Recovery and Restart Procedures

This chapter covers the following topics:

- Introduction
- Extensions and Restrictions
- Adabas User Types
- Automatic ET Calls Issued by ADL
- Restart/Recovery Logic under ADL
- ADL Actions for Basic and Symbolic Checkpoints
- How to Restart a Batch Program

Introduction

This section describes how the Adabas Bridge for DL/I (ADL) handles DL/Icheckpoint calls and, in the case of an abnormal end to a program, how the database can be recovered and the application program restarted.

Note that the recovery and restart logic and procedures for Natural applications or programs using direct Adabas calls are not affected by ADL.

In DL/I, an application program may set checkpoints using the `CHKP` call. This results in the confirmation of all changes to the database made by the program up to this point. In addition, under z/OS, the `XRST` routine allows the use of "symbolic checkpointing", with which user data may be saved for each checkpoint. These data can later be used to restart the program, should it have terminated abnormally.

The ADL reproduces the full functionality of the DL/I `XRST` and `CHKP` calls and thus provides complete restart and recovery capabilities.

Adabas provides two basic methods of checkpointing:

- ET ("end of transaction") logic, intended to be used for updating files concurrently, and
- C1 checkpoint calls, intended to be used by programs which use files exclusively.

Both the `ET` and `C1` checkpoint methods are used by ADL.

Extensions and Restrictions

Symbolic checkpointing is also possible under z/VSE, thus allowing full recovery/restart capabilities.

The following restrictions apply to ADL:

- The use of time stamps instead of checkpoint identifiers is not supported.
- OS/VS checkpoint requests are not supported.
- (for IMS/TP BMP only) `"LAST"` may not be used to indicate that the program is to be restarted from the last issued checkpoint. Instead, establish the checkpoint identifier of the last issued checkpoint using the messages on the `DAZOUT1` file, and then supply this for the `XRST` call.

Adabas User Types

As mentioned above, the restart/recovery logic used by Adabas in a given situation mainly depends on how an application program accesses the data base. Three user types are distinguished: access-only users (ACC), exclusive file users (EXU) and ET logic users (UPD). ADL uses all of these: which type will be used in a given situation depends on the following features:

- the operational environment;
- the mode in which the application program is run (the first three characters in the parameter statement);
- how the files in the data base are accessed (read only, update), as determined by the PSB specified for the program; and
- the method of (DL/I) checkpointing used (without checkpointing, basic checkpointing or symbolic checkpointing).

The following tables show how ADL applies the Adabas user types and checkpoint methods, e.g. the last column in the first table means

- 1st table: "STA" is specified for DAZIFP, i.e. the program runs as "stand-alone".
- Last block "Update": The program makes updates.
- Last column in block "Symb": The program uses symbolic checkpointing.

Under these conditions, ADL runs the program as Adabas "EXU" user (indicated in the third line), it issues no ETs (forth line) but uses "C1" commands (fifth line) for the ADL checkpoint method.

Normal Batch Programs

Two situations may be distinguished:

- where "STA" has been specified as the first positional parameter for DAZIFP
- where "DLI", or "IMS" has been specified as the first parameter for DAZIFP

With "STA"

		Read Only			Update		
Checkpointing		Without	Basic	Symb	Without	Basic	Symb
Adabas User Type		ACC	ACC	EXU	EXU	EXU	EXU
ADL Checkpoint Method	ET		No	No		No	No
	C1		Yes	Yes		Yes	Yes

With "DLI", or "IMS"

		Read Only			Update		
Checkpointing		Without	Basic	Symb	Without	Basic	Symb
Adabas User Type		ACC	ACC	UPD	EXU	EXU	UPD
ADL Checkpoint Method	ET		No	Yes		No	Yes
	C1		Yes	Yes		Yes	Yes

The advantage of using the "STA" parameter for normal batch programs is that no ET calls have to be issued. However, other application programs also using symbolic checkpoints cannot run concurrently. This is because STA programs use the ADL directory file as exclusive (EXU) users.

