Adabas Restart and Recovery

A "user session" is a sequence of Adabas calls optionally starting with an OP command and ending with a CL command. A "user" is either a batch mode program or a person using a terminal. The user can be identified by an 8-byte unique ID provided with the OP command. This ID enables Adabas to retain restart information (ET data) beyond the end of a session.

For communication purposes, a terminal user is identified by machine, address space, and terminal ID, thereby ensuring that each user ID is unique.

During an Adabas "session" (from activation to termination), the Adabas nucleus creates a sequence of protection entries in exact historical sequence reflecting all modifications made in the database. The sequence of protection entries is written to the Work dataset (part 1) and to a protection log in the form of blocks. Each block contains the nucleus session number, a unique block number, and a time stamp.

This chapter covers the following topics:

- Work Dataset
- Protection Log
- Restart Operations
- Database Recovery

Work Dataset

Part 1 of the Work dataset (ADARUN LP parameter) stores the most recent protection entries using a wrap-around method.

The protection entries on the Work dataset are used to execute a BT command and to execute autorestart/autobackout when reactivating the Adabas nucleus after a system failure. The entries may contain all or some of the following:

- before/after images of a data record;
- before/after images of elements of the inverted lists (DVT);
- special before-images of inverted list blocks for automatically repairing the database after a system failure;
- checkpoint entries;
- ET entries including ET data;
- special entries for handling the internal restart procedures.

Adabas identifies a batch user by checking a store clock (STCK) value during the program’s first Adabas call.

Protection Log

The protection log contains the same entries as part 1 of the Work dataset (except the special before-images mentioned in the discussion of the Work dataset. Additional entries on the protection log which are not stored on the Work dataset include an entry for the infrequently used C5 data and an after-image Associator or Data Storage block written during buffer flush; the latter occurs while an online SAVE function of the ADASAV utility is running.

All protection log entries created by the nucleus describe the modifications made to the database in exact historical order. Each block is associated with a block sequence number.

The protection log can be written either

- directly to a sequential protection log (DD/SIBA) dataset; or
- to multiple protection log (DD/PLOGR1, DD/PLOGR2, ... DD/PLOGR8) datasets.

Note:
Adabas version 7.4 still supports dual protection logging using DUALPLD/S and user exit 2.

Multiple protection log datasets are each physical datasets of the same size and block length, randomly accessed, and used in succession. This means that one of the datasets can be used and written by the nucleus while others are being copied in order for archive purposes.

- Sequential Protection Log
- Multiple Dataset Protection Log

Sequential Protection Log

The sequential protection log dataset is opened by the nucleus when a session is started and closed when a session is ended. In general, this dataset is assigned to tape to avoid disk space problems that may cause an unexpected abnormal termination. Because the amount of data to be written to such a dataset depends upon the amount of activity of all users, estimating disk space is difficult.

At the end of the Adabas session, an end-of-file mark (EOF) is written to the tape to indicate end of session. Adabas supports multivolume protection log datasets. If one reel is not enough to store all protection log entries, subsequent tape reels can be used.

The nucleus writes a checkpoint for each volume written. This checkpoint contains information about session number, volume serial number, and block sequence numbers.

The ADARES COPY function must be used to copy such a sequential protection log dataset for archive purposes.

If a nucleus session ended abnormally, the final EOF mark on the tape may not be written. This may cause problems if the protection log is used directly as input to the ADARES BACKOUT or REGENERATE function. The COPY function of ADARES is able to detect the logical end of the in/out protection log and writes a valid EOF mark to the output. ADARES writes checkpoints in the same way the nucleus does for each output volume.

Note:
When using a sequential protection log on tape, one tape unit is allocated for the Adabas nucleus for the duration of the entire nucleus session. This session can last a very long time, during which the log tape unit must be available; a single sequential logging tape is therefore not adequate in every case. Software AG therefore recommends using multiple protection logging.

