Functional Overview

The ADACDC utility takes, as input, one or more sequential protection logs and produces ultimately, as output, a *primary output file* containing the delta of all changes made to the database over the period covered by the input protection logs. *Delta of changes* means that the last change to each ISN in a file that was altered during this period appears on the primary output file.

Notes:

1. If the ADACDC ISN parameter is specified for the run, the delta is not produced. Instead, each individual delete and insert transaction is written to the primary output file and no delta of changes is produced.
2. Spanned records are supported by the ADACDC utility when the SPANREC parameter is specified. However, when the IGNORESPANNED parameter is specified in an ADACDC run, ADACDC processing ignores any spanned records, issues a warning message, and continues its processing. A return code of "4" is returned.
3. Date-time fields with the TZ option will be decompressed in UTC time (Coordinated Universal Time, also known as Greenwich Mean Time).

The ADACDC utility helps to extract data for usage outside of Adabas.

The data in the primary output file output from an ADACDC run may be used on a regular basis to extract delta data for use with other applications.

In order to run the ADACDC utility:

- an external sorter must be available and installed as the standard sorter in the operating system. See Operating System Considerations for more information.
- the ADACDC utility must have access to the database’s Associator containing the FDTs of the files for which records are to be processed.

ADACDC uses this sort package to produce its output in ISN sequence, so all changes are written to the primary output file in ISN sequence.

Note:
A logically deleted field cannot be decompressed in ADACDC utility runs. In other words, the output from the ADACDC run will not contain logically deleted fields.

This chapter covers the following topics:

- Phases of Operation and Resulting Files
- Primary Input Data
- Primary Output File
- Transaction File

Phases of Operation and Resulting Files

ADACDC processes sequential protection logs in two phases. You can execute phase 1 and phase 2 separately, or both at once (the default):

- If phase 2 is being run separately or both phases are being completed together, the data is decompressed and written to the *primary output file*.
- If only phase 1 is being executed, the data is written to an *extract file* . This extract file may then be processed multiple times by a phase 2 operation to decompress the records and write to primary output files.

The extract file contains data records in compressed format whereas the primary output file contains records in decompressed format. Refer to the section *ADACMP (Compress - Decompress)* in the *Adabas Utilities* documentation for more information about these formats.

The primary output file and the extract file are standard sequential files that can handle variable length records.

Phase 1 and the Extract File

During phase 1, updates from the protection logs are analyzed and prefixed with a standard structure called the CDCE. The format of each record on the file is a constant CDCE prefix followed by the compressed record information.

Usually, these records are passed to an external sort routine to establish the most recent update for each ISN on a file. Only the last change for a given file and ISN combination is written to the extract file. However, if the ISN parameter is specified for the ADACDC run, the updates are still sorted in ISN order, but they are not summarized. Instead, every change transaction for an ISN is recorded in the extract file.

The extract file created when phase 1 is run separately makes it possible to process the PLOG data once and then optionally produce multiple primary output files from it based, for example, on file selection criteria. The option is useful if different file changes are required for different purposes.

When the phase 1 process is being run, the extract file is opened for output. As records are output from the sort processing, the updates for each file and ISN combination is written to the extract file if:

- the update was performed by an ET user and belongs to a completed transaction; or
- the update was performed by an EXU user and belongs to a completed command; or
- NOET is specified.

All other updates for the file and ISN combination for that period are discarded if there are no controlled utility operations against that file (see *Checkpoints Written to the Primary Output File*).

Note:
It is possible to have duplicate file and ISN combinations on the file if the ADACDC user exit (described later) adds records with file and ISN combinations that already exist. A record added or modified by the user exit is so marked in the CDCE structure.

Phase 2 or Both and the Primary Output File

The primary output file is used when both stages of ADACDC are run together, or for phase 2 processing only.

- If both phases are run together, the primary output file is opened and created directly using the output from the sort processing. In this case, processing occurs as for the extract file in phase 1 processing.
- If only phase 2 is run, the primary output file is created using input from the extract file.

The format of each record on the file is a constant CDCO prefix followed by the decompressed record information. If for some reason the record cannot be decompressed, a warning message is issued and the compressed record is written to the primary output file. A flag in the CDCO structure informs a user program when decompression for the record has failed.

Note:
It is possible to have duplicate file and ISN combinations on the file if the ADACDC user exit (described later) adds records with file and ISN combinations that already exist. A record added or modified by the user exit is so marked in the CDCO structure.

Checkpoints Written to the Primary Output File

The primary objective of the ADACDC utility is to provide an output file containing the most recent summarized changes for each ISN in a file that has been modified for the period concerned. If the ISN parameter is specified for the run, the primary objective of the ADACDC utility is to provide an output file containing the changes for each ISN in a file that has been modified for the period concerned.

Apart from simple changes to a file, some utility operations executed against a file may fundamentally affect its contents. For example, if the file is deleted, simply providing the last updates for ISNs in the file does not accurately reflect the state of the file since all ISNs have been deleted.

For this reason, the following checkpoints are recorded and written to the primary output file as appropriate with the associated indication in the output record:

```
ADASAV RESTORE FILE          File created
ADAORD STORE FILE            File created
ADALOD LOAD FILE             File created
ADALOD UPDATE FILE           File updated
ADADBS DELETE FILE           File deleted
ADADBD REFRESH FILE          File deleted
```

Because these operations can fundamentally impact a file and its appearance, the checkpoint is written to the primary output file when it occurs relative to the other updates.