SDB, BMP (z/OS) or MPS (z/VSE) Programs

		Read Only			Update		
Checkpointing		Without	Basic	Symb	Without	Basic	Symb
Adabas User Type		ACC	UPD	UPD	EXU	UPD	UPD
ADL Checkpoint Method	ET		Yes	Yes		Yes	Yes
	C1		No	No		No	No

CICS or IMS/TP Message Region

		Read Only			Update		
Checkpointing		Without	Basic	Symb	Without	Basic	Symb
Adabas User Type		ACC	UPD	-	EXU	UPD	-
ADL Checkpoint Method	ET		Yes	-		Yes	-
	C1		No	-		No	-

Automatic ET Calls Issued by ADL

Adabas ET logic allows concurrent processing of one or more files by more than one user. All records which are updated by a program are held in a queue and may not be updated by any other user until the program issues an Adabas ET call. This results in a confirmation of all updates made by the program and the release of the held records. However, the use of ET logic for programs (e.g. batch programs) which use files as exclusive users or which do not update these files is not recommended.

If ET calls are not issued often enough, the Adabas hold queue for update records may overflow, resulting in a fatal error. On the other hand, ET calls should not be issued too frequently, in order to avoid unnecessary buffer flushes.

ADL includes an option to issue ET calls automatically. The actual procedure performed depends on the program type and the operational environment. Automatic ETs will be issued in the following cases:

- **Normal Batch and SDB, BMP (z/OS) or MPS (z/VSE) Programs**
  ADL issues ET calls automatically. The ET parameter provided when creating the ADL parameter module (see the section *ADL Parameter Module* in the *ADL Installation* documentation) or as a keyword for DAZIFP, determines the number of changes which can be made to the root segment position in the database (summarized over all PCBs in the PSB) before an ET call is automatically issued. For some batch programs it may be desirable not to issue ET calls at all. This can be achieved by specifying "ET=NO".

- **IMS/TP**
  ADL automatically issues an ET call each time the application program issues a GU call on the I/O PCB.

- **CICS Environment**
  Under CICS, all users are ET logic users and ADL will automatically issue an ET call in two situations:

  - where the application program issues an explicit termination call, and
  - every time an implicit termination call is taken (i.e. at the end of a task).

When a program terminates abnormally, an Adabas BT call is issued and all database modifications made by this program are backed out.

If OPENRQ=YES has been specified when assembling the parameter module DAZPARM, ADL will issue an Adabas "OP" each time a scheduling call is received, and a "CL" for the termination call.

Restart/Recovery Logic under ADL

The figures on the following pages explain how ADL uses restart/recovery logic for different program types and operational environments.

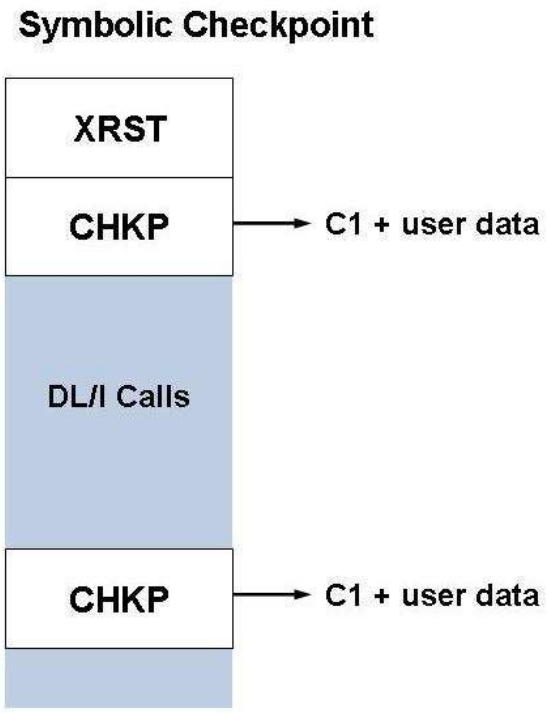
Normal Batch Programs

Two situations may be distinguished:

- with "STA" as the first positional parameter specified for DAZIFP
- with "DLI" or "IMS" specified as the first parameter for DAZIFP

