RECOVER: Build a Recovery Job Stream

Note:
The RECOVER function is currently available for BS2000 and z/OS systems only. Support for z/VSE systems is planned.

The ADARAI RECOVER function builds the job control information (recovery job stream) for recovering the Adabas database or selected database files. The RECOVER function

- reads the PLOG information to determine if a PLCOPY is needed; and
- reads the RLOG to build the recovery job stream from the skeleton job control.

ADARAI RECOVER builds the job stream necessary to restore the database or files to the condition before the RECOVER function was run. The completed job stream is sent to the DD/JCLOUT data set.

Where appropriate, ADARAI includes error or information messages in the generated job stream. You must then manually correct the errors before submitting the job. The existence of messages in the job stream is indicated by a nonzero return code from ADARAI RECOVER.

For BS2000 systems, RECOVER additionally

- performs, when generating the job control, the same checks performed by the LIST function for BS2000; and
- includes BS2000 /REMARK statements in the created job control for checks that produce errors.

Note:
When such errors occur, the job control must be corrected manually.

This chapter covers the following topics:

- Recovery Processing
- Optimized Recovery Processing
- Requirements
- Restrictions
- Input Needed for Recovery
- Output from the Recovery Operation
- Executing the RECOVER Function
- File-Level Recovery
- Syntax
- Optional Parameters and Subparameters

- Examples
- Skeleton Job Control
- User Exit to Change JCL
- Prerecovery Checking
- Restarting the RECOVER Function or Recovery Job Stream

Recovery Processing

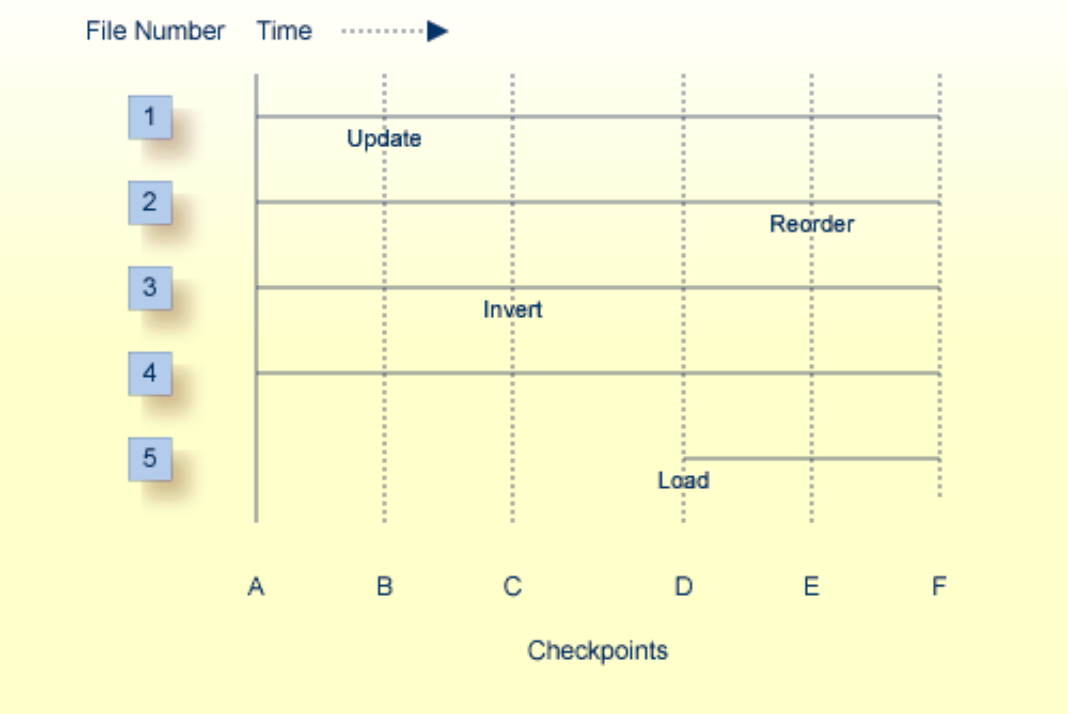
The ADARAI RECOVER function builds a job based on the exact sequence it finds in the generation to be recovered:

- it restores the database from the data sets created by the operation that started the generation;
- it regenerates PLOGs up to the next utility checkpoint found;
- it generates a job step to reexecute the utility and start the regeneration after that checkpoint.

