

Functional Overview

The ADARES utility performs functions related to database recovery.

Information about using ADARES

1. The functions BACKOUT (except BACKOUT DPLOG or MPLOG), REGENERATE, and REPAIR require a sequential dataset containing protection log data as input. A dual/multiple protection log cannot be used directly. To convert dual/multiple protection logs to sequential logs, use the ADARES PLCOPY function.
2. The REGENERATE, BACKOUT, and COPY functions can process several sessions within one job if the following is true:
 - The DDSIIN/SIIN input file contains the sessions in ascending order by session number; gaps in the session number sequence exist only for those sessions representing save dataset generations;
 - Each session on the file begins with block number 1, and there are no gaps in block numbering.
3. To select a *single* session only, specify the session number with PLOGNUM, or with FROMPLOG without TOPLOG; to specify a *range* of sessions, specify FROMPLOG and TOPLOG.
4. The ADARES COPY function accepts ADASAV output save (DD/SAVEn) files. No parameters indicating a save file can be specified; ADARES recognizes a save file by its structure. Only one save file can be copied during an ADARES COPY run. When copying a save file, specify the session number with PLOGNUM.
5. Adabas expanded files: the BACKOUT and REGENERATE (file) functions process complete expanded files only. If the file specified is either the anchor or component file of an expanded file, all other component files of the expanded file must also be specified.
6. To perform the additional Delta Save Facility operations of ADARES, ADARUN parameter DSF=YES must be specified in the DD/CARD input.
7. Multithreaded BACKOUT, BACKOUT DPLOG or MPLOG, and REGENERATE require additional main memory, which can be estimated using the values for the corresponding nucleus ADARUN parameter NT and NU:

$$(NT \times 70,000) + (NU \times 72)$$

For example, if NT=28 and NU=1000, about 2MB of main memory is required.

8. For optimal processing when using the multithreaded backout/regenerate feature, Software AG recommends that you set the nucleus ADARUN parameter NAB to at least

$$NAB=NT \times (32K + 108) / 4096$$

Using ADARES in Adabas Nucleus Cluster Environments

In an Adabas nucleus cluster environment, the protection logs (and optionally, the command logs) of all individual nuclei in the cluster are merged into single log files in chronological order for the cluster database shared by all the nuclei as a whole. The chronological order is determined by timestamps on all individual nucleus log records, which are synchronized across the cluster by the operating system.

Merging Logs

For recovery processing, all protection log datasets (PLOGs) must be merged into a single log "stream" for each cluster database. PLOGs are merged automatically when an ADARES PLCOPY is executed. The PLCOPY process accesses the parallel participant table (PPT) to determine which PLOGs to copy and uses dynamic allocation to access the appropriate datasets.

An existing PLCOPY job must be modified to run in a cluster environment. The user exit 2 may also need to be modified. A sample PLCOPY job ADARESMP that illustrates the necessary addition of the intermediate datasets and a sample user exit 2 (USEREX2P) is provided. See Automatically Copy/Merge Nucleus Cluster Protection Logs. It is not necessary to remove the PLOG DD statements, however. If they remain, they are ignored.

By default, dual/multiple command log datasets (CLOGs) can be copied to a sequential dataset for each nucleus using the ADARES CLCOPY function, but the resulting datasets are not then automatically merged across the cluster into a single CLOG dataset for the cluster database. You can choose to merge the CLCOPY output from each nucleus manually by using the ADARES MERGE CLOG function. By default, the CLOG datasets must be specified in the user exit 2 JCL; they are not dynamically allocated.

However, for accounting or other tracking purposes, you may want to automate the CLOG merge process the same way the PLOG merge process is automated. When you specify ADARUN CLOGMRG=YES, the CLOG merge process is invoked automatically when the ADARES CLCOPY job is submitted from UEX2 and executed. ADARUN LOGGING=YES must also be specified. As with the PLCOPY process, the CLCOPY process then accesses the parallel participant table (PPT) to determine which CLOGs to copy and uses dynamic allocation to access the appropriate datasets.

Existing CLCOPY jobs must be modified to include the intermediate datasets. A sample CLCOPY job ADARESMC is provided that illustrates the necessary addition of the intermediate datasets. See Automatically Copy/Merge Nucleus Cluster Command Logs. The sample user exit 2 (USEREX2P) includes both CLCOPY and PLCOPY functionality for the merge.

The automated PLCOPY and CLCOPY jobs copy/merge as much data as possible; if a nucleus is still writing to a log dataset, the job 'partially' merges the dataset.

Intermediate Datasets

The merge begins with the lowest timestamp from all PLOGs and CLOGs being merged and ends with the lowest of the ending timestamps from all datasets. Records beyond this point are written to an 'intermediate' dataset, which must be supplied as input to the subsequent merge. A cross-check ensures that the correct intermediate dataset has been supplied.

ADARES expects that at least one of the PLOGs or CLOGs being merged is at 'completed' status. If this is not the case, ADARES reports that there is no data to be copied.

A sample user exit 2 (USEREX2P for both PLOGs and CLOGs) is provided that illustrates the necessary JCL for the intermediate datasets. When intermediate datasets are used for both CLCOPY and PLCOPY jobs, the dataset names for each must be unique so that they are not overwritten.

- PLCOPY example:

```
//MARGIN1 DD DISP=SHR,DSN=EXAMPLE.PINTERI
//MARGIN2 DD DISP=SHR,DSN=EXAMPLE.PINTERO
```

- CLCOPY example:

```
//MARGIN1 DD DISP=SHR,DSN=EXAMPLE.CINTERI
//MARGIN2 DD DISP=SHR,DSN=EXAMPLE.CINTERO
```

Depending on whether it is a PLCOPY or a CLCOPY, the job submitted by user exit 2 must refer to the appropriate set of statements.

Once DD statements for the PLOG datasets have been supplied on the session start-up JCL, you do not need to supply them again for ADARES as these are opened using dynamic allocation. If the DD statements are supplied, they are ignored.

It is not necessary to manually change the JCL after each execution. ADARES maintains control information in the parallel participant table (PPT) to determine which intermediate dataset to expect as input. It checks the header information in both datasets to determine which to use for input and which for output.

The following checks are made to ensure that the intermediate dataset has been supplied correctly:

1. The DBID is stored in the intermediate dataset header and must match the DBID in the log.
2. The log number is stored in the intermediate dataset header and must either match or be one less than the current number from the log dataset.
3. The STCK in the intermediate dataset header must match the STCK stored in the PPT.

If any of the checks fails, ADARES ERROR 157 is returned.

ADARES also ensures that the intermediate dataset contains the number of records expected. If not, ADARES ERROR 164 is returned.

Uniquely Identifying Checkpoints

After the protection log (PLOG) merge process, the block number will not necessarily be the same. To uniquely identify the checkpoint in this situation, it is necessary to also specify the NUCID for all functions that can specify a TOBLK/ FROMBLK parameter; that is, BACKOUT and REGENERATE.

The merge process ensures that there is at most one checkpoint per block. It records the (old) block number prior to the merge and the NUCID that wrote the checkpoint. When you then specify the block number and NUCID as reported in ADAREP, ADARES is able to uniquely identify the block.

Note:

In an Adabas nucleus cluster environment, ADAREP includes the NUCID when printing all checkpoint information.

The additional parameters that are required in an Adabas nucleus cluster environment are NUCID, TONUCID, and FROMNUCID. If the NUCID is the same for the starting and ending checkpoint, only the NUCID needs to be specified.

Note:

ADASAV stores this information in the header so that it can uniquely identify the block for the RESTONL and RESTPLOG functions.