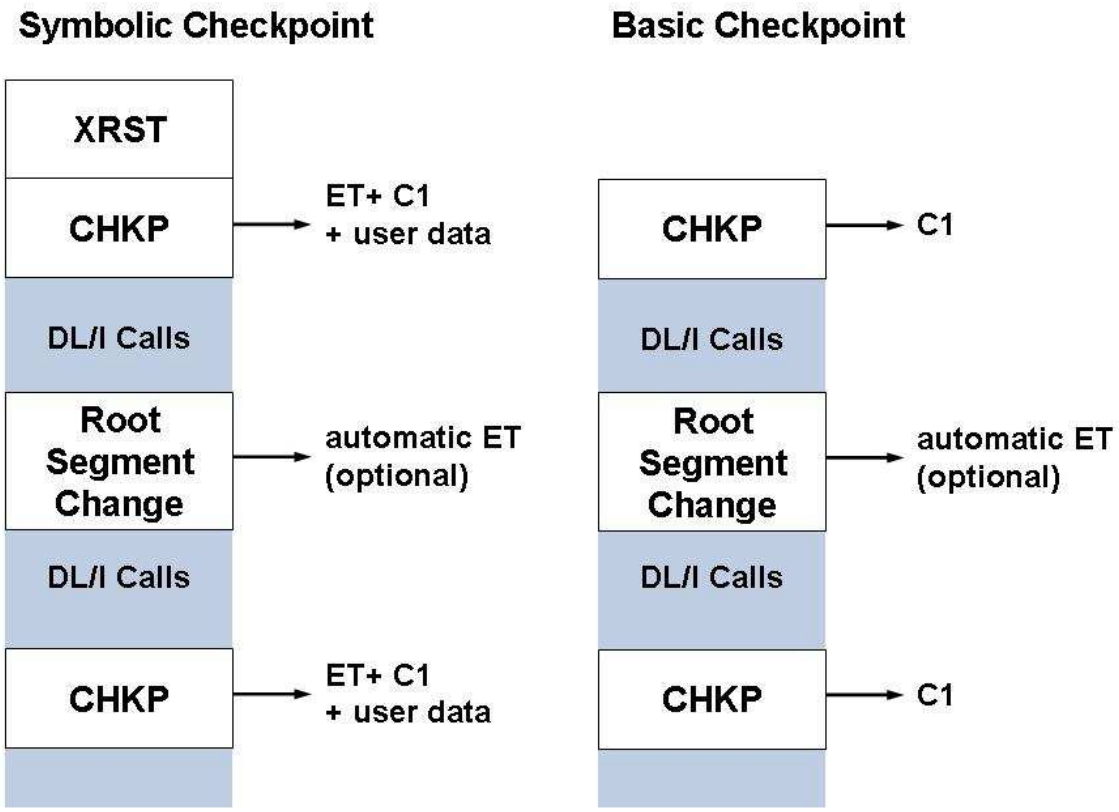
These two cases are described in the tables below.

With "STA"