This sequence continues until all utilities have been replayed and the last PLOG block in the generation has been regenerated.

The following diagram illustrates the functioning of ADARAI where

- a database is saved to start a new generation at A.
- the database runs and at various times during the generation
-
    - an update is run against file 1;
    - a reorder is run against file 2;
    - an invert is run against file 3; and
    - a load is run against file 5.



Given the above, the order of the recovery is as follows:

1. A full save or full save plus delta saves are restored to return the database to the status at A.
2. A database regenerate runs from the first checkpoint at A up to the update checkpoint at B. The regenerate job then terminates.
3. The update utility runs for file 1 and a database regenerate runs between the checkpoint at B and the invert checkpoint at C.
4. The invert utility runs for file 3 and a database regenerate runs between checkpoint C and the load checkpoint at D.
5. The load utility runs for file 5 and a database regenerate runs between checkpoint D and the reorder checkpoint at E.
6. The reorder runs for file 3 and a database regenerate runs between checkpoint E and the most up-to-date level of the database at F.

Optimized Recovery Processing

The OPT parameter for the RECOVER function of ADARAI is used to identify operations or sequences that would minimize the time required to recover a large database.

For example, it a file with 10,000 updates is deleted or reloaded, it should be possible to avoid restoring the file from the start, replaying the 10,000 updates, and then throwing it all away when the delete or load operation occurs.

When optimization is selected, ADARAI does not restore the example file in the main restore for the job. Regeneration for the file occurs only after the file has been deleted or created by the load:

- a deleted file has no more updates;
- for a file created by a load, only updates subsequent to the load are important.

When an optimized job stream is used, the recovered database is rebuilt in a way that is different from the original build. Because optimized recovery jobs do not replay in exactly the same way as the original jobs, problems may occur in the recovery; for example, insufficient space may be available on the database. In most cases, however, the risk is minimal compared with the potential benefits of optimizing the database recovery. Each situation must be examined for potential problems.

Requirements

To generate a recovery job that will run successfully, ADARAI imposes the following conditions:

- the database must be run with dual or multiple protection logging active.
- sequential data sets input to utility functions that update files in the database must be retained.
- sequential output data sets created by SAVE or MERGE functions must be retained. This applies to SAVE FILE functions only if RESTFILE=YES is used for ADARAI RECOVER.
- retained data sets must keep their original names; ADARAI cannot track copies with different names.

Software AG recommends that retained data sets be cataloged.

Restrictions

Shadow Databases

If *shadow* databases, or copies of normal production databases, are built by restoring the delta save output and DSIM data set for a save of the original database, ADARAI has no knowledge of the PLOG activity that occurred during the delta save on the original database and therefore cannot rebuild the DSIM data set if a restore operation becomes necessary on the shadow database.

If, however, the DSIM data set and the delta save data set are merged to create a new *offline* delta save data set and the new merged data set is restored to the shadow database, ADARAI has all the information needed to recover the shadow database since the PLOG is not necessary in this case.

Restoring Delta Saves with a DSIM Data Set

In general, ADARAI handles RESTORE DELTA processing without problems. However, if the RESTORE DELTA uses a DSIM data set (which is essentially a *working* data set), the DSIM data set may not be intact if an ADARAI RECOVER becomes necessary. ADARAI therefore records the COPY or PLCOPY requests used to create a DSIM data set and emits a job step to rebuild the data set before attempting to replay such a RESTORE DELTA during RECOVER processing.

ADARAI searches the entire RLOG for appropriate entries. If the entries cannot be found, ADARAI cannot rebuild the DSIM data set prior to the RESTORE step and therefore cannot replay the RESTORE DELTA.

DD/FILEA File

In a generated recovery job, ADARAI writes the DD/FILEA file of the ADAORD utility. This cannot be avoided because the REORDER functions must be replayed and they require that the DD/FILEA file be written.

In this case, the following restrictions apply:

- ADAORD STORE processing simply reads the same DD/FILEA read when the utility was originally run as part of the generation being recovered.
- A temporary file (DISP=(NEW,DELETE), which is deleted after the step is executed) can be used for DD/FILEA because the recovery job creates and deletes the file again when it is executed.
- An existing file (DISP=OLD) can be used for DD/FILEA. If it still exists when the recovery job is run, ADARAI simply allocates the file with the disposition it had when the original job was run.
- If a new file (DISP=NEW,CATLG) is allocated for DD/FILEA and retained in the original ADAORD REORDER step, and if it still exists when the recovery job comes to the REORDER step (which is normal), ADARAI attempts to create the same file again, which causes the job to fail.
- If a GDG is used, the ADARAI recovery job sees only the name of the actual data set created by the generation. If the data set already exists (which is normal), ADARAI attempts to create the same file again, which causes the job to fail.

