash separating the DD from the remainder of the dataset name to accommodate VSE/ESA dataset names that do not contain the DD prefix. The slash is not part of the dataset name.

This manual describes the installation and operation of the Delta Save Facility, and is intended for analysts, programmers and database administrators (DBAs) who apply, install, and manage Adabas systems.

The manual is organized as follows:

Chap	Content
-------------	----------------

1	introduces the Delta Save Facility and provides a detailed description of the various Delta Save operating modes and functions.
2	describes prerequisites and procedures for installing the Delta Save Facility.
3	tells you how to activate Delta Save, and control it using the related menus and screens of either the demo version or selectable unit Adabas Online System (AOS).
4–10	describes the special Delta Save utility functions: <ul style="list-style-type: none">4 – overview and general information5 – ADAFRM6 – ADAPRI7 – ADAREP8 – ADARES9 – ADASAV10 – ADAULD
A	is a glossary of Adabas and Delta Save terms.

Messages and codes information for Delta Save is included in the *Adabas Messages and Codes* manual.

Related Manuals

The following Adabas mainframe manuals are referred to in this manual, or may be needed when installing and running Adabas with Delta Save:

- Adabas Release Notes
- Adabas Installation Manual
- Adabas Operations Manual
- Adabas DBA Reference Manual
- Adabas Messages and Codes
- Adabas Utilities Manual, Volumes 1 and 2

CONCEPTS AND FACILITIES

Software AG's Adabas Delta Save Facility (DSF) offers significant enhancements to ADASAV utility processing by backing up and restoring only the changed (delta) portions of Adabas databases.

The Delta Save Facility

- includes logic in the Adabas nucleus for recording the RABNs of all changed database blocks. Adabas utilities that change the database contain logic to record which portions of the database have been affected.
- creates and merges interim “delta” save tapes while the database remains online, if required.
- consolidates delta save tapes with the most recent database save tape to create an up-to-date full save tape.
- restores the database from the most recent full save tape and all subsequent delta save tapes.

While maintaining the availability of Adabas SAVE and RESTORE functions, the high performance of Adabas, and the ability of Adabas to survive system failures, DSF increases database availability by

- reducing the volume of save output produced; and
- shortening the duration of save operations.

Because DSF allows more frequent save operations, database recovery time is reduced.

DSF is intended for Adabas sites with one or more large, heavily updated databases that need to be available most of the time. Especially for sites where the volume of data changed on a day-to-day basis is considerably smaller than the total database volume, DSF provides for

- more frequent saves without interrupting database availability;
- enhanced “24 x 7 x 52” operation;
- full offline saving in parallel with the active database; and
- shorter REGENERATE duration during recovery.

This chapter tells you how DSF operates and how it manages Adabas database save operations more effectively.

How Delta Save Works

During normal Adabas database operation, the Delta Save Facility keeps track of changed Associator and Data Storage blocks. The RABNs of blocks that are altered are recorded in a portion of the Associator called the Delta Save logging (DLOG) area, allocated for the purpose. During a delta save operation, images of these changed blocks are copied to create a delta save tape.

Delta save and normal Adabas nucleus operations can occur in parallel. Any changes made by the nucleus during the delta save operation are later retrieved from the Adabas protection log (PLOG). In most cases, the resulting delta save output is written to tape for later use.

Merging the most recent ADASAV SAVE result (SAVE database) with all following delta save output results in a new full database save that reflects the database status following the last delta save operation. Alternatively, the latest SAVE database and all subsequent delta save results can be supplied as input to a database restore (ADASAV RESTORE DELTA) operation.

The specific stages in this process are as follows:

1. Enable Delta Save operation by performing a normal full save (ADASAV SAVE database) that contains the entire database.
2. At some point following step 1, perform a delta save (ADASAV SAVE DELTA) operation, creating output that contains only the blocks that were changed since the last full or delta save operation was performed; this can be done multiple times.
3. Optionally, merge several previously created delta save outputs to produce a “consolidated” delta save tape comprising all changes since the last full save tape creation.
4. Merge the previously created full and delta save output to produce an up-to-date full save tape. Note that the old full or delta save inputs can also be the result of previous merge operations.
5. If the need arises, restore the database by supplying either a fully merged full save tape, or the most recent full save tape and all subsequent delta save tapes.

Conventional Database Save Operation

Without the Delta Save Facility, a conventional database save operation can be performed with the Adabas nucleus inactive (offline) or active (online).

Offline Save

An offline save is performed with the Adabas nucleus inactive. No concurrent access or update of the database is allowed so that the resultant save tape contains all database Associator and Data Storage blocks consistent with the time of the save operation.

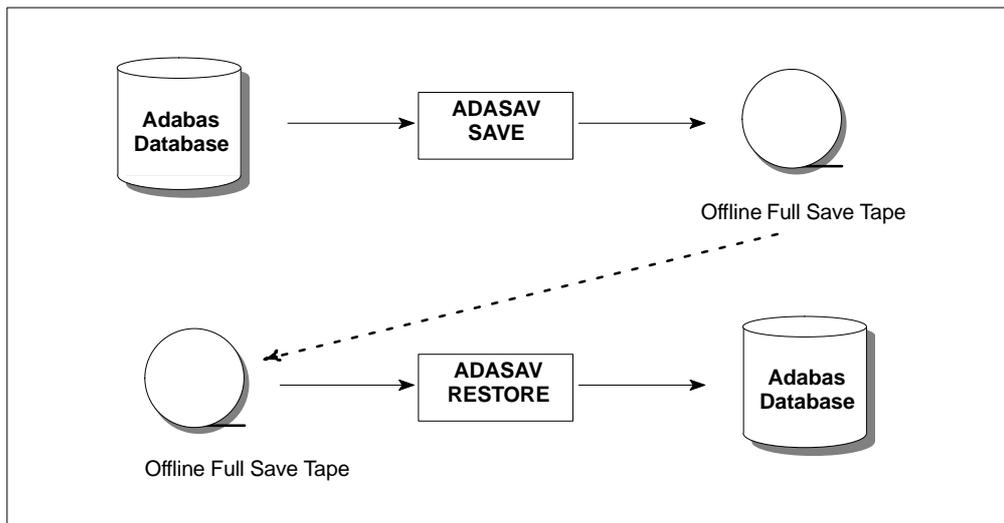


Figure 1-1: Offline Adabas Save/Restore Sequence

Figure 1-1 shows the general sequence of a conventional Adabas database offline save operation without Delta Save. First, the ADASAV utility's SAVE function is used to create a full database save (SAVE database), usually on tape. If the need arises, this full database save can be used for restoring either the entire database or specific files to their state at the time of the save operation (using the ADASAV RESTORE function).

Online Save

An online save is performed while the Adabas nucleus is active and the database is available for normal use. During the online save, users may access and update records in the database. Concurrent utility updates are not allowed.

At the end of the save, ET synchronization ensures that all partial transactions are completed before new transactions are started so that the database is in a consistent state. Since the database can be modified during the save operation, the save tape alone may not be consistent with the database. Therefore, the images of any blocks in the database that were changed while the save operation is in progress are recorded in the protection log (PLOG), as shown in Figure 1-2.

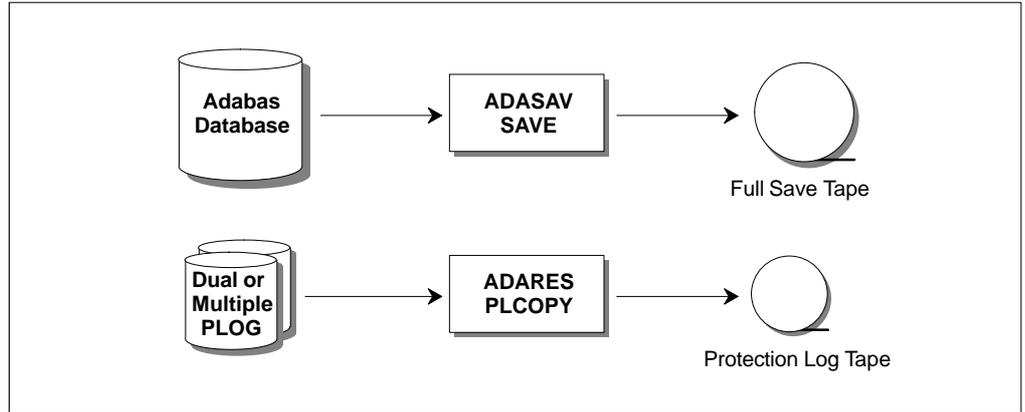


Figure 1-2: Online Adabas Save Sequence

If required, the entire database or specific files can be restored to the state at the end of the online save operation by using the full database save tape together with the associated protection log as input to the ADASV RESTONL function, as shown in Figure 1-3.

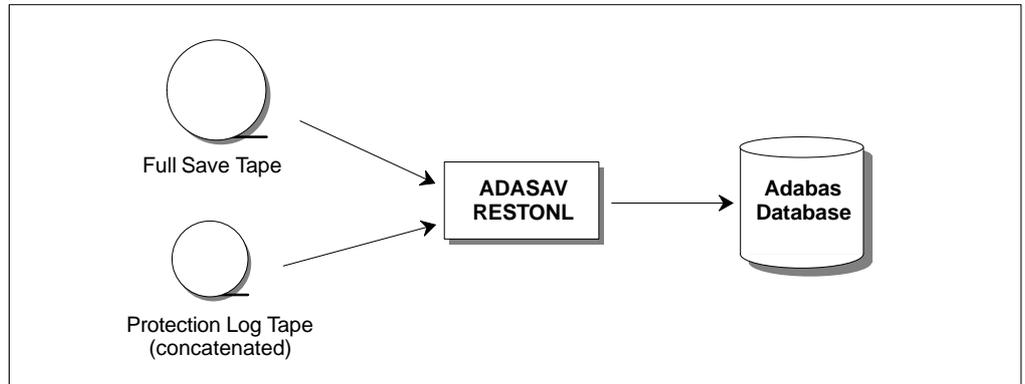


Figure 1-3: Online Adabas Restore Sequence

Save Operation with Delta Save

The Delta Save Facility provides full save and delta save operations. The full save saves the whole database, similar to a conventional database save, and is invoked by the ADASAV SAVE function. The delta save operation saves only blocks changed since the last full or delta save operation, and is invoked by the ADASAV SAVE DELTA function. For full save and delta save operations, the database administration blocks, FCBs, FDTs, and DSST are always saved.

Although the procedure for creating a delta save tape is much the same as for a full database save tape, the actual time required is much less because only the changed areas of the database are saved. Secondly, by performing less frequent full saves but more frequent delta save operations, the overall time needed to run the save operations is normally considerably less than performing the same number of conventional database save operations.

Further processing of full save and delta save tapes is provided by the ADASAV MERGE function. The merge function can be performed offline and therefore does not affect database processing.

When an ADASAV SAVE DELTA operation is performed, the logged Delta Save information is read and compiled into a save tape containing only those changes made since the last full or delta save operation, as shown in Figure 1-4 (offline delta save) and Figure 1-5 (online delta save).

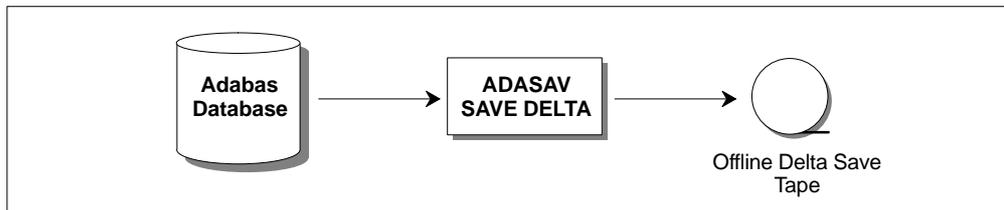


Figure 1-4: Offline Delta Save

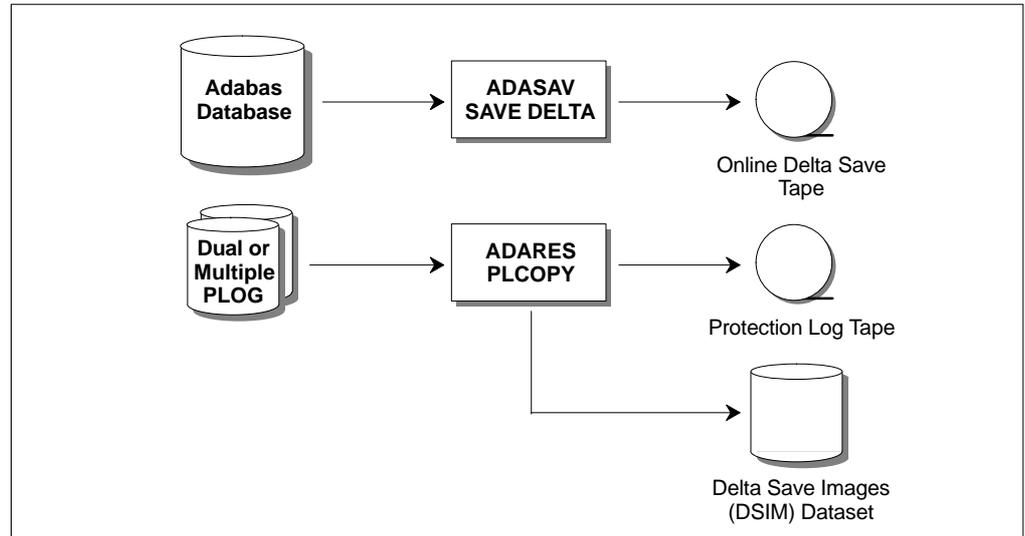


Figure 1-5: Online Delta Save

Figure 1-5 shows that the online delta save—like a full save—results in protection log (PLOG) output for any activities performed while the delta save was in process.

In addition, however, the PLOG copy (ADARES PLCOPY) operation after the delta save produces a delta save images (DSIM) dataset. This new dataset holds copies of all Associator and Data Storage blocks changed while the online delta save operation was in progress, and provides them for a subsequent merge operation.

The save tapes created by the ADASAV SAVE FILE or ADASAV SAVE INCREMENTAL functions cannot be used as delta save tapes and therefore cannot be used in merging operations.

Tracking Database Changes

When the Delta Save Facility is used, the nucleus logs the RABNs of changed blocks in the Delta Save logging area (DLOG). The DLOG area is allocated within the Associator by the Adabas basic online services facility. The delta save operation retrieves RABNs of blocks to be saved from the DLOG area. When a full save or delta save operation completes successfully, the DLOG area is reset.

The DLOG area is continuously updated as more blocks are changed by normal nucleus operation. In addition, Associator and Data Storage changes made by utilities are recorded, although not in the DLOG area but in the FCBs of the affected files.

When a utility modifies a file, information is recorded in the related FCBs that the file's Data Storage, address converter, and/or index was changed. The changed RABNs themselves are not logged. Thus, for example, when the ADAORD REORFDATA function reorders the Data Storage of a file, it sets marks in the FCB of the file to indicate that the entire Data Storage and address converter of the file are to be saved in the next delta save operation.