Multiple Dataset Protection Log

The Adabas multiple dataset protection log consists of two to eight datasets (DD/PLOGRn where "n" is the sequential number of the dataset) with the following attributes:

- fixed block size;
- reside on DASD;
- preformatted by ADAFRM;
- all datasets have the same number of blocks and identical block sizes;
- all datasets can be shared by the nucleus and other utilities (ADARES).

Assuming newly formatted dual or multiple protection log datasets, Adabas selects DD/PLOGR1 at startup and starts writing protection log entries to it. Writing starts at block 2. Block 1 contains status information about the dataset. Other PLOG datasets are still unused. Protection log entries are written to multiple protection log datasets in the same order they are written to a sequential log.

Each protection log dataset need not be large enough to accommodate all protection log entries for a session. When one dataset becomes full, "protection log switching" occurs as follows:

1. status information is written to block 1 to terminate the current protection log dataset;
2. there is a switch to another dataset;
3. a message is written to the operator and to the log output; and
4. user exit 12 is called (see below).

While the nucleus continues writing protection log entries to the other dataset, the first one is copied to a sequential dataset by ADARES PLCOPY. ADARES can be started manually or initiated by user exit 12, which is called whenever a switch from one protection log dataset to another occurs. ADARES writes a checkpoint for each output volume written. This checkpoint contains the session number, volume serial number, and block sequence number.

A protection log switch may occur more than once in a single session. The content of each protection log dataset must be copied to a single sequential dataset. All subsequent copies produced within one session are logically equivalent to the information the nucleus would have written to a sequential protection log (DD/SIBA).

All sequential copies can be concatenated to form a single sequential dataset containing all protection log entries for a session. In fact, a sequential copy is required as input by the BACKOUT/REGENERATE functions of ADARES.

Note:
A tape unit to store the sequential protection log entries is required only during the ADARES PLCOPY run.

If multiple dataset protection logging is used, but user exit 12 is not available to call ADARES PLCOPY, protection log switching occurs as follows:

1. the current protection log dataset is closed; and
2. if no other dataset is empty, the following message is issued and the old data is overwritten:

   ```
   Now it’s too late to copy DDPLOGRn (or PLOGRn)
   ```

   In this case, protection log information is lost.

Restart Operations

Protection entries are needed if any of the following fail:

- a user application program
- Adabas
- the operating system
- the hardware

Restart after a User Application Program Failure

An application program that is in the middle of a transaction can detect that the transaction cannot be completed successfully. Removing the first portion of the transaction, called back out or roll back, is performed by the BT command.

The BT command is executed by reading the Work dataset backwards and executing the entries for the specific transaction in reverse (after-image is used to scratch an element in the database, before-image is used to insert an element in the database). The start-transaction bit in an element serves as the stop indicator for the BT process.

Restart after an Adabas, Operating System, or Hardware Failure

When Adabas is reactivated after any failure that caused the Adabas nucleus to terminate abnormally (that is, failure of Adabas, the operating system, or hardware), an automatic procedure is executed to bring the database to a physically and logically valid status. All partially executed update commands are reset. All incomplete transactions are backed out.

This automatic procedure comprises three steps:

1. repair the database
2. autorestart

3. autobackout

The repair is needed to modify the database to the status it would have if a buffer flush had just been completed at the time of the failure. In other words, all blocks in the database are at a status that enables the nucleus to perform normally by addressing Data Storage records through the address converter and normal index entries through the upper index.

"Autorestart" backs out updates of single update commands that were partially executed when the system failed; "Autobackout" backs out updates of user transactions that were partially executed when the system failed.

The major protection entries used for autorestart and autobackout are the before-images and after-images of Data Storage and the inverted lists (DVT).

Restart after a Power Failure

Depending on the hardware, a power failure during an I/O operation may damage the Adabas blocks that were being processed. This damage cannot be detected during autorestart and therefore can result in problems later, such as unexpected response codes of lost database updates.