Input Needed for Recovery

The following data sets are input to the ADARAI RECOVER function:

- DD/RLOGR1, the recovery log.
- DD/PLOGR1 and DD/PLOGRn, the multiple protection logs, which are required when the ADARAI RECOVER parameter FEOFPL=YES (the default) is used.
- DD/JCLIN, which provides site-dependent skeleton job control statements. The RECOVER operation merges these statements with the RLOG information to create a complete database recovery job stream.

On BS2000 systems, DDJCLIN is a SAM data set with variable record format. EDT can be used to create and edit this data set. See the section Skeleton Job Control for more information.

On z/OS systems, the DDJCLIN data set must be defined with RECFM=FB, LRECL=80, and a BLKSIZE that is a multiple of 80 bytes.

Output from the Recovery Operation

The ADARAI RECOVER output is an execution-ready job stream for recovering the database. This recovery job stream is written to the DD/JCLOUT file. If a possible error condition is detected during the RECOVER operation, ADARAI issues a warning message and ends with a condition code of 4. See the section Prerecovery Checking.

On BS2000 systems, DDJCLOUT and DDJCLCON are SAM data sets with variable record format. They conform to the BS2000 job control conventions.

On z/OS systems, the DDJCLOUT DD statement must point to a data set defined with RECFM=FB, LRECL=80, and a BLKSIZE that is a multiple of 80 bytes.

The recovery job stream includes job steps to start the nucleus

- before the first regenerate job step; and
- after any utility operation that causes the nucleus to terminate automatically.

ADARAI RECOVER jobs replay all utilities with the database active, whether the utility was originally run in single-user mode or not. Utilities originally run in single-user mode are replayed in multiuser mode. These job steps are described in the sections Building the Recovery Job Stream and Skeleton Job Control.

Executing the RECOVER Function

The RECOVER function is run a generation at a time under control of the RELGEN parameter. If RELGEN is not specified, the default is the current generation.

RECOVER can be executed with the nucleus active or inactive. It can be executed more than once for the same generation because it does not change the RLOG information for that generation.

However, if RECOVER is rerun after a failure while running a DD/JCLOUT recovery job stream, the new recovery job stream produced may be different from the original recovery job stream. The reason is that the original recovery job stream may execute utilities against the database that updates the RLOG. The new RECOVER operation then builds a recovery job stream for the utilities that ran as part of the failed recovery job stream.

Also, if the recovery job stream failed after executing an ADASAV RESTORE, a new generation is created. In this case, execute RECOVER using the RELGEN=1 parameter setting to obtain the original generation.

Processing the PLOG

At the start of execution, if ADARAI RECOVER FEOFPL=YES, RECOVER reads the PLOGs, looking for information that must be copied. If necessary, it calls the nucleus to force a PLOG switch. If the nucleus is inactive, it invokes user exit 2.

Reading the Recovery Log

Next, RECOVER reads the skeleton job control into storage and reads the RLOG in chronological order, starting at the beginning of the generation specified by the RELGEN parameter. See Generation: The Unit of Recovery for a definition of generation.

If the entire database is being recovered, RECOVER uses the ADASAV SAVE or RESTORE information to create a new RESTORE/RESTONL database operation. For file-level recovery, it uses the SAVE/RESTORE database information to create a RESTORE FILE=... function.

Building the Recovery Job Stream

After creating a job stream for restoring the database or file, RECOVER creates a job step for starting the nucleus, using the %%JCL-STARTNUC statement.

RECOVER then creates the first regenerate job step. This job step does not contain a FROM checkpoint (FROMCP) unless an online SAVE (or DELTA SAVE) was the basis for starting the generation. In that case, the regenerate starts at the end checkpoint (SYN2) of the online save.

All PLOGs up to and including the next utility checkpoint (at which the REGENERATE must stop) are included and appropriate parameters are provided to the ADARES REGENERATE function. If more than 99 PLOGs are to be regenerated, ADARAI generates multiple REGENERATE job steps, each one processing up to 99 input PLOG data sets.