If the utility function is ADALOD UPDATE, the RABN ranges of changed Data Storage and address converter blocks are stored in a file-specific extension of the DLOG area allocated by ADALOD. Only the blocks in the extension are marked for saving during the next delta save operation. (ADALOD UPDATE always changes, and marks as changed, the entire index of the updated file.) If the area reserved for logging RABNs changed by ADALOD UPDATE overflows, the logging information is compressed and more blocks than were actually modified will be saved in the next delta save operation.

Merging the SAVE Output

The ADASAV MERGE function (see page 103) combines the delta save results and, where applicable, the original or unloaded DSIM dataset from one or more ADASAV SAVE DELTA operations to create a new delta or full save tape. This is illustrated in Figure 1-6.

When you “unload” the DSIM dataset, you transfer its contents to a sequential delta save dataset without specifying and processing the online full or delta save dataset associated with the DSIM dataset. The unloaded DSIM dataset can be used for MERGE or RESTORE DELTA operations.

If only delta saves (that is, no full saves) are merged, the output is a single, consolidated delta save tape. Such a save tape comprises all database changes that occurred in the time period covered by the input delta saves.

If the input contains a full save tape, the merge operation results in a single, merged full save output. A merged full save tape contains the whole database as it was at the time when the last input delta save was written.

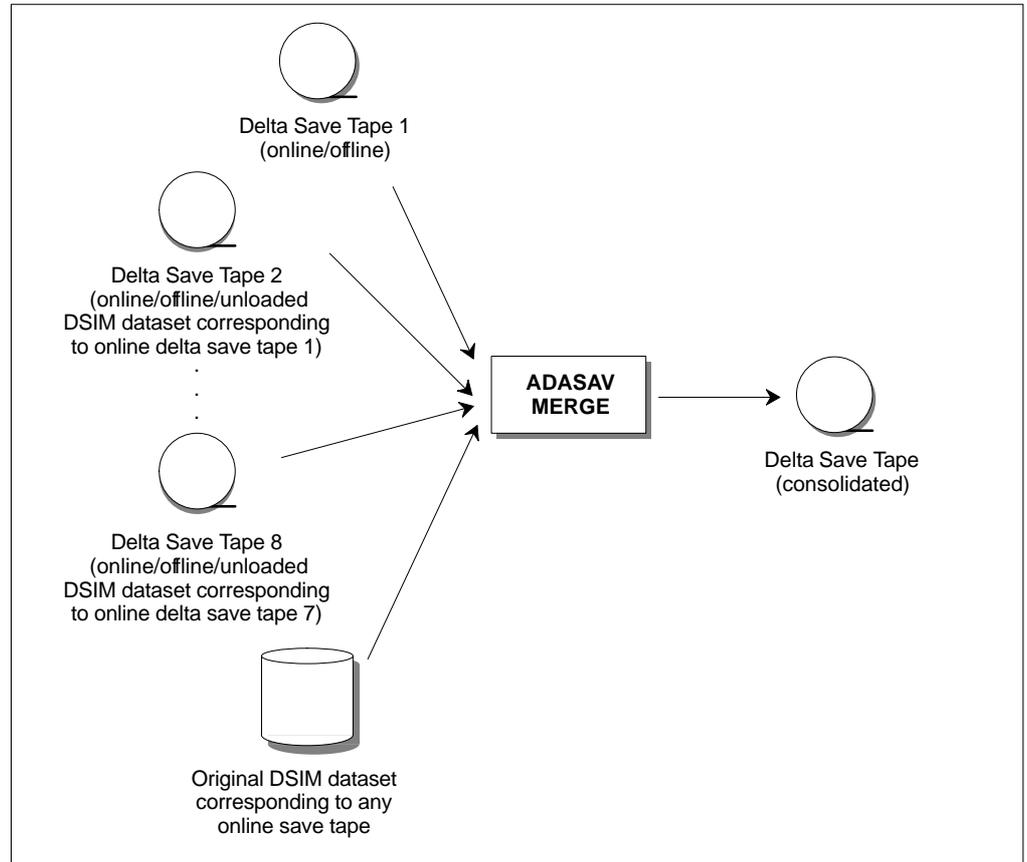


Figure 1-6: Merge Operation—Making a Consolidated Delta Save Tape

The original DSIM dataset corresponds to any online save input. Any online delta save input to which the original DSIM, if supplied, does not correspond must be followed (with the next higher sequence number) by the corresponding unloaded DSIM.

If the full save input is an online save tape, it must be accompanied either by its original DSIM dataset or followed by the delta save dataset created when the original DSIM dataset was unloaded. This is illustrated in Figure 1-7.

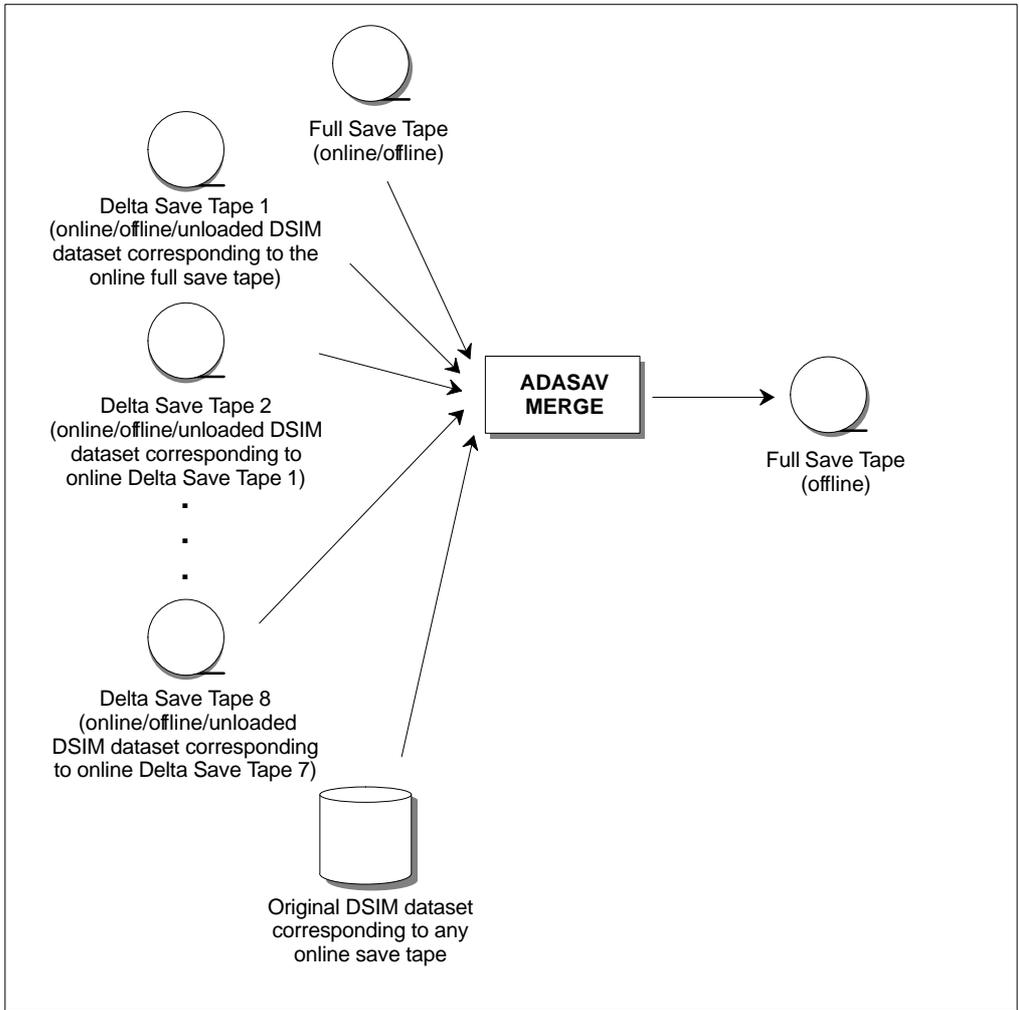


Figure 1-7: Merge Operation—Making a New Full Save Tape

The unload frees the DSIM dataset for the next online save operation. Unlike a MERGE of the DSIM dataset, its “unload” eliminates the need to process the associated online save dataset—which may be very large in the case of a full save—again just after it has been created.

The original rather than the unloaded DSIM dataset related to any online save operation can be supplied for the merge operation. Following the merge operation, the DSIM dataset is reset and can then be used for the next online save operation.

The possible inputs for merging are

- zero or one full save tapes;
- zero to eight consecutive delta save tapes including unloaded DSIM datasets for each online save tape;
- the DSIM dataset for any online save unless the unloaded DSIM dataset is provided.

It is a requirement for all input save tapes to be from a continuous sequence of save operations on the same database. No save tapes can be omitted; otherwise, the result would be inconsistent. Therefore, a non-continuous sequence of input tapes is rejected by the ADASAV MERGE function.

Each input save tape can be the result of earlier merge operations.

The output tape may be used in subsequent merge operations.

Any online save input to the merge operation must be accompanied by either the original or the unloaded DSIM dataset related to that online save operation. Only one original DSIM dataset can be specified.

The output of a merge operation is always the same as an offline save tape; it can be specified with a new online delta save tape in a subsequent merge operation.

If more than eight consecutive delta save tapes have been created, you must merge the first eight, and then the result of that merge with the next delta save tape(s), and so on, in chronological order.

Delta Save Identifier (DSID)

If the Delta Save Facility is used, every save tape (full or delta) is assigned a Delta Save identifier (DSID), which is used for consistency checking in merge operations, for example, to prevent invalid merge operations. The DSID comprises the full save number, the delta save number, and a time stamp.

The full save number counts the number of full save operations since the Delta Save Facility was installed. The delta save number counts the number of delta save operations since the last full save. The time stamp records the date and time of the last full or delta save.

Each full save operation increments the full save number by one, resets the delta save number to zero, and assigns a new time stamp. Each delta save operation leaves the current full save number unchanged, increments the delta save number by one and assigns new time stamp.

When save tapes (full or delta) are subsequently input for merging, they must have the same full save number, consecutive delta save numbers, and ascending time stamps.

During a merge operation, the DSID of the merged save tape (full or delta) is derived from the input DSIDs as follows:

- full save number;
- delta save number range (that is, first delta save number – last delta save number);
- last time stamp.

Example:

Full save input DSID:	4/0/t1	(Full save number 4, delta save number 0, date/ time t1)
Delta save input 1 DSID:	4/1/t2	
Delta save input 2 DSID:	4/2/t3	
Merged full save output DSID:	4/0–2/t3	

Invalid Merges

Before a merge operation starts, DSF checks that the DSIDs of the full and delta save tapes are a valid combination. In almost all cases, this check reliably prevents any invalid merge operation.

There are, however, some unusual cases in which the check is insufficient, namely when DSIDs are “reused” after

- removal and reinstallation of the DLOG area, or
- restore of the database to an old status.

Example:

Full save	DSID 1/0/t1	(t1 is a time stamp)
... updates ...		
Delta save	DSID 1/1/t2	
Restore full save, restores	DSID 1/0/t1	
... updates ...		
Delta save	DSID 1/1/t3	
... updates ...		
Delta save	DSID 1/2/t4	

It is now possible to merge full save 1/0/t1, delta save 1/1/t2, and delta save 1/2/t4, even though this is an invalid combination.

Such a mistake can be avoided by observing the following rules:

Important Rules

1. When the database is restored to an old status, all delta save tapes produced after the time represented by that status must no longer be used and should be discarded.
2. When the DLOG area is removed and reinstalled, all existing delta save tapes should be discarded.

In the example above, delta save tape 1/1/t2 should have been discarded after the restore.

Protection Log (PLOG) and DSIM Dataset Processing

When the Delta Save Facility (DSF) performs an “online” save operation, special processing is required for the PLOG and the DSIM dataset that is not necessary for an offline save operation.

The nucleus should run with a dual or multiple PLOG when performing an online save with DSF. PLOG and DSIM dataset processing proceeds as follows when dual or multiple protection logging is in effect:

1. The DSIM dataset must be free (not in use) initially.
2. The online save acquires the DSIM dataset for use.
3. The online save operation starts (full or delta).
4. The nucleus writes images of changed blocks to the dual or multiple PLOG.

During a single online save operation, several ADARES PLCOPY operations may be performed to copy the dual or multiple PLOG information to the sequential PLOG and to extract images of the changed Associator and Data Storage blocks and write them to the DSIM dataset.

ADARES can be invoked manually or by user exit 2 (dual logging) or user exit 12 (multiple logging) when it is programmed as recommended (see the *DBA Reference Manual* for information about the user exits). The user exit invokes ADARES when a PLOG is switched, which happens either when one PLOG dataset runs full or when the FEOFPL function is issued from ADADBS OPERCOM, AOS, or the operator console.

Creating an online save tape and building a corresponding DSIM dataset are related but separate operations. The DSIM dataset can be built step by step during the several PLCOPY operations that may be performed. However, when the online save operation ends, a PLCOPY operation for completing the DSIM dataset is still outstanding.

5. The online save operation terminates. At this point there is information on the PLOG that must be copied to the DSIM dataset.
6. ADARES PLCOPY is run again to complete the DSIM dataset.
- 7a. A subsequent merge operation combines the online save tape and the DSIM dataset.

For the merge operation:

- zero or one (1) full save datasets and zero to eight (8) delta save datasets can be merged.
- both offline and online save datasets can be supplied to a single merge operation.

- online full or delta save datasets must be supplied in sequence with the appropriate DSIM dataset. Unloaded DSIM datasets count as delta save datasets. Any online save dataset must be accompanied by its associated original or unloaded DSIM dataset.

The result of merging an online save dataset and its DSIM dataset is an “offline” save tape.

The merge function waits for completion of the DSIM dataset if the DSIMWAIT parameter is specified.

- 7b. Alternatively, you can “unload” the DSIM dataset without immediately merging it with the online save, deferring this merge until a later time (see the DSIMWAIT parameter).
8. On completion of the merge or unload operation, the DSIM dataset is freed (no longer in use).

Example

Figure 1-8 (four steps) shows several Delta Save save and merge operations.