ADACDC retains the last change to all ISNs before each of the above checkpoints. This means that a file and ISN combination could appear multiple times on the primary output file if one or more checkpoints were written to it.

Primary Input Data

The primary input data comprises sequential protection logs produced either by the database directly or by the ADARES PLCOPY function. If there are multiple input protection logs, concatenate them.

Note:
Spanned records are supported by the ADACDC utility when the SPANREC parameter is specified. However, when the IGNORESPANNED parameter is specified, ADACDC processing ignores any spanned records, issues a warning message, and continues its processing. When ending normally, the utility sets return code "4".

ADACDC processes this data to ensure that:

- when a new PLOG block is read and the PLOG number is the same, the PLOG block number is 1 greater than the previous PLOG block number.
- when the PLOG number itself changes, the new PLOG number is higher than the previous PLOG number and the new PLOG block number is 1.

Note:
When the PLOG number changes and the difference between the PLOG numbers is greater than 1, a warning message is issued and processing continues as this can legitimately happen if online saves are used.

If any of these checks fail, the utility execution terminates.

Primary Output File

The primary output file is a sequential file comprising all database records that were added, updated, or deleted during the period covered by the input protection logs.

If a record was changed several times, only its last change appears in the output file; ADACDC employs a sort process to identify multiple changes to the same record. However, if the ISN parameter is specified for the ADACDC run, all changes for an ISN appear in the primary output file; ADACDC still employs the sort process to put the primary output file in ISN sequence.

Each primary output file record comprises a fixed-length record prefix followed by the database record in decompressed form. The decompressed data corresponds in format to the output of the ADACMP DECOMPRESS function.

The primary output record prefix is described by the CDCO DSECT. It has the following structure:

Bytes	Description
0-1	record length (binary)
2-3	set to zeros
4-7	constant ’CDCO’
8-9	database ID
10-11	file number
12-15	ISN of the updated record
16-19	length of the decompressed data in bytes
20-47	28-byte communication ID of the last user who updated the record

Bytes	Description
48	change indicator: X’04’ record added X’08’ record updated X’0C record deleted X’10’ file created X’14’ file updated X’18’ file deleted or refreshed
49	flags (independent bit settings): X’80’ record added by user exit X’40’ record modified by user exit X’20’ record still compressed; decompression failed
50	database version indicator
51	reserved for future use
52-59	4-byte STCK, followed by a 4-byte hexadecimal counter. Users can sort on this 8-byte field to put the primary output file records back into PLOG sequence, when necessary. Read *Using ADACDC With ISNREUSE*, elsewhere in this section, for more information about when this might be necessary.
60-67	reserved for future use
68-...	decompressed record data

When SPANREC is specified, the new spanned record CDCH and CDCN output headers are used for all CDCOUT output. DSECTs for the CDCH and CDCN headers can be found in the Adabas source library. These new spanned record headers will be used when the decompressed spanned records from the PLOG exceed the physical record length, requiring the creation of multiple physical records for a single logical record. In this case, the CDCH header will prefix every logical record written to CDCOUT, regardless of whether or not the record is spanned; subsequent physical records belonging to the same logical record will be prefixed by the CDCN header. ADACDC will copy the CDCH sort key to any subsequent CDCN records.

Note:
In some cases no CDCN records may be produced. For example, if the input PLOG logical record is short enough to fit into one output physical record, only a CDCH record will be built.

Transaction File

To maintain input data checking over multiple runs of the utility, ADACDC stores information on the transaction file in a transaction control record containing the last database ID, the PLOG number, and the PLOG block number processed. This information is used to verify the latest input (unless the RESETTXF option is specified - see section *RESETTXF : Reset Input Transaction File* in *ADACDC Optional Parameters*).

ADACDC actually recognizes two different transaction files: input and output. Both transaction files are standard sequential files that can handle variable length records.

Input Transaction File Processing

During the input processing stage, ADACDC processes the input transaction file to the sort program.

Following the control record on the input transaction file, zero or more records may be found. These are database updates related to transactions not completed during the last run of the utility. These records are processed again as part of the input as their transactions will normally have been completed in the next sequential protection logs provided to the utility. This is the reason the sequence of protection logs is so important: updates may remain outstanding forever if the correct sequence is not maintained.

The transaction file also records whether the NOET option was specified during the last phase 1 run of the utility. When ADACDC detects that this option has changed from one utility execution to the next, it uses the information from the control record on the input transaction file; however, all transactional information in the other records is ignored. This is due to the fact that changing this option may cause inconsistent data to be written to the primary output file or extract file, as appropriate. ADACDC issues a warning when this happens.

Output Transaction File Processing

Once output processing from the sort program starts, the input transaction file is closed and the output transaction file is opened. The control record is written to the output transaction file followed by any updates that relate to incomplete transactions or, in the case where the NOET option is in effect or an EXU user is in control, to incomplete commands. The output transaction file is closed once processing is complete.

Using a Single Transaction File

It is possible to use the same file as both the input and output transaction file; however, if the utility fails while writing to the output transaction file (that is, at any time during the output processing of the sort utility), the input transaction file will no longer exist and therefore, rerunning the utility will yield a different result.

For this reason, the transaction file must be backed up prior to the utility run so that it can be restored in the event of a failure.

Alternatively, you could use a facility on your operating system (if available) that produces a new version of a file whenever a program updates the file.