Once the PLOGs are regenerated up to the next utility execution, the utility job step is generated into the output recovery job. ADARAI then inserts another REGENERATE job step that includes all PLOGs up to and including the next utility checkpoint.

The recovery job continues inserting REGENERATE steps and utility steps until it detects the end of the generation specified by the RELGEN parameter. At this point, the completed job stream is sent to the DD/JCLOUT file.

File-Level Recovery

Recovery can be made on a file level by specifying the RECOVER function's FILE parameter. The file-level recovery process is essentially the same as the database-level recovery process, but is restricted to the files specified using the FILE parameter.

ADARAI produces a file-specific result in DD/JCLOUT by adding parameters to utility execution statements. For example, assume that the following statement was in the original ADASAV RESTORE statement:

```
ADASAV RESTORE FMOVE=2,3,NIRABN=100,1000,DSSIZE=550B,20
```

In this case, RECOVER FILE=3 produces the following DD/JCLOUT statement:

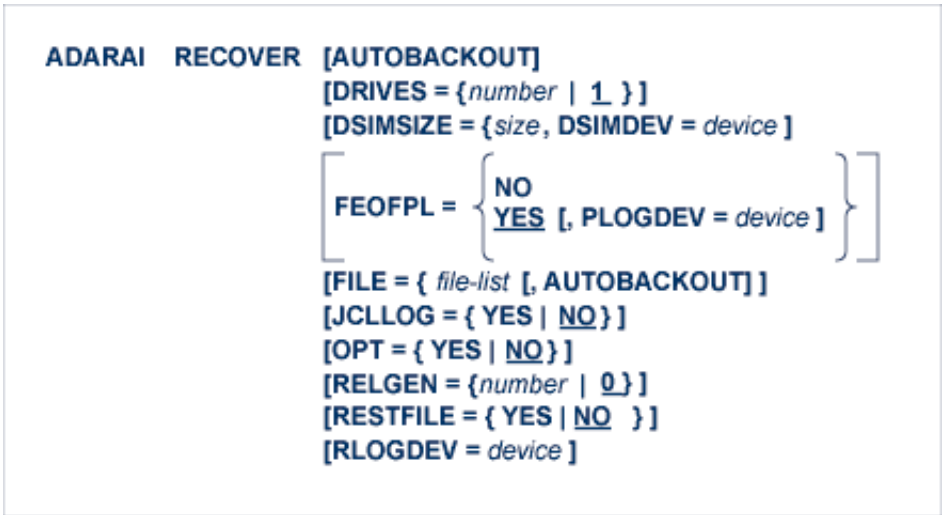
```
ADASAV RESTORE FMOVE=2,3,NIRABN=100,1000,DSSIZE=550B,20
ADASAV EXCLUDE=2
```

Note:
If a file to be recovered is part of an expanded file chain or is coupled, all files in the chain or the coupled list must be recovered together. If all coupled files or expanded file chains are not recovered together, ADARAI detects this and the ADARAI RECOVER function fails.

The Adabas nucleus *must* be active before executing a file-level recovery job. This is different from the database-level recovery job, which starts the database itself.

A file-level RECOVER operation does not create job control for utilities that were executed on the whole database (for example, ADADEF NEWWORK). The exceptions to this are utilities that can be reexecuted for individual files as well as the complete database. An example is ADASAV RESTORE (database), which provides a DD/SAVE input data set that can be used to create ADASAV RESTORE FILE=... job control.

Syntax



Optional Parameters and Subparameters

AUTOBACKOUT: Back Out Uncompleted Transactions

AUTOBACKOUT may only be specified for file-level recovery.

If AUTOBACKOUT is specified, transactions that were not complete at the end of the last REGENERATE function in the recovery job are backed out. Only completed transactions are left on the database.

If AUTOBACKOUT is *not* specified, incomplete transactions are left on the database.

For database-level recovery, incomplete transactions at the end of the last REGENERATE function are always backed out.

DRIVES: ADASAV Restore Input Drive Volumes

DRIVES is the number of input data sets to be used as input to the RESTORE step of the recovery job being generated.

The specified DRIVES parameter must be equal to or less than the DRIVES parameter on the job that started the generation. For example, if the generation was started with a database save with DRIVE=4, the RECOVER DRIVES parameter may only be specified as 1, 2, 3, or 4.