Step 1 : Online Full Save

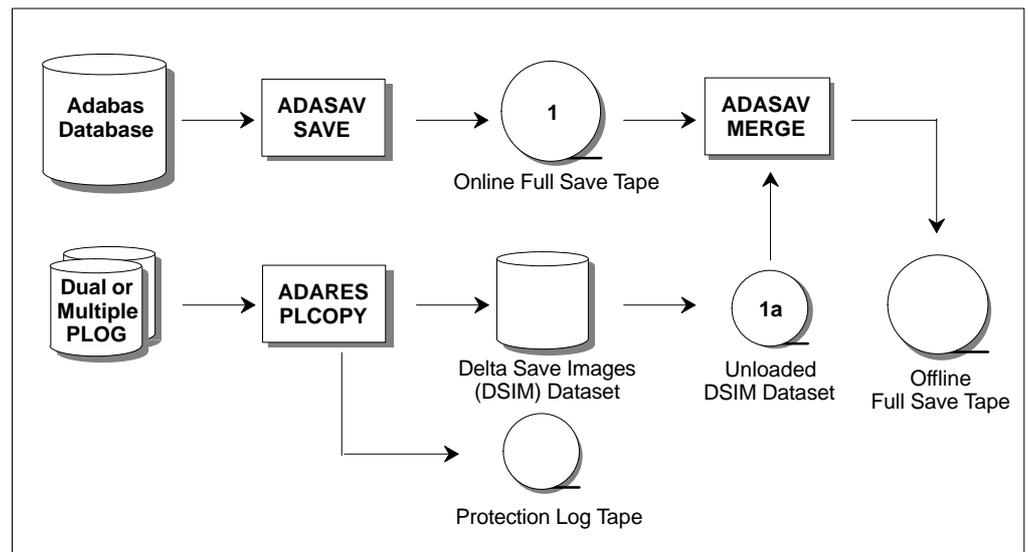


Figure 1-8: Delta Save Save / Merge Sequence, Step 1

- A full save operation is performed online;
- ADASAV SAVE creates an online full save tape;
- Assuming the save is the first after installation of the DLOG area, the DSID of the save tape is 1/0/t1 (that is, full save number 1, delta save number 0, and date/time of the save, t1);
- During the online save, the nucleus writes images of changed database blocks to the dual or multiple PLOG;
- ADARES PLCOPY copies the dual or multiple PLOG to a sequential PLOG; it also extracts block images from the PLOG and writes them to the DSIM dataset;
- When the save operation is complete and the DSIM dataset has been built, the DSIM dataset is unloaded to a sequential file for later use. Its DSID is 1/0/t1;
- It is possible to merge the online full save with the DSIM dataset to produce an offline full save. In this case, the DSID of the merged save tape would still be 1/0/t1.
- The DSIM dataset is now free for the next online save.

Step 2 : Online Delta Save

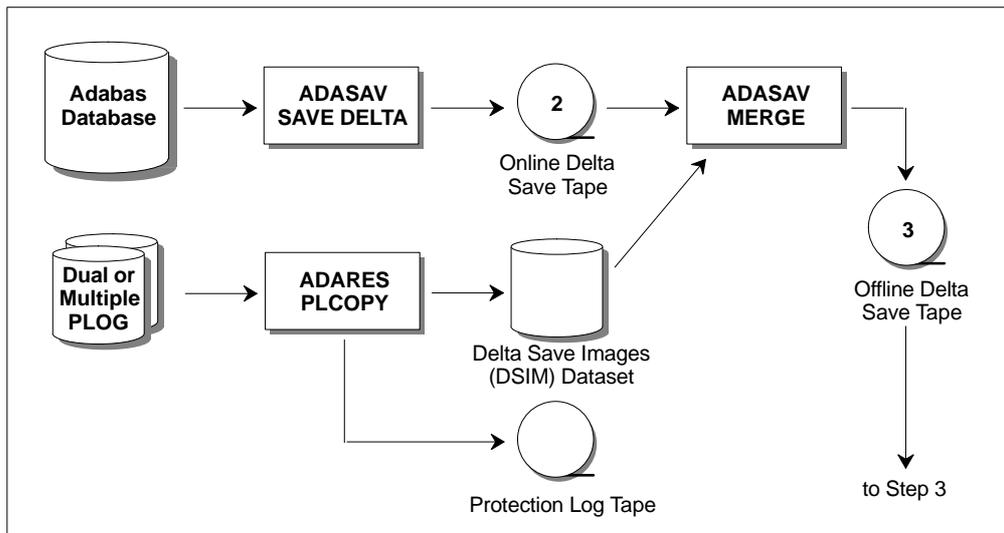


Figure 1-8: Delta Save Save / Merge Sequence, Step 2

- A delta save operation is performed online;
- ADASAV SAVE DELTA creates an online delta save tape;
- The DSID of the delta save tape is 1/1/t2 (where t2 denotes the date/time of the save operation);
- In the same way as in step 1, the images of database blocks changed during the online save operation are copied to the DSIM dataset;
- When the save operation has completed and the DSIM dataset has been built, either the DSIM dataset can be unloaded to a sequential file for later use or ADASAV MERGE can be used to combine the online delta save and the block images on the DSIM dataset to an offline delta save;
- If ADASAV MERGE is used, the DSID of the merged save tape is still 1/1/t2.

Step 3 : Online Delta Save, Merge Consolidated Delta Save

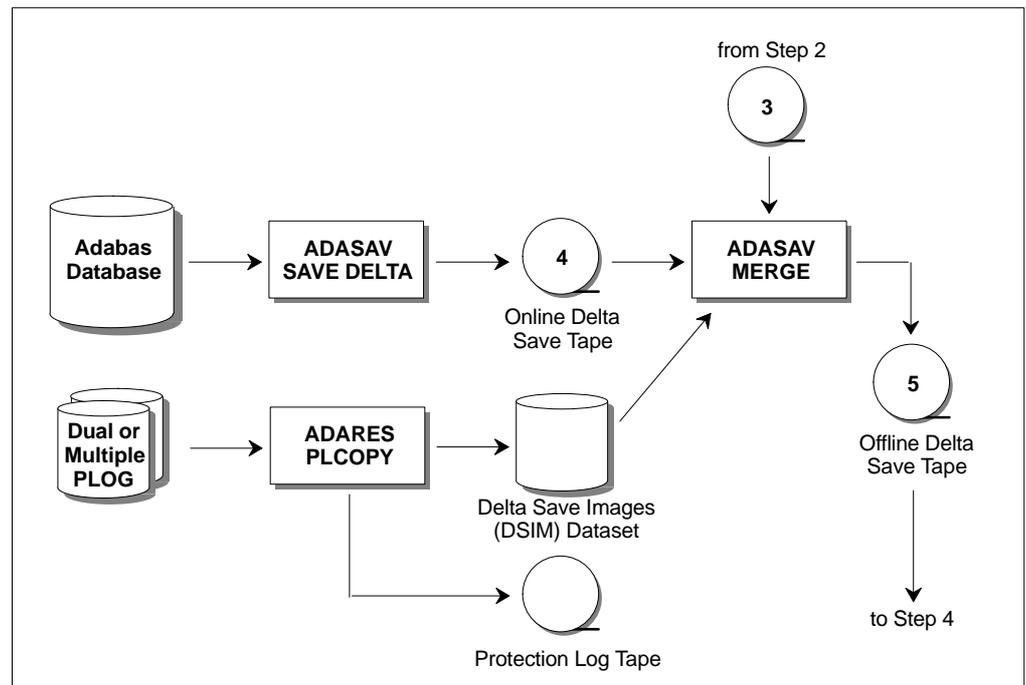


Figure 1-8: Delta Save Save / Merge Sequence, Step 3

- A delta save operation is performed online;
- ADASAV SAVE DELTA creates an online delta save tape,
- The DSID of the delta save tape is 1/2/t3;
- ADASAV MERGE combines the merged delta save tape from step 2, the online delta save tape just created, and its corresponding DSIM dataset to a new, consolidated offline delta save tape;
- The DSID of the merged delta save tape is 1/1-2/t3;
- The merged delta save contains all blocks changed since the full save in step 1.

Step 4 : Online Delta Save, Merge Full Save

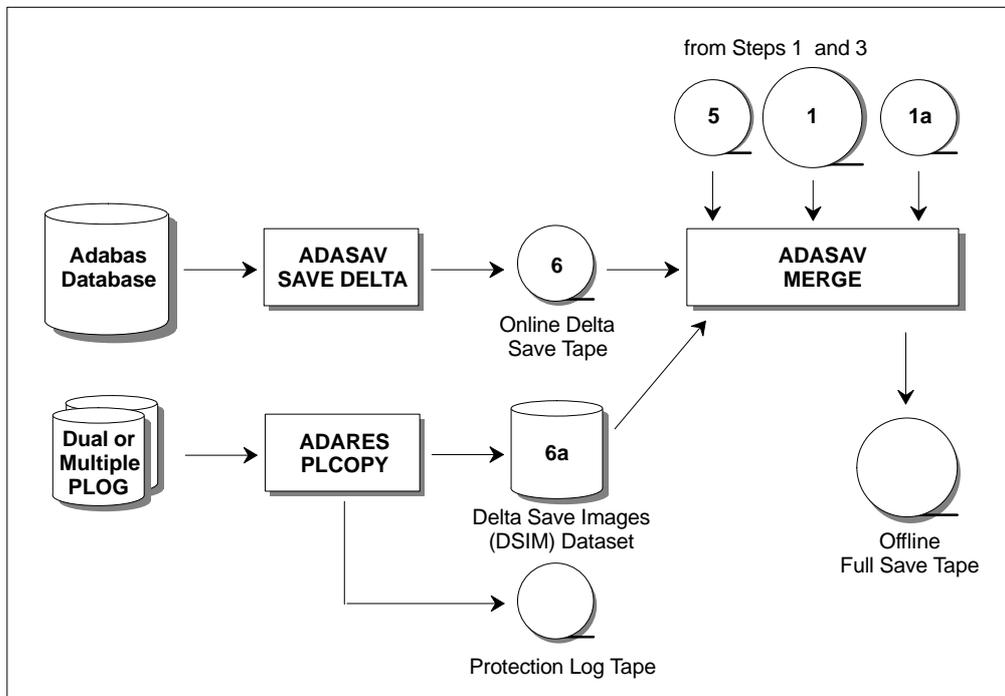


Figure 1-8: Delta Save Save / Merge Sequence, Step 4

- A delta save operation is performed online;
- ADASAV SAVE DELTA creates an online delta save tape;
- The DSID of the delta save tape is 1/3/t4;
- ADASAV MERGE combines the online full save tape (1) and its unloaded DSIM dataset (1a) from step 1, the consolidated delta save tape (5) from step 3, the online delta save tape just created (6), and its corresponding original DSIM dataset (6a) to a new, merged full save tape;
- The DSID of the merged full save tape is 1/0–3/t4;
- The merged full save tape contains the database as it was at the end of the last online save;
- This full save tape may be supplied as input for a ADASAV RESTORE database function;
- The merged full save tape can also be supplied as input for ADASAV RESTORE FILE or RESTORE FMOVE functions.

Sample Schedule for Save and Merge Operations

1. Start a cycle with a full save.
2. Perform a delta save operation every day.
3. Maintain a consolidated delta save tape that contains all blocks changed since the start of the cycle.
4. To achieve this, merge the consolidated delta save tape with each new delta save tape to get a new consolidated delta save tape.
5. Once a week, merge the full save tape and the consolidated delta save tape to get a new full save tape.
6. This completes the cycle and starts a new one.

Restoring Delta Save Save Output

Merged full save tapes can be used for any ADASAV RESTORE function in the same way as conventional database save tapes. If delta saves have been taken since the production of the last full save tape, the full save and the delta saves can be merged to produce a new full save suitable for RESTORE.

To save time when recreating the database or files, full and delta saves can be merged and restored directly by using the ADASAV RESTORE DELTA function. The input for RESTORE DELTA is

- one full save tape;
- zero to eight (8) consecutive delta save tapes including unloaded DSIM datasets for each online save tape;
- the DSIM dataset for the last online save unless the unloaded DSIM dataset is provided.

—as shown in Figure 1-9.

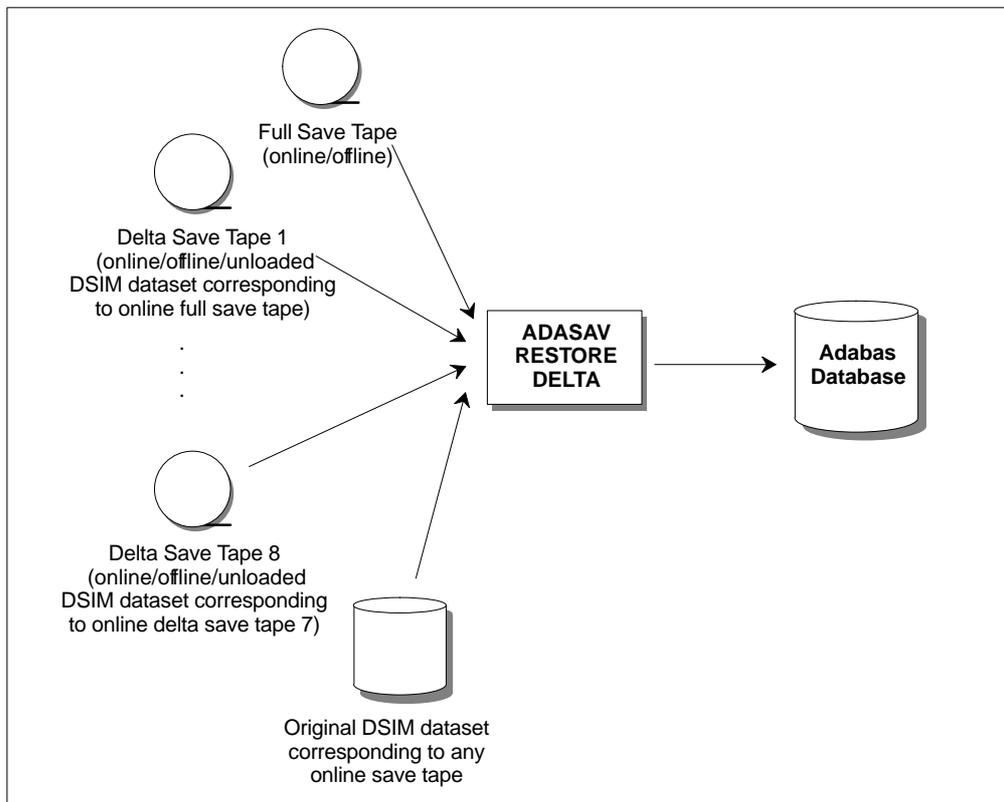


Figure 1-9: Delta Restore Option

Consecutive saves must be specified for RESTORE DELTA: no save tape may be missing. Each save tape may itself be the output of an earlier merge operation. For tapes that originate from online save operations, the corresponding original or unloaded DSIM dataset must be specified. An original DSIM dataset is **not** reset at the end of the RESTORE DELTA operation but remains available for further merge operations.

If no delta save tape is supplied, the full save tape must be online and the corresponding DSIM dataset, original or unloaded, must be specified. If only an offline full save tape is supplied, the conventional RESTORE function must be used.

If save and merge operations are performed as in the schedule described in the section **Sample Schedule for Save and Merge Operations** on page 20, at any time only the last (merged) full save tape, the consolidated delta save tape and, if present, the most recent yet unmerged delta save tape are required to restore the entire database to the status of the last delta save operation. This can be done with RESTORE DELTA.

Although the delta restore operation can restore the entire database, it is also possible to specify the GCB, FILE, or FMOVE parameters to restrict the delta restore operation, similar to a conventional restore.

On the input side, RESTORE DELTA is like MERGE since the input consists of a number of full save and delta save tapes. On the output side, RESTORE DELTA is like RESTORE since data is restored to the physical database.

If a delta restore operation is interrupted, it can be restarted, provided the full save has already been restored. This status is indicated by a message in the ADASAV protocol. In this case, the interrupted delta restore operation may be restarted by resubmitting the job, but without the job control specification for the full save tape. Then, ADASAV will continue the operation and restore the delta save tapes. If a restart is not possible, the whole delta restore operation must be repeated.