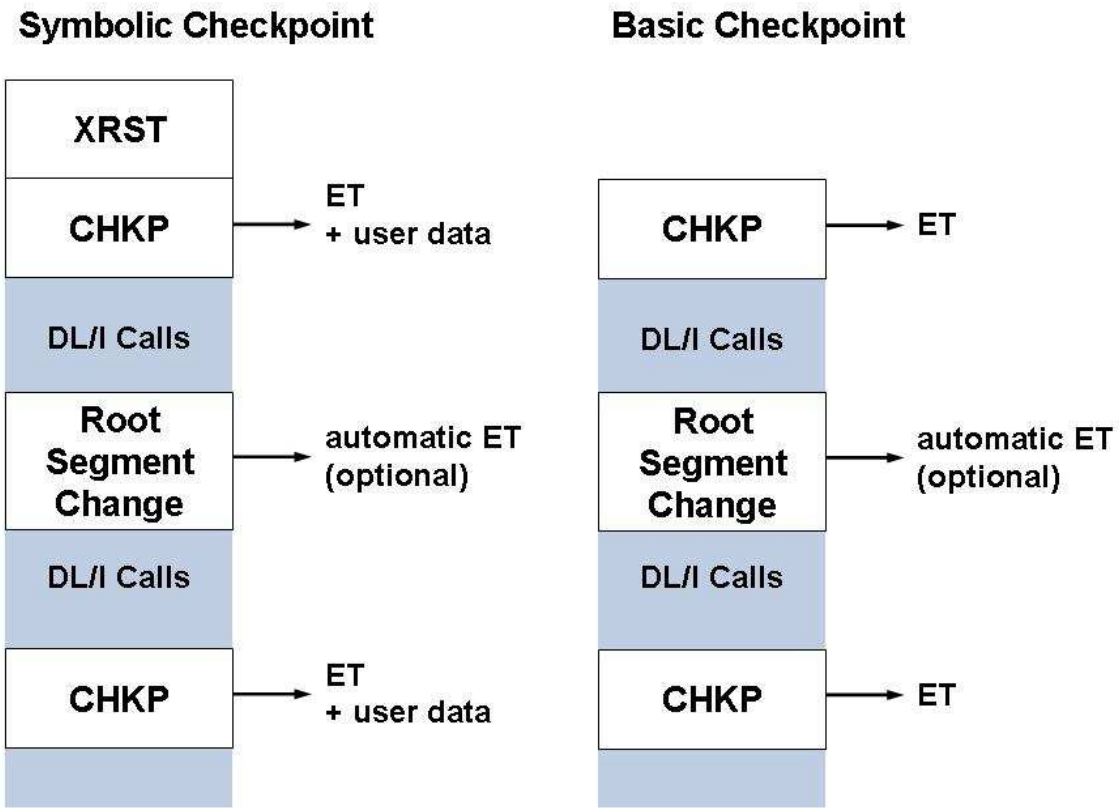


In the case of basic checkpointing, it is recommended that DLI or IMS be used instead of `STA` as the first parameter for `DAZIFP` to avoid the ADL directory file being in exclusive mode.

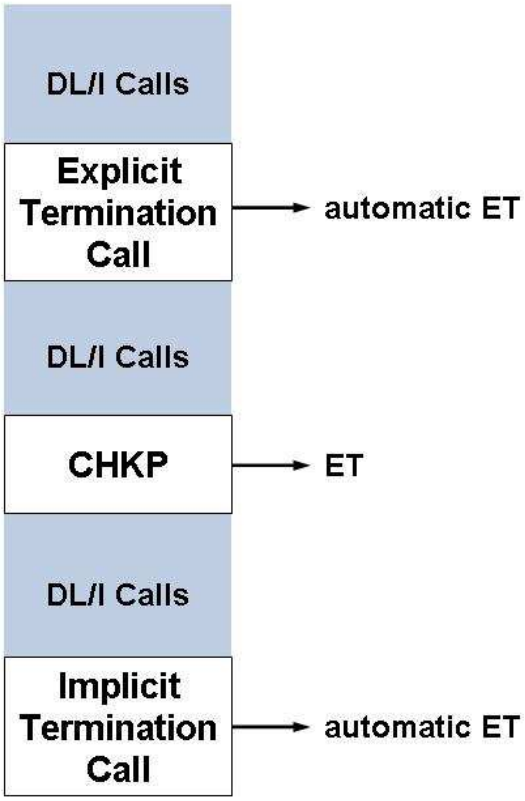
With "DLI", or "IMS"



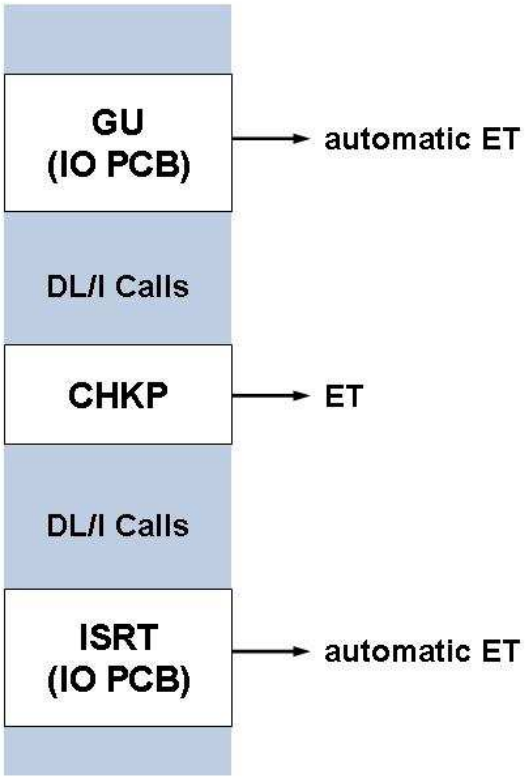
SDB, BMP (z/OS) or MPS (z/VSE) Programs


Symbolic Checkpoint
XRST
CHKP
ET
+ user data
DL/I Calls
Root
Segment
Change
automatic ET
(optional)
DL/I Calls
CHKP
ET
+ user data
Basic Checkpoint
CHKP
ET
DL/I Calls
Root
Segment
Change
automatic ET
(optional)
DL/I Calls
CHKP
ET