When you specify a lower number of DRIVES for the RESTORE step, ADARAI RECOVER allocates only the DD/RESTn DD/DLBLs required and allocates an equal number of input data sets for each DD/RESTn DD/DLBL.

DSIMDEV: DSIM Data Set Device Type

DSIMDEV specifies the DSIM data set device type if different from that specified by the ADARUN DEVICE parameter, which is the default.

DSIMSIZE: Size of the DSIM Data Set

The size is specified in cylinders.

When the Adabas Delta Save Facility is active on the database being recovered, this parameter *must* be specified so that ADARAI can specify the DSIMSIZE parameter for any ADARES COPY operations it may have to generate.

FEOFPL: Synchronize Multiple PLOGs

If FEOFPL=YES (the default), ADARAI ensures that protection log (PLOG) data from all of the multiple PLOG data sets has been copied:

- If the nucleus is active, ADARAI forces a protection log switch. The nucleus then calls user exit 12, which copies the log data; ADARAI waits until the copying is completed. Note that the ADARUN parameter UEX12 must therefore be specified whenever FEOFPL=YES is specified.
- If the nucleus is not active, ADARAI itself calls user exit 12, which in turn copies the log data.

In a nucleus cluster environment, FEOFPL=YES functions differently:

- If at least one Adabas nucleus is available, ADARAI calls the nucleus to switch the PLOGs.
- If no Adabas nucleus is available, ADARAI generates a job that must be executed manually.

In either case, ADARAI must be restarted with FEOFPL=NO.

FILE: File Number

FILE specifies one or more database files to be included when the recovery job stream is built. Specifying FILE causes file- rather than database-level recovery; only those files specified are involved in the RECOVER operation. If FILE is not specified, all database files are included (the default).

JCLLOG: User-Supplied Job Control

JCLLOG controls listing of the user-supplied input job control (the JCL in DDJCLIN or the JCS in JCLIN). If JCLLOG=YES is specified, the user-supplied input job control elements are printed in the utility log. The default is no listing of input job control statements (NO).

OPT: Optimize Recovery Job for a Generation

When OPT=YES is specified, ADARAI attempts to optimize the recovery job it produces for a given generation; that is, it attempts to leave out steps that are not required to bring the database or file back to its original logical state.

When OPT=NO is specified, the recovery job is not optimized.

Note:
When space on the database is limited, an optimized recovery job may fail due to the fact that the database is not built in exactly the same way as it originally was. If this occurs, a recovery job generated without optimization should be used or the size of the database increased before recovery is attempted.

PLOGDEV: Multiple PLOG Device Type

The PLOGDEV value is only used when FEOFPL=YES is specified.

PLOGDEV specifies a PLOG device type different from that specified by the ADARUN DEVICE parameter, which is the default.

RELGEN: Relative Recovery Generation Number

RELGEN specifies the *relative* generation number to be used for recovery. The current generation is always coupled with relative generation "0" (zero), which is also the default. *Two generations ago*, or the generation before the last completed generation, is specified as relative generation "2".

The generation specified must currently be in the RLOG. Use the ADARAI LIST function to see the current RLOG generations available. Note, however, that the listed generations are numbered in ascending order, beginning with generation "1", the first generation following the start of RLOG operation.

RESTFILE: Create Restore File Jobstep

When RESTFILE=NO (the default), the DDJCLOUT recovery job stream does not include ADASAV RESTORE FILE=... job steps for logged ADASAV SAVE FILE= runs. Such job steps are not included because ADARES REGENERATE does not stop at ADASAV SAVE FILE=... checkpoints.

When RESTFILE=YES, ADARAI RECOVER creates an ADASAV RESTORE FILE=... job step in the recovery job stream for every ADASAV SAVE FILE=... utility execution logged.

Note:
When using RESTFILE=YES, you must retain the file save data sets that are created in the generation.

When both RESTFILE=YES and OPT=YES are specified, the created RESTORE FILE= steps can speed the recovery process because restored files up to the RESTORE step are ignored.

When RESTFILE=YES and OPT=NO are specified, an unnecessary RESTORE step is included in the recovery job. You may wish to generate the recovery job in this way and then manually remove all steps prior to the RESTORE steps for the file(s) that are of interest.

RLOGDEV: RLOG Alternate Device

RLOGDEV specifies the device type containing the RLOG file. If the RLOG file is located on the device type specified by the ADARUN DEVICE parameter (the default device type), you do not need to specify RLOGDEV.