Operating a Shadow Database with Delta Save

The Delta Save Facility can be used to set up and maintain a “shadow” database; that is, a copy of the original database. Such a shadow database can

- accelerate reconstruction of the original database in case it is physically destroyed; and
- run read-only analysis programs to reduce the load on the original database.

The Delta Save Facility maintains a shadow database by

- restoring a full save of the original database into the shadow database’s container files (see Figure 1-10, Step 1);
- performing delta save operations on the original database at regular intervals; and
- performing a delta restore into the shadow database with each delta save tape, without supplying a full save tape (see Figure 1-10, Step 2).

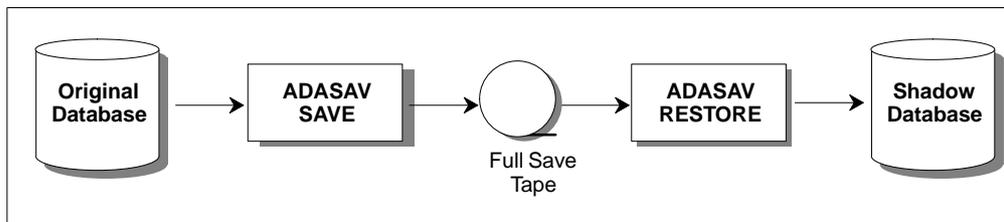


Figure 1-10: Maintaining a Shadow Database, Step 1

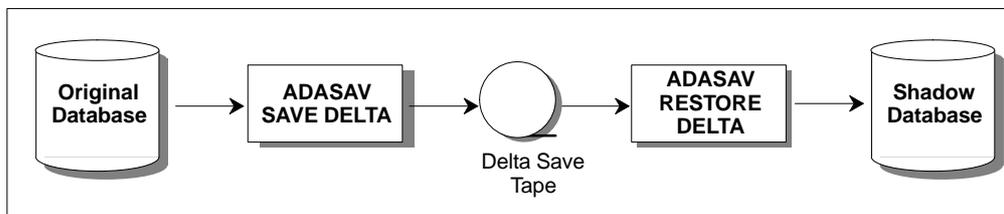


Figure 1-10: Maintaining a Shadow Database, Step 2

Such a shadow database has the following characteristics:

- The shadow database is always a slightly outdated copy of the original database. The size of the discrepancy depends on the time elapsed since the last delta save/delta restore operation.
- At any time, only an ADARES REGENERATE operation is needed to bring the shadow database to the latest status of the original database.
- This regenerate operation processes the PLOG starting from the point of the most recent delta save.
- Since an ET-synchronization is performed at the end of every online save (and when the Adabas nucleus is shut down), all full or delta save tapes, and hence the shadow database, always represent a synchronized status of the original database: all transactions are completed, there are no partial, incomplete transactions.
- Between each two delta restore operations into the shadow database, Adabas nuclei may be active on the shadow database in read-only mode to allow analysis programs to execute.
- During a delta restore operation, no read-only nucleus can be active.
- At no time may an update nucleus or an update utility (ADAINV, ADALOD, ADAORD, ADASAV) be active on the shadow database.
- Once an update nucleus or an update utility is run on the shadow database, it is no longer possible to restore a delta save tape into the shadow database without supplying a full save input.

The shadow database is kept up to date with the RESTORE DELTA function. The input requirements for RESTORE DELTA are as described in the section **Restoring Delta Save Save Output** on page 20, except that the full save tape is omitted.

When the shadow database is updated by the sequence of delta restore operations, consecutive delta save tapes must be supplied; that is, no delta save may be omitted. This is checked using the DSIDs of the save tapes.

The delta save operations on the original database may be performed offline or online. When using online delta saves to update then shadow database, the DSIM dataset processing applies as for normal RESTORE DELTA operations. However, an original DSIM dataset is not freed at the end of the RESTORE DELTA operation, so that it can also be used for a merge operation with the online save tape.

Delta Save Functions

The Delta Save Facility provides the following additional or changed Adabas utility functions:

ADASAV

The following ADASAV functions have been added for use with the Delta Save Facility:

- **MERGE** to combine the latest full save result with one or more subsequent delta save results, in order to create a new full save tape, or to combine multiple delta save results into a single “consolidated” delta save tape;
- **RESTORE DELTA** to combine the latest full save result with one or more subsequent delta save results, in order to restore the combination directly into the database;
- **SAVE DELTA** to save only those blocks changed since the last full or delta save operation.

As the result of “online” SAVE DELTA operations, the images of changed database blocks are written to the Delta Save images (DSIM) dataset, a direct access dataset that must be located on disk. The DSIM dataset is specified as

- **DDDSIMR1** on BS2000, OS/390, z/OS, z/VM, and VM/ESA systems, or
- **DSIMR1** on VSE/ESA systems.

New datasets used for the MERGE and RESTORE DELTA functions are specified as

- **DDFULL** and **DDDEL1**, **DDDEL2**, ... **DDDEL8** on BS2000, OS/390, z/OS, z/VM, and VM/ESA systems, or
- **FULL** and **DEL1**, **DEL2**, ... **DEL8** on VSE/ESA systems.

ADARES

The following ADARES functions have been revised for use with the Delta Save Facility:

- COPY to extract database block images written during an online save operation;
- PLCOPY to extract database block images written during an online save operation.

ADAREP

ADAREP utility operation has been revised to include Delta Save usage, save (full and delta), and logging information.

ADAFRM

The following ADAFRM functions have been added for use with the Delta Save Facility:

- DSIMFRM to format the Delta Save images (DSIM) dataset, which is used to hold images of changed Adabas blocks during online save/merge operation;
- DSIMRESET to reset the DSIM dataset, making it usable for later online save/merge operation.

ADAPRI

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

ADAULD

The ADAULD utility has been modified to 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.

INSTALLATION

This chapter describes the prerequisites and installation procedure for installing the Delta Save Facility.

Supported Operating Systems

The Delta Save Facility can be installed on the following operating systems at the current level of support provided by Adabas:

- BS2000 (SNI)
- FACOM (Fujitsu)
- OS/390 and z/OS
- VM/ESA and z/VM
- VSE/ESA

For information about other compatible or derivative operating systems, consult your Software AG marketing or technical support representative.

Prerequisites for Installing Delta Save

Delta Save version 7.4 requires the following products/versions:

- Adabas version 7.4
- Adabas Online System (AOS) demo version delivered with Adabas version 7.4 or alternatively, the full version AOS selectable unit. The load module AOSASM from the Adabas 7.4 load library must be included in the link of the Natural nucleus.

If you want to use online facilities to restart an interrupted online save operation after a system or other user ABEND (see page 134), you will need to use the selectable unit AOS.

Delta Save Installation Procedure

To install Delta Save, perform the following steps:

1. Install the ADADSF load modules in the load library used for running the Adabas nucleus and utilities.
2. If you plan to perform save operations online, allocate space for the DSIM dataset and format the dataset with the ADAFRM DSIMFRM function, as described in chapter 4.
3. Add the ADARUN parameter DSF=YES to the nucleus and utility jobs used to operate the database (DSF=YES is required for the nucleus and the DSF-related functions of the ADASAV, ADARES, and ADAULD utilities; it is optional for the other utilities. See the section **ADARUN Parameter DSF** on page 38 for more information);
4. To automatically submit a delta save job when the DLOG area usage exceeds a certain threshold, provide the delta save user exit along with the job control statements it requires, and specify the ADARUN DSFEX1 parameter.
5. Start the nucleus (with DSF=YES);
6. Using the Adabas Online System (either the demo or full version) “Install DSF” special function described on page 47, allocate the DSF logging area;
7. Perform a normal Adabas database save operation (ADASAV SAVE) to create a full save dataset and to enable Delta Save logging. If the save operation is online, merge the online full save dataset with the DSIM dataset to produce an offline full save, and to free the DSIM dataset for further use.

VSE/ESA Considerations for Installing Delta Save

Defining the Delta Save Sublibrary

Delta Save users must define an additional sublibrary in the Adabas library for the Delta Save components. A sample job to accomplish this is as follows:

```
// JOB ADEDEF          DEFINE NON-VSAM DELTA SAVE SUBLIB
// OPTION LOG
// DLBL SAGLIB,'ADABAS.Vvrs.LIBRARY',2099/365,SD
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=vvvvvv,SHR
// EXEC LIBR
//   DEFINE S=SAGLIB.ADEvrs REUSE=AUTO R=Y
//   LD L=SAGLIB OUTPUT=STATUS
/*
/ &
```

—where

SYS010 is the logical unit for the Adabas library
 vvvvvv is the volume for the Adabas library
 vrs is the Adabas version/revision/system maintenance (SM) level

Restoring the Delta Save Sublibrary

A sample job to restore the Delta Save components is as follows:

```
// JOB ADERST          RESTORE NON-VSAM DELTA SAVE
// OPTION LOG
// ASSGN SYS006,cuu
// PAUSE MOUNT ADABAS INSTALL TAPE cuu
// MTC REW,SYS006
// MTC FSF,SYS006,tt
// DLBL SAGLIB,'ADABAS.Vvrs.LIBRARY'
// EXTENT SYS010
// ASSGN SYS010,DISK,VOL=vvvvvv,SHR
// EXEC LIBR
//   RESTORE SUB=SAGLIB.ADEvrs          -
//   TAPE=SYS006 LIST=Y R=Y
//   LD SUB=ADEvrs OUTPUT=NORMAL
/*
// MTC REW,SYS006
/ &
```

—where

SAGLIB	is the Adabas library name
SYS010	is the logical unit for the Adabas library
SYS006	is the Adabas installation tape
cuu	is the physical unit address of the tape drive
tt	is the number of tape marks to space forward (see the <i>Report of Tape Creation</i>)
vvvvvv	is the volume for the Adabas library
vrs	is the Adabas version/revision/system maintenance (SM) level

Cataloging Procedures for Defining the Delta Save Libraries and Files

Sample JCL for cataloging the VSE/ESA procedures is available in member ADEPROC.X on the Delta Save sublibrary.

Job ADEPROC contains procedures ADAVvLIB and ADAVvFIL to define the Delta Save Adabas libraries and files.

Customize these procedures before cataloging them into the procedure library.

These procedures can then be used by Delta Save users for all subsequent Adabas jobs.

Sequential File Table

This section summarizes the sequential files added or changed by the Adabas ADASAV utility for Delta Save operation. BS2000, VM/ESA, z/VM, OS/390, z/OS and OS-compatible files have “DD...” names (DDFULL, DDDEL1, etc.); VSE/ESA file names are without the “DD...” prefix. Appendix A of either volume of the *Adabas Utilities Manual* contains a listing of all sequential files, by utility.

Utility	File Name	VSE Tape SYS	Out	In	BLKSIZE by device	Concatenation
ADASAV	DD/FULL	30		x		Yes
	DD/DEL1	31		x		Yes
	DD/DEL2	32		x		Yes
	DD/DEL3	33		x		Yes
	DD/DEL4	34		x		Yes
	DD/DEL5	35		x		Yes
	DD/DEL6	36		x		Yes
	DD/DEL7	37		x		Yes
	DD/DEL8	38		x		Yes

Note:

For VSE users, the highest logical unit is SYS038 for the new ADASAV MERGE function. Be sure to provide enough programmer logical units to run the ADASAV MERGE function in the desired partition.

Applying Fixes to Delta Save

Fixes to Delta Save follow the same guidelines as fixes to Adabas. See the *Adabas Installation Manual* for information about applying fixes to Adabas.

Applying Fixes to Delta Save Using MSHP PATCH

A sample job to insert a fix to Adabas Delta Save facility using MSHP PATCH is as follows:

```
// JOB PATCHD          APPLY PATCH TO ADABAS DELTA SAVE
// OPTION LOG
// EXEC PROC=ADEVvLIB
// EXEC MSHP
    PATCH SUBLIB=saglib.ADEVrs
    AFFECTS PHASE=phasenam
    ALTER offset vvvv : rrrr
/*
/ &
```

—where

vrS is the Adabas version/revision/system maintenance (SM) level
saglib is the Adabas library name in the procedure ADEVvFIL
phasenam is the Adabas phase to be zapped
offset is the hexadecimal offset into the phase
vvvv is the verify data for the ZAP
rrrr is the replace data for the ZAP

The above sample JCL is available in member MSHPATD.X on the Delta Save sublibrary.

Applying Fixes to Delta Save Using MSHP CORRECT

Because Delta Save is installed into a separate sublibrary (ADEvrs) and has a separate product code, it is maintained as a separate MSHP component. Delta Save users must therefore run a MSHP ARCHIVE job for Delta Save.

Note:

This job uses the history file identified by the IJSYSHF label in the VSE standard label area.

```
// JOB ARCHIVE          ARCHIVE ADABAS DELTA SAVE
// OPTION LOG
// EXEC PROC=ADAVvLIB
// EXEC MSHP
ARCHIVE ADEvrs
  COMPRISES 9001-ADE-00
  RESOLVES 'SOFTWARE AG - ADABAS DSF Vv.r'
ARCHIVE 9001-ADA-0E-vrs
  RESIDENCE PRODUCT=ADEvrs -
            PRODUCTION=saglib.ADEvrs -
            GENERATION=saglib.ADEvrs
/*
/ &
```

—where

vrs is the Adabas version/revision/system maintenance (SM) level
saglib is the Adabas library name in the procedure ADEVvFIL

The above sample job is available in member MSHPARCD.X on the Delta Save sublibrary.

A sample job illustrating the use of MSHP CORRECT to install a fix to Adabas Delta Save is as follows:

```
// JOB CORRECT          APPLY FIX TO ADABAS DSF
// OPTION LOG
// EXEC MSHP
CORRECT 9001-ADE-00-vrs : ATnnnnn
AFFECTS MODE=modname
ALTER  offset vvvv : rrrr
INVOLVES LINK=lnkname
/*
/ &
```

—where

vrs	is the Adabas version/revision/system maintenance (SM) level
nnnnn	is the Adabas fix number
modname	is the Adabas object module to be zapped and then relinked
offset	is the hexadecimal offset to the beginning of the ZAP
vvvv	is the verify data for the ZAP
rrrr	is the replace data for the ZAP
lnkname	is the link book for the phase affected

The above sample JCL is available in member MSHPCORD.X on the Delta Save sublibrary.

The following link books are defined on the Delta Save sublibrary :

ADADSFN
ADADSFR
ADADSFS
ADAMGS

User ZAPs to Change Dataset Options

The “verify/replace” values (VER/REP) shown in the following table permit user-specified device types to be changed (“zapped”) for VSE/ESA files, and/or changes to the rewind options for BS2000 files. The *Adabas Installation Manual* contains a complete table of values.