Note:
If the cause of the abend was a power failure, Software AG strongly recommends recovering the affected files using the ADASAV and ADARES utilities as described in the section *Database Recovery* .

Whenever an Adabas session is reactivated with the IGNDIB=YES parameter, which forces the new session to ignore an existing session communication block (DIB) in the Associator, Adabas checks whether a buffer flush was active when the abend occurred. If a buffer flush was in process, the autorestart shuts down and issues an ADAN58 message:

ADAN58 BUFFER-FLUSH START RECORD DETECTED DURING AUTORESTART. THE NUCLEUS WILL TERMINATE AFTER AUTORESTART. IN CASE OF POWER FAILURE, THE DATABASE MIGHT BE INCONSISTENT...

The message also includes a list of the files that were being updated when the buffer flush was in process. In this case, the DBA must check whether the cause of the abend was a power failure.

If the abend was *definitely not* a power failure and the integrity of the information on the output hardware can be guaranteed, the database can be reactivated immediately. Database recovery is not necessary.

Using Automatic Restart Management (ARM)

Automatic restart management (ARM) is used to automatically restart a nucleus when it abends. Automatic restart is suppressed when the abend is intentional; for example, when it results from a parameter error.

ARM can be used for Adabas nuclei in both cluster and non-cluster environments.

The ADARUN parameter ARMNAME is used to identify the element in the ARM ’policy’ that is to be activated. Each element specifies when, where, and how often an automatic restart is to be attempted. If an ARM policy has not been defined, the ARMNAME parameter has no effect.

Database Recovery

If an Adabas, operating system, or hardware failure occurs and the physical database is still readable (which is the normal case), the Adabas nucleus automatically takes all necessary steps to ensure that database processing can be continued in a normal manner.

Database recovery is described under the following headings:

- Recreating a Database
- Database or File Recovery Considerations
- Database Recovery Guidelines
- Using the Adabas Recovery Aid (ADARAI)

Recreating a Database

In case of a head crash (physical damage to the database), application program error (logical damage to the database), or a power failure during a buffer flush (described in the section *Restart after a Power Failure*, the utilities ADASAV and ADARES must be used to recreate the database.

To restore and regenerate the entire database

1. Restore the database with ADASAV using the sequential dataset containing the most recent copy of the database.
2. Restore the database blocks that were updated during the SAVE of the database (online SAVE).
3. Regenerate with ADARES from the checkpoint taken at the end of the SAVE function to the latest point at which the database was still intact (this is done automatically by Adabas).

To restore and regenerate a single file of the database

- Use the steps described for restoring and regenerating an entire database, but make the appropriate changes in the parameter statements of the utilities.

To restore and regenerate single blocks of Data Storage

- Use the REPAIR function of ADARES.

To restore the status of a database at the start of a single batch update run

1. If a long-running batch program performs a large number of erroneous updates (e.g., logic error in program), and the program was the only user performing updates, it may be desirable to back out all updates performed by the program.
2. Use the ADARES BACKOUT function with the sequential, dula, or multiple protection log.

Database or File Recovery Considerations

The information written on the protection log (SIBA/PLOG) contains record-based information, including the record identifiers (ISNs). The ADARES utility passes the ISN-based information to Adabas; however, Adabas can neither verify the validity of the ISN nor check that the logical content of the record is correct.

Therefore, you must ensure that the combinations of ISN and logical record used in the original session are also used during the procedure for recreating the file or database. Use the ISNs and parameters that were specified when the file was originally loaded.

Database Recovery Guidelines

To ensure that the database can be recovered in the event of a software or hardware failure, Software AG recommends that the DBA

- create a sequential dataset copy of the database using the ADASAV utility (SAVE function) and archive the output dataset (normally on tape). This dataset reflects the status of the database at a specific time.
- retain all protection log data written to the protection log dataset for each nucleus session.