Examples

Example 1:

```
ADARAI RECOVER,DRIVES=3
```

The RECOVER function builds a recovery job stream based on the current generation (0, the default). The SAVE RESTORE portion of the job stream includes statements for three input data sets: DDREST1, DDREST2, and DDREST3.

Example 2:

```
ADARAI RECOVER  FILE=3,4,7,8,11
ADARAI          RELGEN=2, JCLLOG=YES
```

The recovery job stream is based on the third oldest generation; it includes activity for database files 3, 4, 7, 8, and 11 only; and creates a file-level job control. RECOVER also adds the user-supplied job control from data set DDJCLIN to the utility log.

Example3:

```
ADARAI RECOVER,RELGEN=1,OPT=Y
```

The RECOVER function builds a recovery job stream based on the last generation (i.e. the one preceding the current generation). ADARAI removes any unnecessary processing in order to speed up the recovery job.

Skeleton Job Control

Skeleton job control is contained in the DD/JCLIN file and is read as input to the RECOVER function. RECOVER merges it with the RLOG information to create the recovery job stream. Skeleton job control usually remains stable and is specific to your operating environment.

Each function in the skeleton job control is identified by a statement with the following format:

`%%`*`name`*

The name is specific to the function, such as %%JCL-ADASAV or %%JCL-STARTNUC. The job control statements follow the %% *name* statement; they are ended by the next %%JCL statement. Each skeleton section can contain any valid job control statement, including comments or program execution. This ability provides flexibility for the recovery process.

ADARAI does not check the validity of the statements in the skeleton job control. Invalid statements are first apparent when a job control error occurs during execution of the recovery job stream.

Job Header: %%JCL-JOB-HEADER

Job header statements are placed at the beginning of the recovery job stream before any other job control statements.

This job control relates to the complete recovery job and includes statements such as JOB and JOBLIB statements for z/OS or POWER JCL and JOB statements for z/VSE.

Job Trailer: %%JCL-JOB-TRAILER

Job trailer statements are placed at the end of the recovery job stream.

If the nucleus was started with the ADARUN UTIONLY=YES parameter as recommended in the %%JCL-STARTNUC section, you may want to provide a statement to execute an ADADBS OPERCOM UTIONLY=NO function in this section to make the database available after the recovery operation (see the skeleton job control examples later in this document).

Step Trailer: %%JCL-STEP-TRAILER

Step trailer statements are placed after each step in the recovery job stream.

DD/KARTE Job Control: %%JCL-DDKARTE

The operating-system-dependent DD/KARTE statements are included in each job step before DD/KARTE parameters generated by ADARAI from the RLOG.

For z/OS and z/VSE, these statements should indicate that the DD/KARTE parameters are contained in the job stream.

DD/FILEA Job Control: %%JCL-DDFILEA

This (optional) JCL card is provided to avoid problems with ADAORD REORDER processing. As a placeholder, it may be specified to provide a different DD/DLBL statement to the original DD/FILEA statement in the job. If specified, it will be inserted instead of the original DD/FILEA statement when an ADAORD REORDER is subsequently encountered.

Utility Job Control: %%JCL-*utility*

These skeleton sections are used to create utility job steps in the recovery job stream. The following utility jobs should be available in DD/JCLIN:

%%JCL-ADADEF	%%JCL-ADAORD
%%JCL-ADAINV	%%JCL-ADARES
%%JCL-ADALOD	%%JCL-ADASAV

Each of the sections should contain the following:

- The database files; for example, DD/ASSOR1, DD/DATAR1, DD/WORKR1, DD/SORTR1, DD/TEMPR1, and so on, as needed for the utility execution;

- DD/FILEA for ADALOD if used as a DD/TEMPR1 overflow file;
- A DD/PRINT and DD/DRUCK statement or assignment;
- A DD/CARD statement or assignment and all required ADARUN parameters; for example, DBID, DEVICE, PROG, SVC, and so on;
- Information needed about the Adabas library or other library.

It is possible to use a procedure or partitioned data set (PDS) member for the DD/CARD parameters, database files, or libraries.

Job Control to Start the Nucleus: %%JCL-STARTNUC

This job control comprises all the statements needed to start the Adabas nucleus. The RECOVER function uses this job control to create a job step for starting the nucleus before the first regenerate job step and, if the nucleus is not already active, before each call to a utility that requires an active nucleus.