VSE/ESA ZAP Table for Delta Save

Utility	VSE/ESA File*	Default SYS Number	PTT Offset*	VER	REP
ADASAV	FULL	SYS030	2D8	1A1E	1Axx
	DEL1	SYS031	2E8	1A1F	1Axx
	DEL2	SYS032	2F8	1A20	1Axx
	DEL3	SYS033	308	1A21	1Axx
	DEL4	SYS034	318	1A22	1Axx
	DEL5	SYS035	328	1A23	1Axx
	DEL6	SYS036	338	1A24	1Axx
	DEL7	SYS037	348	1A25	1Axx
	DEL8	SYS038	358	1A26	1Axx

BS2000 ZAP Table for Delta Save

The tape rewind defaults in the following table can be changed to

0000: rewind
 1000: no rewind
 2000: rewind/unload

All offset values are hexadecimal:

Utility	BS2000 File*	PTT Offset*	Default
ADASAV	FULL	2DC	2000
	DEL1	2EC	2000
	DEL2	2FC	2000
	DEL3	30C	2000
	DEL4	31C	2000
	DEL5	32C	2000
	DEL6	33C	2000
	DEL7	34C	2000
	DEL8	35C	2000

* The PTT base address is CSECT I_PTT in module ADAIOR. Add the PTT offset value to this base address to obtain the complete address.

ADMINISTRATION

The DBA is concerned with four types of administrative activity for the Delta Save Facility:

- Setting the ADARUN DSF parameter for Delta Save operation mode

The ADARUN DSF parameter must be set to DSF=YES to enable Delta Save operation. See page 38 for more information.

In addition, the ADARUN DSFEX1 parameter can be set if a user exit will be used to automate delta save job submission. See page 40.

- Starting, controlling, and ending Delta Save operation with Adabas basic online services

The Delta Save online functions allow you to

- display Delta Save status;
- install, change, or remove the Delta Save logging (DLOG) area;
- disable Delta Save logging activity.

For more information about selecting online menus and functions, see the *Adabas DBA Reference Manual*.

- Allocating the Delta Save logging (DLOG) area

Delta Save uses the DLOG area within the Associator to keep track of the RABNs of all blocks changed during normal nucleus operation. This information is used in Delta Save operations. See page 55 for information about allocating the DLOG area.

- Allocating the Delta Save Images (DSIM) dataset

The DSIM dataset holds the images of all blocks changed during an online save. Following the save, these blocks are used to construct a full or delta save tape for restoring the database later. See page 57 for information about allocating the DSIM dataset.

ADARUN Parameter DSF

The ADARUN parameter DSF specifies the Delta Save operation mode. For more information about ADARUN parameters, see the *Adabas Operations Manual*.

Parameter	Use	Possible Values	Default
DSF	Specifies Delta Save mode	YES NO	NO

This parameter is only valid if the Delta Save Facility is installed on your system. If Delta Save is not installed in the system, specifying this parameter may cause an error.

Value	Meaning
-------	---------

YES	Run this nucleus or utility (as specified in the ADARUN PROGRAM parameter) in Delta Save mode. This enables Delta Save functions such as installing or removing the DLOG area, performing Delta Save operations, or merging full and/or delta save tapes. Specifying DSF=YES causes certain Delta Save program modules to be loaded.
NO	Run this nucleus or utility in non-Delta Save mode. If DSF=NO (the default) is specified or the parameter is omitted, Delta Save functions cannot be performed. For the nucleus, specifying DSF=NO after specifying DSF=YES switches from Delta Save to non-Delta Save mode. If a DLOG area is defined when the nucleus is started, and DSF=NO is specified, the nucleus removes the DLOG area. If a DLOG area is defined when the nucleus is started, the DSF parameter may not be omitted. You can also remove the DLOG area from the database using the “Remove DSF” command (see the section Remove Delta Save Logging on page 54).

Example:

ADARUN PROG=ADANUC,DSF=YES

Run the nucleus in Delta Save mode; Delta Save-related functions can be performed.

DSF=YES is required for the nucleus and for Delta Save functions of the ADASAV and ADARES utilities; it is optional for the other functions and utilities.

Note:

*If a DLOG area is defined when the nucleus is started, the DSF parameter may **not** be omitted.*

DSF=YES is required for

- ADASAV SAVE database (if a DLOG area exists);
- ADASAV SAVE DELTA;
- ADASAV MERGE;
- ADASAV RESTORE database*;
- ADASAV RESTORE DELTA;
- ADARES PLCOPY (only if online saves are performed);
- ADARES COPY (only for rebuilding the DSIM dataset).
- ADAULD UNLOAD FILE (from full and delta save tapes with or without DSIM datasets)

Examples of functions and utilities for which DSF=YES is optional include

- ADASAV SAVE FILE
- ADASAV SAVE INCREMENTAL
- ADASAV RESTORE FILE
- ADARES REGENERATE
- ADAREP . . .

* Full save tapes (merged or unmerged) are restored with the normal “RESTORE database” function. In this case:

- if DSF=YES is specified or the DSF parameter is omitted, ADASAV restores the database including all Delta Save status information; that is, the restored database has a DLOG area, the delta save ID equals that of the save tape, and Delta Save logging is enabled.
- if DSF=NO is specified, ADASAV restores the database without the Delta Save status information and the Delta Save Facility is no longer installed on the restored database.

ADARUN Parameter DSFEX1

The ADARUN parameter DSFEX1 names the delta save user exit that reads the job stream from a dataset specified in the nucleus JCL. The user exit is loaded if the ADARUN parameter DSF=YES is specified for an Adabas nucleus run. For more information about ADARUN parameters, see the *Adabas Operations Manual*.

Parameter	Use	Default
DSFEX1=exit-name	Specifies name of the delta save user exit	none

This parameter is only valid if the Delta Save Facility is installed on your system. If Delta Save is not installed in the system, specifying this parameter may cause an error.

Automatic Delta Save Job Submission

An overflow of the Delta Save logging (DLOG) area makes a full save operation necessary outside the normal schedule. Although the Delta Save Facility prints warning messages to the operator console as the DLOG area fills, these are easily overlooked as they scroll over the console screen buried among unrelated messages from other jobs. Even if such messages were highlighted or fixed on the operator screen, it would still be necessary for the operator to discover the messages and react properly by submitting an online SAVE DELTA job before the DLOG area is completely filled.

The Delta Save Facility instead provides a user exit named in the ADARUN DSFEX1 parameter that automatically submits the necessary job when the DLOG reaches a user-defined threshold percentage (1–99) of the total area size. No operator intervention is required. The job resets the DLOG area to start another cycle.

Activating the Sample User Exit

OS/390 or z/OS

The sample user exit is delivered on OS/390 and z/OS as source and as a load module that can be used without change or reassembly.

The source form of the user exit DSFUEX1 is delivered in the Adabas source library. The job to assemble the user exit, ASMDUEX1, is located in the Adabas jobs library. The jobs library also contains a sample (DSFUJOB) of the job submitted by DSFUEX1 that

- runs an online delta save;
- switches the PLOG with FEOFPL causing the PLOG containing the SYN2 checkpoint to be copied to the DSIM dataset using user exit 2 (dual logging) or 12 (multiple logging); and
- merges the actual online delta save with the most recently created delta save to produce a new consolidated delta save.

On successful completion of DSFUJOB, the DLOG area is reset and the DSIM dataset is set to a status 'unused'. The next delta save job can then be submitted normally.

Activate the sample user exit as follows:

1. In addition to ADARUN DSF=YES, specify ADARUN DSFEX1=DSFUEX1 for the Adabas nucleus.
2. Supply the delta save job control that the user exit is supposed to submit under the DD-name/DLBL/link-name SAVEJOB.

Provide the following DD statement:

```
//INTRDR2 DD SYSOUT=(*,INTRDR)
```

Note:

The same DD statement is used by the sample user exit 2 or 12 for submitting PLCOPY or CLCOPY jobs.

3. The sample user exit is set to submit the supplied job when the DLOG area usage reaches 75 percent. You can modify this threshold value by
 - changing, assembling, and linking the supplied user exit source; or
 - zapping the supplied user exit load module as follows:

```
NAME DSFUEX1  
VER 003E 004B  
REP 003E 00xx
```

—where “xx” is the desired percent value in hexadecimal notation.

VSE/ESA

The sample user exit is delivered on VSE/ESA as source that must be assembled and linked.

Activate the sample user exit as follows:

1. In addition to ADARUN DSF=YES, specify ADARUN DSFEX1=DSFUEX1 for the Adabas nucleus.
2. Modify the source in member DSFUEX1.A on the Delta Save sublibrary to match the JCL requirements of the batch job that is submitted when the DLOG threshold is reached. This job
 - runs an online delta save;
 - switches the PLOG with FEOFPL causing the PLOG containing the SYN2 checkpoint to be copied to the DSIM dataset using user exit 2 (dual logging) or 12 (multiple logging); and
 - merges the actual online delta save with the most recently created delta save to produce a new consolidated delta save.

On successful completion of DSFUJOB, the DLOG area is reset and the DSIM dataset is set to a status 'unused'. The next delta save job can then be submitted normally.

The DLOG threshold value may also be modified. By default, the threshold is set to 75%.

3. Run a job to assemble and link the updated user exit DSFUEX1.

Sample JCL for this step is provided in member DSFU1AL.X on the Delta Save sublibrary.

BS2000

The sample user exit is delivered on BS2000 as source and as an object module that can be used without change or reassembly.

The source form of the user exit DSFUEX1 is delivered in the Adabas source library. The job to assemble the user exit, ASMDUEX1, is located in the Adabas jobs library.

The jobs library also contains a sample (DSFUJOB) of the job submitted by DSFUEX1 that

- runs an online delta save;
- switches the PLOG with FEOFPL causing the PLOG containing the SYN2 checkpoint to be copied to the DSIM dataset using user exit 2 (dual logging) or 12 (multiple logging); and
- merges the actual online delta save with the most recently created delta save to produce a new consolidated delta save.

On successful completion of DSFUJOB, the DLOG area is reset and the DSIM dataset is set to a status 'unused'. The next delta save job can then be submitted normally.

Activate the sample user exit as follows:

1. In addition to ADARUN DSF=YES, specify ADARUN DSFEX1=DSFUEX1 for the Adabas nucleus.
2. Customize the job DSFUJOB as necessary for functionality, database ID, and ID table name using the DSFUJOB example above; save it in the location that is defined by step 3.
3. If the machine can process job variables, place the following statement in the delta save nucleus job:

```
/SET-JV-LINK LINK-NAME=*DDJBDSF, -
/          JV-NAME=#JV.DSF
/MODIFY-JV  JV-ID=JV-NAME (JV-NAME=#JV.DSF) , -
/          VALUE='ENTER-JOB FROM-FILE=*LIB-ELEM(, -
/LIB=<library>, ELEM=<dsfjob>), -
/JOB-CLASS=JCDSF'
```

—where

<library> is the library; and

<dsfjob> is the member where the customized job from step 2 is stored.

If the machine cannot process job variables, Adabas attempts to submit the job

(ADABAS.SRC,DSFUJOBP)

An example job that allows the database ID to be supplied as a parameter is supplied in the Adabas source library ADAvrs.SRC.

If this is not satisfactory, you can modify the supplied source DSFUEX1.

User Exit Interface

The Adabas Delta Save Facility calls the user exit during initialization to determine the DLOG area usage threshold specified in the exit. The exit is thereafter called whenever the DLOG usage reaches the specified threshold.

Optionally, the user exit may initialize its operation. It may store any value in the USERENV field of the parameter block to keep track of its resources. This field is considered as “owned” by the user exit and is supplied again for all subsequent executions of the exit.

The user exit is called again during termination to do any necessary finishing or ‘cleanup’ work.

Parameters

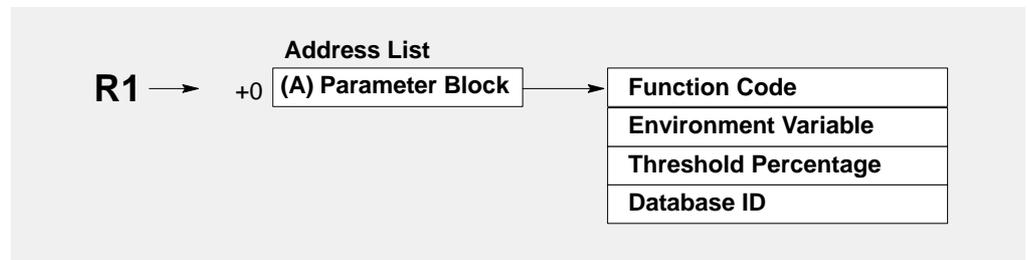


Figure 3-1: Delta Save Facility User Exit Parameters

DSECT of the Parameter Block

PARAM	DSECT		
FNCODE	DS	F	FUNCTION CODE:
*			01 - INITIALIZATION
*			02 - THRESHOLD REACHED
*			03 - FINALIZATION
USERENV	DS	A	USER ENVIRONMENT
PERCENT	DS	F	THRESHOLD PERCENTAGE
DBID	DS	F	DATABASE ID

Special Online Functions

When the Adabas Delta Save Facility is installed on your system, either the demo or full version of Adabas Online System (AOS) provides access to the optional Delta Save control screens.

```

14:40:40          ***** A D A B A S ONLINE SYSTEM *****          1999-01-28
                      - Main Menu -                                PMAIN02

Code  Basic Services          Code  Other Services
----  -
A     Session monitoring      1     Cache Services
C     Checkpoint maintenance  2     Delta Save Facility
F     File maintenance        3     Trigger Maintenance
*     Database maintenance    4     AOS Security
O     Session opercoms        5     Transaction Manager
R     Database report         6     Adabas Statistics
*     Space calculation       7     Vista
?     Help                    8     Fastpath
.     Exit                    9     SAF Security
----  -

Code ..... _
Database ... 105      (RD-105)

Command ==>
PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help                Exit

```

Note:

The exact sequence of screens and display characteristics (intensified values, etc.) displayed when using AOS depends on the configuration, the features installed on the system, and the current operating status of the system, and therefore may differ from the features and characteristics described in this chapter.

The remainder of this section describes how to use the Delta Save control screens to install, change, disable, and remove Delta Save logging from the system.

Logging On to the Adabas Delta Save Facility

- ▶ Select the “Delta Save Facility” option on the Adabas Online System (AOS) Main Menu.
- When the Delta Save Facility is active, the Delta Save Facility main menu is displayed.

```

13:16:45          ***** A D A B A S  BASIC SERVICES *****      1997-02-20
Enabled          - Delta Save Facility -                          PDSF002

                Code      Function
                ----      -
                A         Allocation/Status of DSF
                C         Change DSF logging area
                D         Disable DSF logging
                E         Enable DSF logging
                F         Files modified by utilities
                I         Install DSF logging area
                R         Remove/Deinstall DSF
                S         Size Estimate for Delta
                ?         Help
                .         Exit
                -----

                Code ..... _
                Database ID .. 105      (ADA-MPM105)

                Command ==>
                PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
                Help           Exit           Menu

```

This menu indicates the status of the Delta Save Facility in the upper left corner:

- | | |
|---------------|--|
| Not installed | The ADARUN DSF=YES parameter is set but the Delta Save Facility is not yet installed. |
| Enabled | The Delta Save Facility is currently active and functioning normally. |
| Disabled | Delta Save logging has been deactivated. |
| Initializing | The Delta Save Facility has been installed but the initialization process has not yet completed. If this status continues, deinstall and then reinstall the product. |

Installing Delta Save Logging

If the Delta Save Facility is not yet installed, option I displays the following window:

```

14:55:16          ***** A D A B A S  BASIC SERVICES  *****          1997-02-26
Not installed          - Delta Save Facility  -                               PDSF002

+-----+
| 15:16:44          Install DSF                               1997-02-26 |
| DSF Status .... Not installed |
| Specify the values for the following parameters:- |
| DLOG Area Size ..... (in Blocks) |
| DLOG Area Location .. (1st RABN) optional |
| Device Type ..... 3380          optional |
| PF1----- PF2----- PF3----- PF4----- PF6----- PF12 |
| Help          Exit          Menu |
+-----+

Command ==>
PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help          Exit          Menu

```

The DLOG area must be defined before Delta Save can begin operation.