CICS



IMS/TP Message Region



ADL Actions for Basic and Symbolic Checkpoints

Normal Batch and SDB/BMP/MPS Programs

With a CHKP call, the following message is written to the DAZOUT1 file:

```
'DAZCHKP - CHECKPOINT WRITTEN. DATE: dd/mm/yy TIME: hh/mm/ss FOR PROGRAM progname
 WITH CHECKPOINT-IDENTIFIER chkpiden'
```

where

dd/mm/yy	is the actual date,
hh/mm/ss	the actual time,
progname	the name of the application program,
chkpiden	the eight-character checkpoint identifier as passed to the CHKP call.

If the symbolic checkpoint call is used, up to seven data areas are stored on the ADL directory file. The checkpoints on the latter are identified by the name of the program and the eight-character checkpoint identifier.

Finally, ADL issues an ET and/or a C1 call. In the Adabas checkpoint table, the checkpoints associated with the C1 call are identified by the checkpoint name "CHKP". Please refer to the *Adabas Utilities* documentation for details on how to obtain and read the Adabas checkpoint table.

If a program ends normally, the corresponding checkpoint entries on the ADL directory file will be deleted. If a program ends abnormally, these entries will be deleted after the program has been restarted. An abnormally ended program which is never restarted will thus produce garbage in the directory file. In order to clear the ADL directory file, run a dummy program containing only an XRST call. This dummy program must have the same name as the abnormally terminated program.

Alternatively, you can list and delete the checkpoint entries in the ADL directory with the ADL Online Services as described in the section *The ADL Online Services* in this documentation.

CICS and IMS/TP

ADL issues an ET call for each CHKP call.

In all cases, the current position in the data base will be deleted after a CHKP call. In the case of a restart, the position is *NOT* reestablished after an XRST call.

How to Restart a Batch Program

To restart a batch program, perform the following five steps:

Step 1

Use the output of the application program on the DAZOUT1 file to find the checkpoint identifier and the date and time of the checkpoint at which you want to restart your program.

Step 2

Find the corresponding protection log number and block number of the desired checkpoint in the Adabas checkpoint table. This table may be obtained using the Adabas ADAREP (report) utility. See the *Adabas Utilities* documentation for more details.

Step 3

Make a copy of the protection log file using the Adabas ADARES (restart) utility with either the COPY or the PLCOPY function, for single or dual protection logging respectively. The following ADARUN and ADARES statements are provided as an example (dual protection logging was active):

ADARUN input statement:

```
ADARUN PROGRAM=ADARES,MODE=SINGLE
```

ADARES input statement:

```
ADARES PLCOPY DUALPLD=3350
```

See the *Adabas Utilities* documentation for more details.

Step 4

Restore the entire data base or one or more of its files to the state in effect when the desired checkpoint was taken, using the Adabas ADARES utility with the BACKOUT function. The following ADARUN/ADARES input statements are provided as an example.

ADARUN input statement:

```
ADARUN PROGRAM=ADARES,MODE=SINGLE
```

ADARES input statement:

```
ADARES BACKOUT FILE=37,PLOGNUM=8,TOCP=CHKP,TOBLK=19
```

File 37 is restored to the checkpoint which is found on protection log file 8 in block 19 (all checkpoints issued by ADL are named CHKP). Please refer to the *Adabas Utilities* documentation and the *Adabas Operations* documentation for more details.

Step 5

Restart your program. The eight-character checkpoint identifier may be supplied either with the CPID keyword for DAZIFP or in the I/O area referenced by the XRST call in the application program. If both methods are used at the same time, the value provided with the CPID keyword will be used.

If the checkpoint is not found on the ADL directory file, or if the XRST call requests more data than have been previously stored with the CHKP call, ADL will issue an error message and abend the application program immediately.