If you are using sequential protection logging (SIBA), you can use the ADARES COPY function to archive the log information. If you are using multiple dataset protection logging (PLOG), use the ADARES PLCOPY function.

Each nucleus session is identified by a unique session number. This number is assigned to all SAVE and protection log data.

Example 1: Inactive Nucleus Save Operation

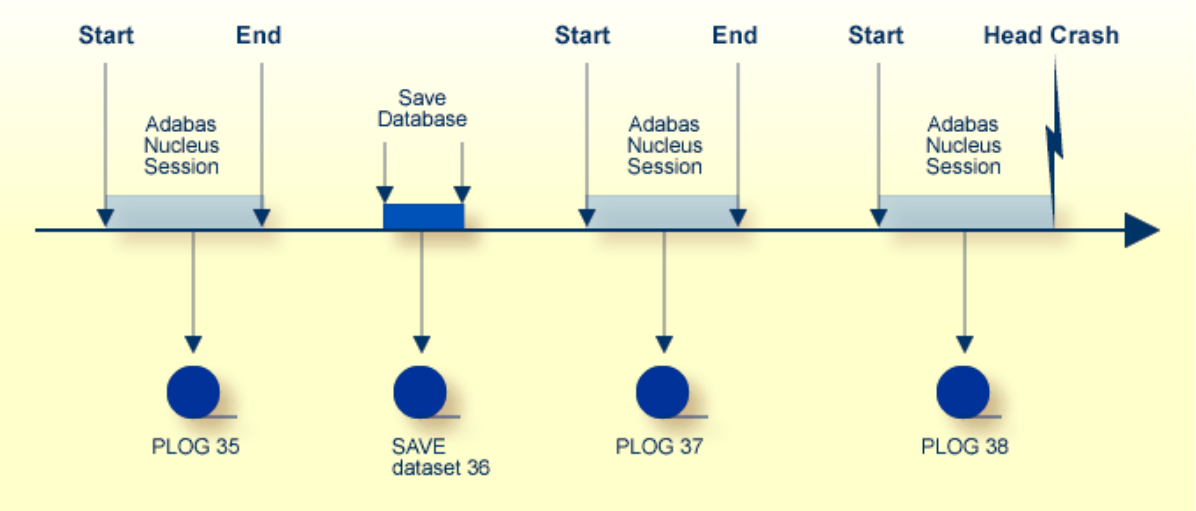
To recreate the database to the status of the failure

1. Restore the database using SAVE dataset 36.

   With the nucleus *not* active, execute

   ```
   ADASAV RESTORE
   ```

2. Start the Adabas nucleus.



3. Reapply all modifications made in session 37.

   With the nucleus active and running, execute

   ```
   ADARES REGENERATE PLOGNUM=37
   ```

4. Reapply all modifications made in session 38.

   With the nucleus active and running, execute

   ```
   ADARES REGENERATE PLOGNUM=38
   ```

Example 2: Active Nucleus Save Operation

This example shows the assignment of session numbers to SAVE and protection log datasets when a SAVE database function is executed parallel to an active nucleus.