Use the Install DSF screen to define the Delta Save logging (DLOG) area and thereby install Delta Save logging. Enter the DLOG size in blocks, and, optionally, the starting RABN in the Associator and/or the device type to be used for allocation:

- **DLOG Area Size** (required) is the number of Associator blocks to be allocated. The area must be large enough to hold all blocks changed between two consecutive Delta Save operations. About 5 bytes of DLOG area storage are needed for every database write operation. To calculate the size of the DLOG area, see page 55.
- **DLOG Area Location** (optional) is the first RABN to be allocated.
- **Device Type** (optional) is the device type to be used for DLOG area allocation.

When the DLOG area is defined, you are returned to the Delta Save Facility main menu and the message

DSF function Install DSF successfully executed

—is displayed. The status of the Delta Save Facility in the upper left corner of the screen is “disabled”. The “disabled” state continues until the next successful full save operation is performed with the ADASAV SAVE utility function.

Enabling Delta Save Logging

If the Delta Save Facility is disabled, option E displays the following window:

```

13:20:36          ***** A D A B A S  BASIC SERVICES *****          1997-02-20
Disabled          - Delta Save Facility -                               PDSF002

+-----+
|
| DSF is currently disabled and needs to be
| enabled before it can start logging.
| To enable DSF, a FULL database backup must
| be run using ADASAV SAVE. Once the FULL
| database save completes, DSF will become
| enabled automatically.
|
+-----+

          .          Exit
          -----

Code ..... e
Database ID .. 105      (ADA-MPM105)

Command ==>
PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help          Exit          Menu

```

Once a successful full save operation is performed with the ADASAV SAVE utility function, Delta Save logging is automatically enabled.

Displaying Current Allocation Information

You can use option A to display additional status and allocation information:

```

13:16:45          ***** A D A B A S  BASIC SERVICES *****          1997-02-20
Enabled          - Delta Save Facility -                               PDSF002

+-----+
|                                     |
|               Allocation Display   |
|                                     |
| DSF Status ..... Enabled         |
|                                     |
| Last full save no ... 20          | on 1997-01-30 at 11:19:04 |
| Last delta save no .. 5           | on 1997-03-03 at 05:06:56 |
|                                     |
| DLOG area size ..... 500         | blocks                |
| DLOG area usage ..... 5%         |                        |
| DLOG area location .. 6292        | (start RABN)           |
| Device type ..... 3380           |                        |
|                                     |
|               Press 'Enter' to    | continue/exit         |
|                                     |
+-----+

Command ==>
PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help                Exit                Menu

```

In addition to information about the last full save and the last delta save, this display provides information about the current DLOG area size and location, the device type, and the percentage of the DLOG area size currently being used.

Displaying Files Modified by Utilities

Option F, files modified by utilities, indicates the name and number of the files changed on the first line and other information for the file on a second line.

The second line indicates the areas logged (IX for index; DS for Data Storage; AC for address converter) as well as block use and modification statistics for each file:

- total blocks changed in all logged areas;
- total DS/AC blocks used, changed, and the percentage of the used that were changed;
- total index blocks used, changed, and the percentage of the used that were changed.

Another line provides separate totals for the number of DS/AC blocks and index blocks changed for all files listed.

```

12:55:42          ***** A D A B A S  BASIC SERVICES *****          1997-03-21
DBID 105                - Display DSF Files -                               PDSFF02

File No   Logged   Tot Blks   AC/DS   AC/DS Blk D %   Index   Indx Blks I %
/Name     Areas    Changed    Blks Used   Changed   Log Blks Used   Changed Log
-----
13 - ADABAS-LISTING
      IX                1081      281          0  0       1081      1081 100
80 - PRIME-TIME
      IX                 3         2          0  0         3         3 100

      Accumulated Totals:  DS/AC ..          0       Index ..          1084

PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help   Repos   Exit                --         -         +         Menu

```

PF2 invokes a window that allows you to reposition the list to a specified file number.

Estimating the Size of a Delta Save Operation

Option S, size estimate for delta, displays the estimated number of changed blocks. This number provides an estimate of the number of database blocks that would be saved in a Delta Save operation at the current time.

Allow a few seconds for the system to perform the calculation. The estimate is not precise: the actual size may be larger or smaller.

```

17:05:45          ***** A D A B A S  BASIC SERVICES  *****          1997-03-21
Enabled          - Delta Save Facility -          PDSF002
+-----+
|          - Estimated Number of Changed Blocks -          |
|                                                         |
| Fixed number of RABNs to be saved ....          530 blocks |
| Index RABNs changed by utilities .....          1084 blocks |
| DS/AC RABNs changed by utilities .....           0 blocks |
|                                                         |
| Number of RABNs logged in DLOG Area .. ca.          2000 blocks |
|                                                         |
| Approximately          3600 blocks will be saved with the next |
| SAVE DELTA function. |
|                                                         |
|          Press 'Enter' to continue/exit          |
+-----+
Command ==>
PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help          Exit          Menu

```

The fixed area includes the blocks for the first 30 RABNs, all FDTs and all FCBs.

Separate totals are provided for the number of index blocks as opposed to the number Data Storage and address converter blocks changed by utilities.

The calculated number of blocks for the DLOG area is a rounded number.

The total of all blocks to be saved is approximate to the nearest 100.

Disabling Delta Save Logging

Disabling Delta Save logging stops the logging and prevents any further ADASAV SAVE DELTA utility operations.

Option D displays a warning about disabling the Delta Save Facility and asks for confirmation.

```

13:18:20          ***** A D A B A S  BASIC  SERVICES  *****          1997-02-20
Enabled          - Delta Save Facility  -          PDSF002

+-----+
|                                     |
|               - Warning -         |
|                                     |
| When DSF logging is disabled, you may no |
| longer perform Delta Save operations. To |
| enable DSF logging again, you will need to |
| do a full database save using ADASAV SAVE. |
|                                     |
| To confirm that you want to disable DSF |
| logging, enter 'CONFIRM' ... _____ |
|                                     |
+-----+

Code ..... d
Database ID .. 105      (ADA-MPM105)
DSFF04: Size Estimate found nothing for this request
Command ==>
PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help              Exit                               Menu

```

Note:

It is not possible to start or restart Delta Save logging with an online function. Delta Save logging is started (“enabled”) only after successful completion of a full save operation.

If you confirm, you are returned to the Delta Save Facility main menu and the following message is displayed:

DSF function Disable DSF logging successfully executed.

Once Delta Save logging is disabled, you can either change the DLOG area (see the section **Changing the DLOG Area** on page 53) or logically remove Delta Save logging from the system (see the section **Removing Delta Save Logging** on page 54).

Changing the DLOG Area

You can close the current DLOG area and allocate a new and possibly larger DLOG area elsewhere in the Associator. However, the DLOG area specification can only be changed when Delta Save logging is “disabled”.

Option C displays a the following window allowing you to change the values for the DLOG area size and location, and the device type.

```
13:19:01          ***** A D A B A S  BASIC SERVICES  *****          1997-02-20
Disabled          - Delta Save Facility -                               PDSF002
```

```
+-----+
| 13:19:27          Change DLOG area          1997-02-20 |
| DSF Status .... Disabled |
| Change the values for the following parameters:- |
| DLOG Area Size ..... 500          (in Blocks) |
| DLOG Area Location .. 6292        (1st RABN) optional |
| Device Type ..... 3380          optional |
| PF1----- PF2----- PF3----- PF4----- PF6----- PF12 |
| Help          Exit          Menu |
+-----+
```

```
DSF function Disable DSF logging successfully executed
Command ==>
```

```
PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help          Exit          Menu
```

In the “Change DLOG Area” window, enter the new DLOG size in blocks and optionally, the starting RABN in the Associator and/or the device type to be used for allocation.

Removing Delta Save Logging

Once Delta Save logging is “disabled”, you can logically remove the Delta Save Facility from the system. This means that the DLOG is deallocated and the Delta Save logging status is reset to “not installed”.

Option R displays the following window warning you about removing the Delta Save Facility and asking for confirmation.

```

13:20:36          ***** A D A B A S  BASIC  SERVICES  *****          1997-02-20
Disabled          - Delta Save Facility -                               PDSF002

+-----+
|                                     |
|               - Warning -         |
|                                     |
| When DSF logging is removed, Delta Save |
| operations will no longer be possible. To |
| use DSF again, a completely new install |
| (including a full database save with    |
| ADASAV SAVE) must be done.           |
|                                     |
| To confirm the removal of DSF, enter   |
| 'CONFIRM' ... _____            |
|                                     |
+-----+

Code ..... r
Database ID .. 105      (ADA-MPM105)
Function interrupted with PF-key
Command ==>
PF1----- PF2----- PF3----- PF4----- PF6----- PF7----- PF8----- PF12-----
Help                Exit                                Menu

```

If you confirm, you are returned to the Delta Save Facility main menu and the following message is displayed:

DSF004 : Delta Save Facility terminated at user's request.

Calculating the DLOG Area Size

To calculate a DSF logging area size (DLOGSIZE), use the following procedure as a guide:

1. First, determine
 - the estimated number of Associator/Data Storage write I/O operations that will occur between consecutive delta save operations (WIOS);
 - the Associator dataset block size (ASSOBSIZ) for the Associator device from the tables in appendix A of the *Adabas Operations Manual*; and
 - the security factor (DLOGFAC), a multiplier to ensure that no DLOG overrun occurs.
2. Calculate the approximate number of logged RABNs per DLOG block (LOGRABNS):

$$\text{LOGRABNS} = \frac{\text{ASSOBSIZ}}{5}$$

3. Calculate the needed number of DLOG detail blocks (DLOGBLKS):

$$\text{DLOGBLKS} = \frac{\text{WIOS}}{\text{LOGRABNS}}$$

4. Calculate the DLOG block count (DLOGRAW), including the security factor:

$$\text{DLOGRAW} = \text{DLOGBLKS} \cdot \text{DLOGFAC}$$

The value DLOGSIZE is the calculated size of the DLOG area, in blocks.

Example:

The DLOG size is calculated using an average number of I/O operations between Delta Save operations of 1,000,000 based on a 3390 device type:

The approximate number of logged RABNs per block is

$$\frac{2544}{5} = 508$$

The required detail blocks are calculated as

$$\frac{1000000}{508} = 1968$$

The total DLOG block count including the security factor (in this case, 3) is calculated as

$$1968 \cdot 3 = 5904$$

—or 5904 blocks of DLOG area.

This value is entered in the “DLOG Area Size” field.

Calculating the DSIM Dataset Size

The Delta Save Images (DSIM) dataset must be large enough to hold an image of every Associator and Data Storage block written or changed by the database nucleus while an online save operation is active.

The DSIM dataset can be any device type. If an Associator or Data Storage block to be stored in the DSIM dataset does not fit into one DSIM block, it is split into two or more components. Using a device type with a block size large enough to prevent a split reduces the number of DSIM I/Os and thereby improves performance.

The following sequence of calculations can be used to estimate and define the size, in cylinders, of the DSIM dataset (DSIMSIZE):

1. First, determine the following factors:

PLBLKS	estimated number of PLOG blocks written during an online save operation.
PLBLOKSIZ	PLOG block size, by device type. This is available from the block tables in the <i>Adabas Installation Manual</i> .
DSIMBSIZ	DSIM dataset block size for the DSIM device type. The DSIM device type is the same as the sort/temp device type.
DSIMBCYL	DSIM dataset blocks per cylinder for the DSIM device type. Obtain the number by multiplying the “RABNs/track” table value by the “tracks/cylinder”; see the block tables in the <i>Adabas Installation Manual</i> .
SECFAC	security factor, a multiplier to ensure that no DSIM overrun occurs.

2. Convert the PLOG block count (PLBLKS) into a DSIM block count (DSIMBLKS):

$$\text{DSIMBLKS} = \frac{\text{PLBLKS} \cdot \text{PLBLOKSIZ}}{\text{DSIMBSIZ}}$$

When the PLOG block size and the DSIM block size are equal, then this calculation is not necessary; just set DSIMBLKS equal to PLBLKS.

3. Calculate a secured (using the security factor SECFAC) “raw” DSIM size, in blocks:

$$\mathbf{DSIMRAW = DSIMBLKS \cdot SECFAC}$$

4. Round the DSIM size, in blocks, to the next larger cylinder boundary:

$$\mathbf{DSIMSIZE = \frac{DSIMRAW + DSIMBCYL - 1}{DSIMBCYL}}$$

The value DSIMSIZE is the calculated size of the DSIM dataset, in cylinders.

Example:

If the average number of PLOG blocks written per save is 5000, based on a 3390 device type for both the PLOG and the DSIM dataset, the DSIM size is calculated as follows:

$$\frac{5000 \cdot 5724}{8904} = 3214$$

The total “raw” DSIM block count is determined by multiplying the DSIM block count by the security factor—in this case, 2:

$$\mathbf{3214 \cdot 2 = 6248}$$

The “raw” block count is rounded up to the next 3390 cylinder boundary:

$$\frac{6428 + 90 - 1}{90} = 72$$

—or 72 cylinders for the DSIM dataset.

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 manual library. The information in this manual 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 manual for the Delta Save Facility:

Utility	Function	Action
ADAFRM (formatting)	DSIMFRM	Format the Delta Save Images (DSIM) dataset. See page 63.
	DSIMRESET	Reset an incorrect DSIM dataset for reuse. See page 65.
ADAPRI (printing)	DSIMPRI	Print DSIM dataset blocks. See page 73.
ADAREP (reporting)		Functions expanded for Delta Save operation where applicable. See page 79.
ADARES (recovery)	COPY	Copy a sequential log dataset. See page 87.
	PLCOPY	Copy dual or multiple PLOG datasets to a single sequential dataset. See page 92.
ADASAV (save/restore)	MERGE	Combine delta save outputs with each other, and/or with the last full database save output. See page 103.
	RESTORE DELTA	Restore files/database from separate full and delta save outputs. See page 111.
	SAVE	Save the complete database to a full save dataset. See page 127.
	SAVE DELTA	Save the changed portions of the database to a delta save dataset. See page 131.
ADAULD (unloading)	UNLOAD with SAVETAPE	Unload from a full save tape, 1–8 delta save tapes, and optionally the DSIM dataset. See page 153.

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.

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 **Catalog Procedures for Defining Delta Save Library and Files** on page 30.

ADAFRM UTILITY

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

- DSIMFRM (Format the DSIM dataset);
- DSIMRESET (Reset DSIM dataset blocks).

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 <u>ADARUN-device</u> }] [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 <u>ADARUN-device</u> }] [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.

