Running the Utility

```
ADACDC  [FILE= filelist ]
        [ISN]
        [NOET]
        [PHASE={ 1 | 2 | BOTH ]
        [RESETTXF]
```

The first time you run the ADACDC utility, use the following syntax and either not specify or dummy the input transaction file (CDCTXI) to create a valid transaction file for input to subsequent ADACDC runs:

```
ADACDC RESETTXF,PHASE=BOTH
```

The RESETTXF option ignores the absent or dummied input transaction file, reads the primary input data, and produces primary output using the input data.

After the input transaction file has been created during the first run, only the utility name ADACDC is required to run this utility; the PHASE parameter defaults to BOTH. Parameter options are explained in the following sections.

- Optional Parameters
- Using ADACDC With ISNREUSE

Optional Parameters

FILE: Files Processed

Use the FILE parameter to limit the file(s) processed by the utility and written to the output file:

- For phase 1 operation, only records relating to the files specified are written to the extract file.
- For phase 2 and BOTH operations, only records relating to the files specified are written to the primary output file.

Note:
Clearly, files required for phase 2 processing must have been specified on the previous phase 1 operation that created the input extract file.

When this parameter is not specified, all files are processed by the utility.

ISN: Record Delete and Insert Transactions Separately

Ordinarily, ADACDC processing consolidates all delete and insert transactions to the same ISN, creating a single update transaction for the ISN. However, if you specify the ISN parameter, each delete and insert transaction is recorded in the primary output file (CDCOUT) individually. So, when you use the ADACDC utility with the ISN parameter, the number of records produced in the primary

output file will increase, possibly dramatically.

NOET: Bypass ET Processing

ADACDC normally accepts for processing only those records that are part of completed transactions or, in the case of EXU users, records that are part of completed commands.

Use the NOET option to bypass this transaction processing when PHASE=1 or PHASE=BOTH. NOET has no effect when PHASE=2 because the input is the extract file from phase 1 which has already processed the protection log (PLOG) input with or without the NOET option.

When NOET is specified, any update made to the database is processed and written to the extract file (PHASE=1) or primary output file (PHASE=BOTH) as soon as it is encountered on the PLOG.

Warning:
Specifying this option may result in updates recorded on the primary output file that are related to transactions that were not complete at the end of the input PLOG.

PHASE: Execution Phase

The PHASE parameter determines the input the utility requires and the output it produces:

- PHASE=1 reads the sequential PLOG input and produces an interim extract file for later processing by a phase 2 step.
- PHASE=2 reads an extract file produced by a previously executed phase 1 step and produces a primary output file containing the delta of changes made to the file.
- PHASE=BOTH (the default) reads the sequential PLOG input and produces the primary output file containing the delta of changes directly without reading or writing an extract file.

Refer to the section *Phases of Operation and Resulting Files* for more information.

RESETTXF: Reset Input Transaction File

ADACDC checks the primary input data to the utility to ensure that the PLOGs are read in sequence, by PLOG block and PLOG number. If these checks fail, the utility execution terminates.

To maintain the checks over multiple runs of the utility, ADACDC maintains input and output transaction files. These files also track record updates related to incomplete transactions or, in the case of EXU users, incomplete commands from one utility execution to the next. Normally, such incomplete transactions or commands are completed in the next sequential PLOGs provided to the utility.

However, if the need arises to process PLOGs out of sequence and the information in the transaction file can be safely removed, the RESETTXF option can be used to reset the transaction file so that the checks are bypassed and all outstanding transaction or command data is ignored for a given run. ADACDC ignores information on the input transaction file and writes the output transaction file at end of job.

Warning:
If the sequence of PLOGs is interrupted, record updates related to incomplete transactions recorded in the transaction file may remain outstanding indefinitely.

Using ADACDC With ISNREUSE

Normal ADACDC processing produces a primary output file in ISN sequence. Ordinarily, this processing works fine. However, if the database file was created with the ADADBS or ADALOD ISNREUSE option specified, errors (response 98) can occur. These errors can occur because an ISN might have been reused, so multiple transactions may reside in the PLOG for the same unique descriptor key (UQ) with different ISNs.

To resolve these problems, the following steps should be taken:

1. Run the ADACDC utility with the ISN parameter specified. This will give you a granular list of changes in the primary output file, instead of attempting to summarize the changes by ISN. The data in the primary output file after this run will still be in ISN sequence.

2. Sort the primary output file (CDCOUT) in PLOG sequence prior to applying its data to your data warehouse or other application. This sort should be performed on the CDCOUT data at offset 52 for 8 bytes.

   Once the data is sorted in PLOG sequence, the data can be applied to your data warehouse or other application.

3. Sort the transaction file (CDCTXI/CDCTXO) back into PLOG sequence prior to running any additional ADACDC jobs. If you neglect to do this, future runs of ADACDC may be compromised.

   To get the transaction file (CDCTXI/CDCTXO) in the correct sequence, two sorts are actually required, in the following sequence:

   1. First run a sort that puts the CDCE records in PLOG sequence. This sort should be run on offset 68 for 4 bytes and then on offset 16 for 4 bytes. For example:

      ```
      SORT FIELDS=(68,4,BI,A,16,4,BI,A),RECORD TYPE=V,LENGTH=32756
      ```

   2. The second sort should sort the transaction file back into CDCC, CDCE, CDCX order. This sort should be run on offset 4 for 4 bytes.

   When these two sorts have been run, the transaction file should be ready to be processed by future ADACDC jobs.