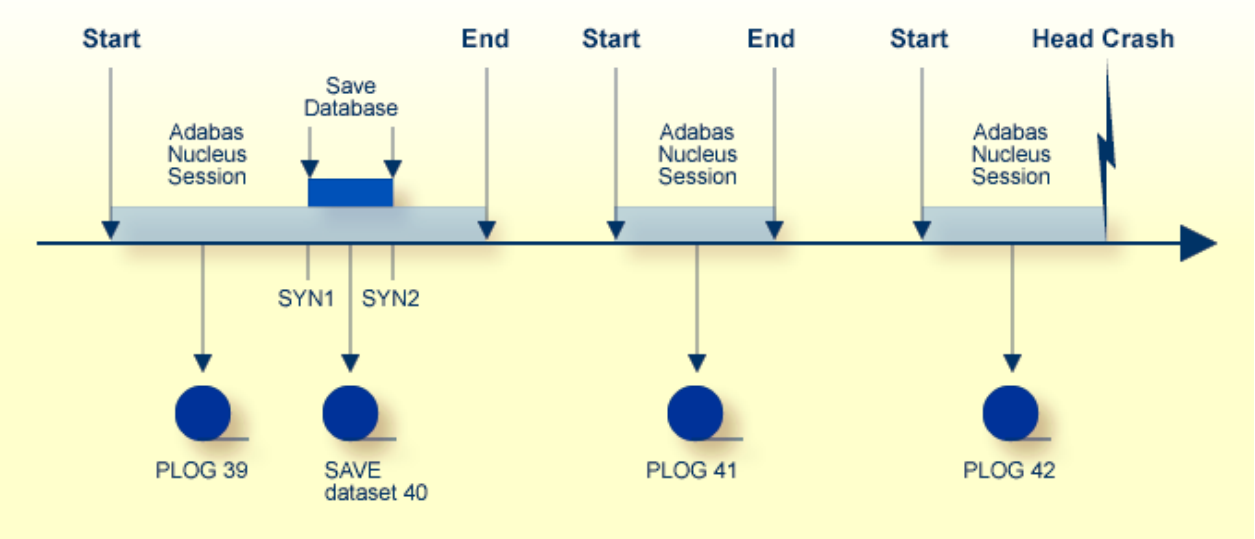
The nucleus writes a protection log number 39.

When ADASAV starts the SAVE function, the nucleus writes a SYN1 checkpoint to the data protection log.

At the end of the SAVE function, a SYN2 checkpoint is written. The SYN2 checkpoint is a synchronized checkpoint where all users are at ET status.

All writes of the nucleus to the Associator and Data Storage are written to the protection log as well (between SYN1 and SYN2).

The output of ADASAV is associated with session number 40. The nucleus still writes to protection log 39, even if the SYN2 checkpoint has been written.



To reset the database to the status of the failure

1. Restore the database using SAVE dataset 40 as input.

   Modifications made to the database during the online SAVE operation are found on protection log 39, starting at the SYN1 checkpoint. ADASAV also requires the correct protection log input.

2. Stop the nucleus; then execute

   ```
   ADASAV RESTONL PLOGNUM=39,SYN1=blocknumber
   ```

3. Start the Adabas nucleus.

4. Reapply modifications made in session 39, starting from the SYN2 checkpoint.

   With the nucleus active, execute

   ```
   ADARES REGENERATE PLOGNUM=39,FROMCP=SYN2,FROMBLK=blocknumber
   ```

5. Reapply modifications made in session 41.

   With the nucleus active, execute

   ```
   ADARES REGENERATE PLOGNUM=41
   ```

6. Reapply modifications made in session 42 up to the time of failure.

   With the nucleus active, execute

```
ADARES REGENERATE PLOGNUM=42
```

To find the SYN1 and SYN2 block numbers needed as input for ADASAV and ADARES, check the output report written by the ADASAV SAVE function or perform the "ET Checkpoint" function in Adabas Online System.

Using the Adabas Recovery Aid (ADARAI)

"Transaction" recovery is provided whenever an Adabas session is abnormally terminated. The Adabas autobackout routine, which is automatically invoked at the beginning of every Adabas session, removes the effects of all interrupted transactions from the database.

However, when a database dataset (ASSO, DATA, or WORK) is destroyed, it is necessary to restore and regenerate the database to recover the lost data.

The Adabas Recovery Aid helps automate and optimize "database" recovery. It records and reports all information needed to recover the database and builds the recovery job stream (JCL/JCS), which is the basis for reexecuting the jobs performed from the time of the last SAVE to the point of failure and error. For information, see the description of the ADARAI utility in the Adabas Utilities documentation.

Note:
The job stream generation function is not yet available under VSE/ESA or VM/ESA.