JCL/JCS Requirements and Examples

This section describes the job control information required to run the Delta Save functions of ADAFRM with BS2000, OS/390, z/OS, VM/ESA, z/VM, and VSE/ESA systems, and shows examples of each of the job streams.

BS2000

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>Operations Manual</i>
ADAFRM messages	SYSLST/DDDRUCK		<i>Messages and Codes</i>
ADARUN messages	SYSOUT/ DDPRINT		<i>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

```

OS/390 or 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>Operations Manual</i>
ADAFRM messages	DDDRUCK		<i>Messages and Codes</i>
ADARUN messages	DDPRINT		<i>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=*
```

VM/ESA or 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>Operations Manual</i>
ADAFRM messages	DDDRUCK	disk/terminal/printer	<i>Messages and Codes</i>
ADARUN messages	DDPRINT	disk/terminal/printer	<i>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

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>Operations Manual</i>
ADAFRM messages		printer	SYS009	<i>Messages and Codes</i>
ADARUN messages		printer	SYSLST	<i>Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information on page 61.

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
/*
```


ADAPRI UTILITY

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 <u>ADARUN-device</u>}] [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, OS/390, z/OS, VM/ESA, z/VM, and VSE/ESA systems, and shows examples of each of the job streams.

BS2000

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>Operations Manual</i>
ADAPRI messages	SYSLST/DDDRUCK		<i>Messages and Codes</i>
ADARUN messages	SYSOUT/DDPRINT		<i>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 DDDSIMR1, 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

```

OS/390 or 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>Operations Manual</i>
ADAPRI messages	DDDRUCK		<i>Messages and Codes</i>
ADARUN messages	DDPRINT		<i>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=*
```

VM/ESA or 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>Operations Manual</i>
ADAPRI messages	DDDRUCK	disk/terminal/printer	<i>Messages and Codes</i>
ADARUN messages	DDPRINT	disk/terminal/printer	<i>Messages and Codes</i>

Example

```

DATADEF DDDSIMR1, DSN=ADABASV.V.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

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>Operations Manual</i>
ADAPRI messages		printer	SYS009	<i>Messages and Codes</i>
ADARUN messages		printer	SYSLST	<i>Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information on page 61.

Example

```
// EXEC PROC=ADAVvFIL
// EXEC PROC=ADAVvLIB
// EXEC ADARUN,SIZE=ADARUN
ADARUN PROG=ADAPRI,...
/*
ADAPRI DSIMPRI FROMRABN=...,TORABN=...
/*
```

ADAREP UTILITY

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.

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

D S F   L O G G I N G   A R E A

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
DSF I 3380  2004 I    100          0 I  1340     1439 I    74          0 I
I      I      I      I
-----I-----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, OS/390, z/OS, VM/ESA, z/VM, and VSE/ESA systems, and shows examples of each of the job streams.

BS2000

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

```

OS/390 or z/OS

Dataset	DD Name	Storage	More Information
Associator	DDASSORn	disk	
Data Storage	DDDATARn	disk	
ADAREP parameters	DDKARTE		
ADARUN parameters	DDCARD		<i>Operations Manual</i>
ADAREP messages	DDDRUCK		<i>Messages and Codes</i>
ADARUN messages	DDPRINT		<i>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=3380,DB=99
//DDKARTE DD   *
ADAREP  CPLIST
```

VM/ESA or 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>Operations Manual</i>
ADAREP messages	DDDRUCK	disk/terminal/printer	<i>Messages and Codes</i>
ADARUN messages	DDPRINT	disk/terminal/printer	<i>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=3380 , DB=99
```

Contents of ADAREP CONTROL A1:

```
ADAREP  CPLIST
```

VSE/ESA

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>Operations Manual</i>
ADAREP messages		printer	SYS009	<i>Messages and Codes</i>
ADARUN messages		printer	SYSLST	<i>Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information on page 61.

Example

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


ADARES UTILITY

This section 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.

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** (see page 95).

Syntax

ADARES COPY	DSIMSIZE= DSIM-dataset-size { PLOGNUM= protection-log-num[, SYN1= chkpt-block-num] FROMPLOG= start-session[, TOPLOG= stop-session]} [DSIMDEV= {device-type <u>ADARUN-device</u> }] [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** (see page 95).

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.

JCL/JCS Requirements and Examples

This section describes the job control information required to run the Delta Save functions of ADARES with BS2000, OS/390, z/OS, VM/ESA, z/VM, and VSE/ESA 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.

BS2000

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>Operations Manual</i>
ADARES messages	SYSLST/DDDRUCK		<i>Messages and Codes</i>
ADARUN messages	SYSOUT/DDPRINT		<i>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         DDDSIMR1, 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            DO . PLCOPY . LST
/ ASS-SYSOUT            DO . 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         DDASSOR1, ADABAS . DB010 . ASSOR1, SUP=DISK (SHARE-UPD=YES)
/ SET-FILE-LINK         DDDSIMR1, 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

```

OS/390 or 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>Operations Manual</i>
ADARES messages	DDDRUCK		<i>Messages and Codes</i>
ADARUN messages	DDPRINT		<i>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=*
```

VM/ESA or 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>Operations Manual</i>
ADARES messages	DDDRUCK	disk/terminal/printer	<i>Messages and Codes</i>
ADARUN messages	DDPRINT	disk/terminal/printer	<i>Messages and Codes</i>

Rebuild DSIM Dataset from Sequential Protection Log (ADARES COPY)

```

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

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>Operations Manual</i>
ADARES messages		printer	SYS009	<i>Messages and Codes</i>
ADARUN messages		printer	SYSLST	<i>Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information on page 61.

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= . . .  
/*
```

ADASAV UTILITY

This section 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.

MERGE

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/SAVE_n 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 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 unloaded 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 systems, by using the DDFULL01, DDFULL02, ... link names for the second, third, ... datasets;
- For OS/390, z/OS, z/VM, and VM/ESA systems, by using operating system concatenation capabilities;
- For VSE/ESA 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 starting on page 136. For more information on dataset concatenation on BS2000 and VSE/ESA systems, see the *Adabas Utilities Manual*.

Syntax

ADASAV	MERGE	[DRIVES={ count 1 }] [DSIMDEV={device-type <u>ADARUN-device</u>}] [DSIMWAIT={ seconds 0 }] [NOUSERABEND] [PATTERN=merge-pattern] [PERDRIVE=disks-per-tape] [TEST] [TWO_COPIES]
---------------	--------------	---

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 page 107.

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** starting on page 136.

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.

RESTORE DELTA : Restore Without Prior Merging

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.

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 unloaded 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 systems, by using the DDDELn01, DDDELn02, ... link names, where the last delta save input has the serial number “n”;
- For OS/390, z/OS, z/VM, and VM/ESA systems, by using the available operating systems capabilities;
- For VSE/ESA, 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 starting on page 136. For more information on dataset concatenation on BS2000 and VSE/ESA systems, see the *Adabas Utilities Manual*.

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.

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

If FMOVE is specified, any or all of the parameters ACRABN, ALLOCATION, ASSOVOLUME, 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-database-name]

[OVERWRITE]

[PLOGDEV={PLOG-device-type | ADARUN-device}]

FILES=file-list

[ALLOCATION={**FORCE** | **NOFORCE**}]

[NEWFILES=file-list]

[PASSWORD=password-list]

FMOVE=file-list

[ACRABN=AC-start-rabn-list/

[ALLOCATION={**FORCE** | **NOFORCE**}]

[ASSOVOLUME='ASSO-extent-volume-list']

[DATAVOLUME='DATA-extent-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]

[UI SIZE=UI-size-list]

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}]

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 examples starting on page 125.

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 ASSOVLUM parameter must correspond to the list of files in the FMOVE parameter. If no ASSOVLUM value is to be given for a file, its entry in the list of volumes must be left empty. See the examples starting on page 125.

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 examples starting on page 125.

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 examples starting on page 125.

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 examples starting on page 125.

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.

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, ASSOVOLUME, DATAVOLUME, 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.

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.

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 examples starting on page 125.

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 examples starting on page 125.

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 examples starting on page 125.

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 examples starting on page 125.

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. Refer to the examples starting on page 125 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 examples starting on page 125.

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 examples starting on page 125.

Examples

Example 1:

```
ADASAV RESTORE DELTA,OVERWRITE
ADASAV      PATTERN=FDD
ADASAV      DSIMDEV=8381
ADASAV      DRIVES=4
ADASAV      PLOGDEV=3380,CLOGDEV=3380
```

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 3380. 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.

SAVE : Save Database

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 TT SYN 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** (see page 134).

Syntax

```

ADASAV    SAVE    [BUFNO={ number-of-buffers|1 }]
                [DRIVES={ count|1 }]
                [DSIMDEV={device-type | ADARUN-device}]
                [NOUSERABEND]
                [PERDRIVE=disk-drives-per-tape-drive]
                [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** starting on page 136.

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 ($TT \cdot 1.05$ 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.

SAVE DELTA : Save Changed Database Blocks

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 TTSYN 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** (see page 134).

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 ($TT \cdot 1.05$ 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.

Restarting an Interrupted Save Operation

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.

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 page 65.

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.

JCL/JCS Requirements and Examples

This section describes the job control information required to run the ADASAV functions for the Delta Save Facility with BS2000, OS/390, z/OS, VM/ESA, z/VM, and VSE/ESA 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.

BS2000

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>Operations Manual</i>
ADASAV parameters	SYSDTA/ DDKARTE		
ADARUN messages	SYSOUT/ DDPRINT		<i>Messages and Codes</i>
ADASAV messages	SYSLST/ DDDRUCK		<i>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      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      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        DO.SAV.LST
/ASS-SYSOUT        DO.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

```

OS/390 or 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>Operations Manual</i>
ADASAV parameters	DDKARTE		
ADARUN messages	DDPRINT		<i>Messages and Codes</i>
ADASAV messages	DDDRUCK		<i>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=3380,CLOGDEV=3380
//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
//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.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=*

```

VM/ESA or 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	DDSAVEEn	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>Operations Manual</i>
ADASAV parameters	DDKARTE	disk/terminal/ reader	
ADARUN messages	DDPRINT	disk/terminal/ printer	<i>Messages and Codes</i>
ADASAV messages	DDDRUCK	disk/terminal/ printer	<i>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 DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
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...)

```
DATADDEF DDDATAR1,DSN=ADABASVv.DATA,VOL=DATAV1
DATADDEF DDASSOR1,DSN=ADABASVv.ASSO,VOL=ASSOV1
DATADDEF DDWORKR1,DSN=ADABASVv.WORK,VOL=WORKV1
DATADDEF DDDSIMR1,DSN=ADABASVv.DSIM,VOL=DSIM
DATADDEF DDREST1,DSN=ADABASVv.SAVE040.PART1,MODE=A
DATADDEF DDREST1,DSN=ADABASVv.SAVE040.PART2,MODE=A,CONCAT=1
DATADDEF DDREST1,DSN=ADABASVv.SAVE040.PART3,MODE=A,CONCAT=2
DATADDEF DDREST1,DSN=ADABASVv.SAVE040.PART4,MODE=A,CONCAT=3
DATADDEF DDDEL1,DSN=ADABASVv.DELTA1,MODE=A
DATADDEF DDDEL2,DSN=ADABASVv.DELTA2,MODE=A
DATADDEF DDPRINT,DSN=ADASAV.DDPRINT,MODE=A
DATADDEF DDDRUCK,DSN=ADASAV.DDDRUCK,MODE=A
DATADDEF DDCARD,DSN=RUNSAV.CONTROL,MODE=A
DATADDEF 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 . ASSO , 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 DDRUCK , DSN=ADASAV . DDRUCK , 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 DDRUCK , DSN=ADASAV . DDRUCK , 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

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
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>Operations Manual</i>
ADASAV parameters		reader	SYSIPT	
ADARUN messages		printer	SYSLST	<i>Messages and Codes</i>
ADASAV messages		printer	SYS009	<i>Messages and Codes</i>

* Any programmer logical unit can be used.

See the VSE/ESA-related job information on page 61.

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

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=3380,CLOGDEV=3380
/*
```

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,...
/*

```

ADAULD UTILITY

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.

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** on page 161.

For more information, see the section **Save Tape Input Processing** on page 158.

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 Reference Manual*. 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 Reference Manual* 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           1997-04-23  13:33:25

PARAMETERS:
-----
ADAULD  UNLOAD FILE=3, SAVETAPE

*****
*
* UNLOAD FROM          ONLINE DATABASE SAVE          *
* CREATED AT          1997-04-23  13:32:19          *
* 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 chapter **ADAULD** in the *Adabas Utilities Manual* 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. See page 103. 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 S T A T I S T I C S

```
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 Reference Manual* for more information about user exits.

JCL/JCS Requirements and Examples

This section describes the job control information required to run ADAULD with BS2000, OS/390, z/OS, VM/ESA, z/VM, and VSE/ESA systems and shows examples of each of the job streams.

BS2000

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>Operations Manual</i>
ADAULD parameters	SYSDTA/DDKARTE		
ADARUN messages	SYSOUT/DDPRINT		<i>Messages and Codes</i>
ADAULD messages	SYSLST/DDDRUCK		<i>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 Examples

```

/. ADAULD LOGON
/OPTION MSG=FB,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
    
```

OS/390 or 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>Messages and Codes</i>
ADARUN messages	DDPRINT	printer	<i>Messages and Codes</i>
ADARUN parameters	DDCARD	reader	<i>Operations Manual</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.

OS/390 or 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=3380,DB=99,DSF=YES
//DDKARTE     DD   *
ADAULD   FILE=1,SAVETAPE,PATTERN=FDD
```

VM/ESA or 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
Recovery log (RLOG)	DDRLOGR1	disk	Required for ADARAI
ADAULD messages	DDDRUCK	disk/terminal/printer	<i>Messages and Codes</i>
ADARUN messages	DDPRINT	disk/terminal/printer	<i>Messages and Codes</i>
ADARUN parameters	DDCARD	disk/terminal/reader	<i>Operations Manual</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.

VM/ESA or 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=3380,DB=111,DSF=YES
```

Contents of ADAULD CONTROL A1✧

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

VSE/ESA

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>Messages and Codes</i>

File	Symbolic Name	Storage	Logical Unit	More Information
ADARUN parameters	SYSRDR CARD	reader/ tape/disk		<i>Operations Manual</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 Example

See the VSE/ESA-related job information on page 61.

```
// 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
/*
```



APPENDIX A : GLOSSARY OF TERMS

The terms defined here are referred to in this manual, and apply to both Adabas and the Delta Save Facility.

ADADSF

The Adabas load modules comprising the Delta Save Facility.

ADADSFN module

A load module of Delta Save Facility that is loaded to the Adabas nucleus and contains the program logic for recording changed RABNs in the DLOG area.

ADADSFR module

A load module of Delta Save Facility that is loaded for ADARES executions and contains the program logic for building the DSIM dataset.

ADADSFS module

A load module of Delta Save Facility that is loaded for ADASAV executions and contains the program logic for the input parts of SAVE DELTA, MERGE, and RESTORE DELTA processing.