The entire nucleus job must be included in this job control, including

- job statements;
- program execution statements;
- library definitions;
- database file definitions; and
- DD/CARD information, including the ADARUN parameters.

This section also requires a method for submitting the nucleus job control to the appropriate job entry system, such as EDT in procedure mode for BS2000 and IEBGENER for z/OS. For examples of this job control, see the %%JCL-STARTNUC sections in the examples of skeleton job control later in this document.

It is also important that this job control contain a way to stop execution of the recovery job stream until the nucleus is actually active. For example, a program can be created to issue a CL (close) command to the database; if a response code 148 (ADARSP148) indicates that the database is not active, the program can wait a specified time and reissue the CL command. The program continues until response code 0 (ADARSP000) occurs, and then ends to allow the next recover step to be performed. You can use the ADARAI CHKDB ACTIVE function for this purpose.

Job Control to Stop the Nucleus: %%JCL-ENDNUC

Whenever it detects a utility that requires an inactive nucleus, RECOVER inserts the %%JCL-ENDNUC job control in the job stream to ADAEND the Adabas nucleus. The ADADBS OPERCOM ADAEND function can also be used to stop the nucleus. If ADADBS OPERCOM is used, these job control statements must contain all necessary statements for running the ADADBS OPERCOM function. Like the Start Nucleus skeleton job control, a method to stop execution of the recovery job stream until the nucleus becomes inactive is also needed; the ADARAI CHKDB INACTIVE function can be used for this purpose.

Special Job Control Statements

The following special keywords/statements are used in the DD/JCLIN skeleton job control to control the generation of the DD/JCLOUT recovery job stream:

%STEP	When the (optional) %STEP keyword is included on the program execution statement, it generates a step number in the job stream for each job step that also includes the %STEP keyword. The step numbers run in ascending sequence, beginning with 1.
%SEQUENTIAL	*Must* be included in each %% skeleton section that generates a sequential file job control statement. ADARAI creates the necessary sequential job control statement in place of the %SEQUENTIAL statement. If this statement is not included, an error occurs during processing.
%KARTE	*Must* be included in each %% skeleton section where Adabas DD/KARTE parameters are generated. ADARAI creates the necessary DD/KARTE parameters in place of the %KARTE statement. If this statement is not included, an error occurs during processing.
%DBID	When the (optional) %DBID keyword is included on the program execution statement, it generates the five-digit database ID number. If the database number has less than five digits, the number is padded with leading zeros.

User Exit to Change JCL

ADARAI provides the user exit UEXRAI so that users may change an automatically generated recovery job before submitting it. Changes required might include the device type or the volume name.

UEXRAI obtains control of a JCL record immediately before it is written to DDJCLOUT.

The user exit is called with the following registers set:

R1	JCL record line that is about to be written to DDJCLOUT.
R13	standard 72-byte register save area
R14	return address
R15	entry point

Prerecovery Checking

Check the status of the recovery database and the recovery job stream before starting the recovery job stream.

For database-level recovery, check that

- the existing nucleus session has ended.
- the session entry has been deleted from the ID table.

Note:
Any remaining DIB entry or pending nucleus session autorestart can be ignored; it is handled automatically by the initial RESTORE step.

- all required database components (ASSO, DATA, etc.) have been formatted at least once.
-
    - Allocate and format any components changed during the generation to be recovered to the sizes and device types valid at the *beginning* of the generation.
    - Allocate and format any components that have changed size to the largest size used during the generation to be recovered.

For file-level recovery, check that

- the nucleus is active. The recovery job created by ADARAI does not start the nucleus automatically.

Restarting the RECOVER Function or Recovery Job Stream

If the ADARAI RECOVER function is interrupted, it can be restarted from the beginning, since the RECOVER function only reads the RLOG and does not change it.

The DD/JCLOUT recovery job stream created by the RECOVER function can be restarted as in a normal restore/regenerate process. However, the job stream may need to be edited to remove steps for the utility operations that were successfully completed. Following this, the recovery process can continue (providing the cause of the interruption has been removed), beginning with the failed utility operation.

It is always possible to restart an interrupted recovery job from the beginning. It may also be possible to restart the recovery job at the job step that failed or a few steps earlier, depending on the cause of the error and the job step that contained the error.