Adalink

The teleprocessing-monitor-dependent interface module that connects the application/user to Adabas. The actual module name depends on the TP monitor being used; for example, the module name for linking to a batch or TSO program is ADALNK, and for CICS, the module name is ADALNC. The term “Adalink” refers to the module appropriate for the given environment. The terms “Adalink(s) and “ADALNKS” are synonyms.

address converter

Adabas stores each database record in a Data Storage block, identified by a relative Adabas block number (RABN). Each record’s RABN is kept in a table called the address converter. The address converters, one for each database file, are stored in the Associator. Address converter entries are in ISN order (that is, the first entry tells the RABN location of data for ISN 1, the 15th entry holds the RABN location of data for ISN 15, and so on).



address space

The storage area assigned to a program task/work unit. In MVS, an address space is a region; in VSE, a partition; and in BS2000, a task. In this manual, the term “region” is used as a synonym for “partition” and “task”.

consolidated delta save tape

A consolidation of the continual sequence of individual delta save tapes produced since creation of the last full save tape. The full save tape and the consolidated delta save tape together represent the database status following the most recent save operation.

database administrator (DBA)

Controls and manages the database resources. Tasks include defining database distribution, assigning a structure and resources, creating and maintaining programming and operation standards, ensuring high performance, resolving user problems, defining and teaching user training, controlling database access and security, and planning for growth and the integration of new database resource applications and system upgrades. Also known as the database analyst.

delta save

The ADASAV utility operation that saves only those blocks of the Adabas database that have been added or altered since the last full or delta save operation; also, the resulting save dataset.

Delta Save Facility

The optional Adabas feature that provides the ability to perform save operations that save only changed blocks of the database. Using the Delta Save Facility, you can make interim delta save tapes and then combine them into a final normal full database save tape.

delta save exits

ADADSF functional units that are called at certain times during operation to implement delta save functionality.

delta save identifier (DSID)

A DSID identifies each save dataset produced by the Delta Save Facility.

delta save logging

The process of recording changed RABN lists to the delta save logging (DLOG) area.

delta save logging (DLOG) area

The DLOG area is space allocated in the Adabas Associator to record the RABNs of changed Adabas blocks written to the database.

A DLOG area “detail entry” describes a changed block or range of blocks in the DLOG area.

A DLOG area “detail block” is a type of block that contains detail entries.

delta save status

The current status of the Delta Save Facility. The possible status conditions are

not installed	DLOG area has not been defined, although the nucleus may be running with DSF=YES;
disabled	DLOG area has been defined but no delta save logging occurs, and no delta save operations are possible;
enabled	DLOG area is defined and delta save logging is being performed; delta save operations are possible.

delta save images (DSIM) dataset

An interim direct access dataset holding the images of all database blocks changed during an online save operation. These blocks are held for subsequent consolidation with the associated online save dataset. The DSIM dataset is a companion to the related online save output tape.

delta save images (DSIM) directory

A table of DSIM dataset contents which, after consolidation, is used to retrieve the database blocks stored in the dataset in ascending RABN sequence.

expanded file

A logical file comprising physical files in one or more locations. The physical files have the same field definition table (FDT), but non-overlapping ISN ranges. The data content of at least one field (the field value criterion) determines the physical file in which a data record is located.



field definition table (FDT)

A table that defines each file's record structure and content. There is one FDT for each database file. FDTs, stored in the Associator's fixed area, have three parts: the first is a list of the file's fields in physical record order, the second part is a "quick index" to the records in the first part, and the third part defines the files sub/superfields and sub-/super-/hyper- and phonetic descriptors.

full save

An ADASAV utility save operation that saves the complete current database; also, the resulting save dataset.

internal sequence number (ISN)

Every Adabas record is assigned an internal sequence number (ISN) to identify the record. Each record keeps its original ISN, regardless of where it is located.

Records in a physical database file have four-byte ISNs ranging from MINISN to MAXISN. In replicated files, a record has the same ISN in all file copies. In partitioned files, the ISN ranges are non-overlapping for each physical file.

merged delta save tape

The result of combining multiple consecutive delta save tapes into a single consolidated delta save tape.

merged full save tape

A save tape constructed by the Delta Save merging function, containing the complete database.

physical file

A physical file contains database records. Each physical file is identified by a file number. The number of physical files (and physical file numbers) per physical database is limited to 5000 or one less than the ASSOR1 block size, whichever is lower.

recover mode

The creation of an interim DSIM dataset from sequential protection log (PLOG) datasets. This mode is used when normal DSIM dataset creation fails.

record buffer

The portion of the calling program's parameter area, called the user buffer, that contains the data transferred during Adabas read, search, and update operations. When reading data field definitions, Adabas also returns the field definition information in the record buffer.

relative Adabas block number (RABN)

Adabas divides Data, Associator, and Work disk space into device-dependent logical blocks. The blocks in each of the three areas are numbered consecutively in ascending sequence beginning with RABN 1. The data blocks themselves as well as their addresses are referred to throughout Software AG publications as "RABNs". In other words, the sentence, "Adabas assigns RABNs 1–10 to the Associator" means ten Adabas storage blocks numbered 1–10 are assigned—not just the block numbers, whereas "Adabas assigns 50 RABNs to the Associator" means 50 blocks of storage with unspecified RABN numbers is assigned.

service

A processor of Adabas calls and issuer of replies. An Adabas nucleus is an example of a service (see also target).

target

A receiver of Adabas calls. A target maintains a command queue, and communicates with routers using ADAMPM. A target is also classified as a service. The Adabas nucleus is a target.

user

A batch or online application program that generates Adabas calls and uses an Adalink for communication.

INDEX

A

Adabas Online System, Delta Save Facility functions, 45
ADAFRM utility, DSIM dataset format/reset functions, for Delta Save Facility, 63
Adalink, definition, 167
ADAPRI utility, DSIMPRI function, for Delta Save Facility, 73
ADAREP utility, description, Delta Save Facility, 79
ADARES
 utility, for Delta Save Facility, 87
 utility functions for DSF
 COPY (copy sequential log file), 87
 PLCOPY (copy PLOG log files), 92
ADARUN, parameters
 Delta Save Facility, 40
 DSF (Delta Save Facility control), 38
ADARUN TT parameter, 130, 133
ADASAV, utility functions for DSF
 MERGE, 103
 RESTORE DELTA, 111
 SAVE, 127
 SAVE DELTA, 131
ADAULD utility, 153–165
 BS2000 JCL, 162
 input processing, 158
 LOAD FILE function, DDISN parameter, 155
 OS/390 or z/OS JCL, 163
 output processing, 160
 user exit 9 processing, 161
 VM/ESA or z/VM JCL, 164
 VSE JCS, 165
Address Converter (AC), definition, 167
Address space, definition, 168
AOS (Adabas Online System), overview (figure), 45

Audience, intended, for this manual, 1

B

BS2000
 examples, Delta Save (DSF) utility functions
 ADAFRM, 67
 ADAPRI, 75
 ADAREP, 82
 ADARES, 97
 ADASAV, 137
 JCL requirements, Delta Save (DSF) utility functions
 ADAFRM, 67
 ADAPRI, 75
 ADAREP, 82
 ADARES, 96
 ADASAV, 136
 rewind options, changing, 34

C

Checkpoints, DSF-related nucleus/utility, 81
Content, Of this manual, overview of the, 1
COPY function, ADARES utility, for Delta Save Facility, 87

D

Database, administrator, definition, 168
Databases
 partially saving a, ADASAV SAVE DELTA function, 131
 restoring from partial save input, ADASAV RESTORE DELTA, 111
 saving complete, ADASAV SAVE function, 127

Adabas Delta Save Facility Manual

- Datasets, DSF images (DSIM), allocating space for, 57
- Delta Save Facility, removing, 54
- Delta Save identifier (DSID), description, 168
- Delta Save logging
 - calculating the DLOG area size, 55
 - changing the size of the DLOG area, 53
 - disabling, 52
 - enabling, 51
 - estimating size of current save, 51
 - installing, 47
- Delta Save logging (DLOG) area, description, 169
- Delta Save operation
 - estimating the size of, 51
 - restarting after interrupt, 134
- DLOG, avoiding area overflow, 40
- DLOG area, defining and calculating, 47
- DLOG logging area, for Delta Save, changing the size, 53
- DSF (Delta Save Facility)
 - ADARUN parameter, 38
 - ADARUN parameters, 40
- DSFEX1=user-exit, ADARUN parameter, 40
- DSID, Delta Save identifier, 13
- DSIM dataset
 - definition, 169
 - description, 8
 - for Delta Save, calculating the size, 57
 - rebuilding, 95
 - recover mode, definition, 170
- DSIMFRM function, formatting the Delta Save DSIM dataset, ADAFRM utility, 63
- DSIMPRI function, printing the Delta Save DSIM dataset, ADAPRI utility function, 73
- DSIMRESET function, resetting the Delta Save DSIM dataset, ADAFRM utility, 65

E

Examples

- ADAPRI printing functions, 74
- ADARUN parameter, 38–39
- BS2000, Delta Save (DSF) utility functions
 - ADAFRM, 67
 - ADAPRI, 75
 - ADAREP, 82
 - ADARES, 97
 - ADASAV, 137
- calculating DSIM dataset size, for Delta Save Facility, 58
- COPY function of ADARES, 91
- DSID assignment, 13–24
- MERGE function of ADASAV, 110
- OS/390 or z/OS, Delta Save (DSF) utility functions
 - ADAFRM, 69
 - ADAPRI, 76
 - ADAREP, 83
 - ADARES, 98
 - ADASAV, 141
- PLCOPY function of ADARES, 94
- RESTORE database function of ADASAV, 125
- SAVE database function of ADASAV, 130, 133
- VM/ESA or z/VM, Delta Save (DSF) utility functions
 - ADAFRM, 70
 - ADAPRI, 77
 - ADAREP, 84
 - ADARES, 100
 - ADASAV, 145
- VSE, Delta Save (DSF) utility functions
 - ADAFRM, 71

ADAPRI, 78
 ADAREP, 85
 ADARES, 101
 ADASAV, 150
 Exits, DSF, definition, 168

F

FDT (field definition table), definition, 170
 Field, definition table (FDT), definition, 170
 Files
 partitioned logical, definition, 169
 physical, definition, 170
 sequential, determining names and blocks
 sizes, 31
 Formatting the DSIM dataset, for Delta Save
 Facility, ADAFRM utility, 63
 Full save tape, definition, 170
 Functions, utility, Delta Save Facility (DSF), 59

G

Glossary, Delta Save Facility, 167

I

installation instructions, 27
 Introduction, to this manual, 1
 ISNs
 definition, 170
 file of unloaded, 155

J

JCL requirements
 BS2000, Delta Save (DSF) utility functions
 ADAFRM, 67
 ADAPRI, 75

ADAREP, 82
 ADARES, 96
 ADASAV, 136
 OS/390 or z/OS, Delta Save (DSF) utility
 functions
 ADAFRM, 69
 ADAPRI, 76
 ADAREP, 83
 ADARES, 98
 ADASAV, 141
 VM/ESA or z/VM, Delta Save (DSF) utility
 functions
 ADAFRM, 70
 ADAPRI, 77
 ADAREP, 84
 ADARES, 99
 ADASAV, 145
 JCS requirements, Delta Save (DSF) utility
 functions
 ADAFRM, 71
 ADAPRI, 78
 ADAREP, 85
 ADARES, 101
 ADASAV, 149
 Jobs, automatic submission, 40

L

Logging
 DLOG area, allocating space for, 47
 DSF area (DLOG), changing the size, 53
 Logical, units, changing VSE, 34

M

Manuals, related, 2
 MERGE function, ADASAV utility, for Delta
 Save Facility, 103
 Merging delta save output, MERGE function,
 103

N

- Nucleus
 - checkpoints, 81
 - single SAVE operation restriction if active, 127

O

- Options, tape rewind, changing BS2000, 34
- OS/390 or z/OS
 - examples, Delta Save (DSF) utility functions
 - ADAFRM, 69
 - ADAPRI, 76
 - ADAREP, 83
 - ADARES, 98
 - ADASAV, 141
 - JCL requirements, Delta Save (DSF) utility functions
 - ADAFRM, 69
 - ADAPRI, 76
 - ADAREP, 83
 - ADARES, 98
 - ADASAV, 141

P

- parameters, ADARUN, DSF (Delta Save Facility), 38
- Partitioned files, definition, 169
- Physical, file, definition, 170
- PLCOPY function, ADARES utility, for Delta Save Facility, 92
- Printing database information, for DSIM (Delta Save) datasets, ADAPRI utility, 73

R

- RABN (relative address block number), definition, 171
- rebuilding, the DSIM dataset, 95
- Record buffer, definition, 171
- Recover mode, DSIM dataset, definition, 170
- Reporting database information, for Delta Save, ADAREP utility, 79
- Requirements and restrictions
 - DSF partial save operations after ADAFRM changes, 60
 - for NONUC parameter when copying current PLOG, 89
- restart, after interrupted delta save operation, 134
- RESTORE function, ADASAV utility, for Delta Save Facility, 111
- Restoring delta save output, RESTORE function, 111

S

- SAVE DELTA partial save function, ADASAV utility, for Delta Save Facility, 131
- SAVE function, ADASAV utility, for Delta Save Facility, 127
- SAVE function of ADASAV, active nucleus restriction to one operation, 127
- Save tape, unload, using utility, 153
- Save tape output, definition, 170
- Saving the database
 - partial, SAVE DELTA function, with Delta Save Facility, 131
 - SAVE function, for Delta Save Facility, 127
- Service, definition, 171
- Shadow database, 23
 - maintaining, 114
- Space
 - allocating Delta Save Facility, for DSIM dataset, 57
 - allocating DLOG area, 47

changing Delta Save Facility logging area, 53
 Status, DSF, description, 169

T

Tapes

merged full save, definition, 170
 output, save, definition, 170
 Target, definition, 171
 TTSYN parameter, 127, 130, 131, 133

U

User, definition, 171
 User exit
 automatic job submission, 40
 DSF, 44
 to avoid DLOG area overflow, 40
 User exits, 9, used with ADAULD utility, 161
 Utility
 checkpoints, Delta Save Facility-originated,
 81
 Delta Save Facility functions
 ADAFRM, 63
 ADAPRI, 73
 ADAREP, 79
 ADARES, 87
 ADASAV, 87, 103
 general information, 59

V

VM/ESA or z/VM

examples, Delta Save (DSF) utility functions
 ADAFRM, 70
 ADAPRI, 77
 ADAREP, 84
 ADARES, 100
 ADASAV, 145
 JCL requirements, Delta Save (DSF) utility
 functions
 ADAFRM, 70
 ADAPRI, 77
 ADAREP, 84
 ADARES, 99
 ADASAV, 145

VSE

examples, Delta Save (DSF) utility functions
 ADAFRM, 71
 ADAPRI, 78
 ADAREP, 85
 ADARES, 101
 ADASAV, 150
 JCS requirements, Delta Save (DSF) utility
 functions
 ADAFRM, 71
 ADAPRI, 78
 ADAREP, 85
 ADARES, 101
 ADASAV, 149
 logical units, changing, 34
 VSE/ESA, job control for DSF, 61

Z

ZAPs

for changing BS2000 rewind options, 34
 for changing VSE logical units, 34

