LOAD: Load a File

LOAD: Load a File

Use the LOAD function to load a file into a database.

```
ADALOD LOAD  FILE = file-number [ , filetype ]
             DSSIZE = size
             MAXISN = max-number-of-records [MAXISN2 = max-number-of-secondary-spanned-records ]
             SORTSIZE = size
             TEMPSIZE = size
             [ACRABN = starting-rabn ] [AC2RABN = starting-rabn ]
             [ADAMFILE ADAMDE = { field | ISN } [ADAMOFLOW = size ] [ADAMPARM = { number | 0 } ] ]
             [ALLOCATION = { FORCE | NOFORCE } ]
             [ANCHOR = file-number MINISN = lowest-allocated-isn , NOACEXTENSION ]
             [ASSOPFAC = { padding-factor | 10 } ] [ASSOVOLUME = ' Associator-extent-volume ' ]
             [{BASEFILE | LOBFILE} = file-number ]
             [DATAFRM = { YES | NO }]
             [DATAPFAC = { padding-factor | 10 } ] [DATAVOLUME = ' Data-Storage-extent-volume ' ]
             [DSDEV = device-type ] [DSRABN = start-rabn ] [DSREUSE = { YES | NO }]
             [ETID = owner-id ]
             [IGNFDT]
             [INDEXCOMPRESSION = { YES | NO }]
             [ISNREUSE = { YES | NO ] [ISNSIZE = { 3 | 4 } ]
             [LIP = { isn-pool-size | 2000 } ]
             [LOWNERID = { owner-id-length | 0 } ]
             [LWP = { work-pool-size | 1048576 } ]
             [MAXDS = max-DS-secondary-allocation ]
             [MAXNI = max-NI-secondary-allocation ]
             [MAXRECL = max-compressed-record-length ]
             [MAXUI= max-UI-secondary-allocation ]
             [MINISN = { lowest-allocated-isn | 1 } ]
             [MIXDSDEV]
             [NAME = { name | TESTFILE } ]
             [NIRABN = start-rabn ] [NISIZE = size ]
             [NOACEXTENSION]
             [NOUSERABEND]
             [NUMREC = max-number-of-records-to-load ]
             [PGMREFRESH = { YES | NO } ]
             [READONLY = { YES | NO } ]
             [REPLICATOR]
             [RESTART]
               RPLTARGETID = ' reptor-target-id '
                   [RPLDSBI]
                   [RPLERRORDEACTFILE = { NO | YES} ]
                   [RPLINITERROR = { FAIL | CONTINUE} ]
                   [RPLKEY = ' primary-key-for-replication ' ]
                   [RPLLOAD = { YES | FILE | NO } ]
             [RPLUPDATEONLY = { YES | NO } ]
             [SKIPREC = { number | 0 } ]
             [SLOG]
             [SORTDEV = device-type ]
             [SYFMAXUV = nn ]
             [TEMPDEV = device-type ]
             [TEST]
             [UIRABN = start-rabn ] [UISIZE= size ]
             [UQDE = descriptor-list ]
             [USERISN = { YES | NO } ]
             [VERSION = { 4 | 5 | 6 | 7 } ]
```

This chapter covers the following topics:

- Essential Parameters
- Optional Parameters and Subparameters
- Examples
- LOAD Data and Space Requirements
- Loading Expanded Files
- Loading Multiclient Files

Essential Parameters

DSSIZE: Extent Size for Data Storage

DSSIZE is the count of blocks or cylinders to be assigned to the file's Data Storage logical extent. This value must be specified. Block values must be followed by a "B" (for example, "5000B").

The number can be taken directly from the Space Requirements report produced by the ADACMP utility. If the specified extent size exceeds the largest free size, ADALOD allocates as many file extents as necessary (up to a total of 5) to satisfy the request.

If a small number of records is being loaded now and a larger number of records is to be added later, the ADACMP report value should be increased in proportion to the total records to be added; otherwise, the space allocation for Data Storage (the original and four additional extents) may not be large enough to accommodate the records to be added. The file must then be unloaded and reloaded (or reordered) to increase the Data Storage space allocation. For more information, see the section *LOAD File Space Allocation* in the *LOAD Data and Space Requirements* section.

FILE: File Number, File Type

FILE specifies the Adabas file number and file type to be assigned to the file.

The number specified must not be currently assigned to another file in the database, unless that file was first deleted using the KEEPFDT parameter (see ADADBS DELETE function). The number must not be greater than the maximum file number defined for the database; for a checkpoint, security, or trigger, or system file, the number must be 5000 or lower. File numbers may be assigned in any sequence.

The file type is optional and is used to indicate that the file is an Adabas system file or an Adabas LOB file. One of the following keywords may be specified:

CHECKPOINT	Adabas checkpoint system file
LOB	Adabas *LOB file*
SECURITY	Adabas security system file
SYSFILE	Adabas system file
TRIGGER	Adabas trigger system file

Notes:

1. An existing checkpoint system file created using the ADADEF utility cannot be overwritten.
2. The security system file is required if Adabas Security is to be used.
3. In an Adabas Transaction Manager (ATM) database, SYSFILE numbers 5 and 6 are reserved for the ATM nucleus. For Adabas version 7.1, these file numbers cannot be changed. The file numbers are more flexible in subsequent versions of Adabas.
4. Use the following parameters to load the ATM system files on an ATM database (ADARUN DTP=TM): ADALOD LOAD FILE=5,SYSFILE , ADALOD LOAD FILE=6,SYSFILE
5. If CHECKPOINT, SECURITY, or TRIGGER is specified, the contents of //DDEBAND are ignored.
6. No //DDEBAND data set need be supplied if you are loading an empty LOB file.
7. CHECKPOINT, SECURITY, or SYSFILE files can be deleted only by the ADADBS DELETE function running as the only Adabas user; deleting a system file terminates Adabas when deletion is completed.
8. Adabas allows a maximum of eight (8) system files.
9. If a *LOB file* is being loaded, the parameters ADAMFILE, ANCHOR, LOWNERID, NUMREC, SKIPREC, and UQDE cannot be specified in the ADALOD LOAD run.

MAXISN: Maximum ISN Count

The MAXISN parameter is required. Specify the maximum number of ISN mappings in the address converter (AC). ADALOD determines the number of ISN mappings to allow space for in the AC using the calculation:

```
(MAXISN - MINISN) + 1
```

There is no default value.

The MAXISN and MINISN values you specify are used to calculate the initial number of AC blocks to allocate during the ADALOD execution. Depending on the size of RABNs in the database (which is determined by the ADADEF DEFINE parameter RABNSIZE), each RABN requires 3 or 4 bytes in the AC. In addition, the block size of each AC block depends upon the device type of the Associator. So the number of AC blocks that should be allocated is affected by the number of ISN mappings allowed, the RABN size, and the block size of the Associator device.

To calculate the number of AC blocks that must be allocated, ADALOD uses the following calculation and rounds up to the nearest integer:

```
(#-of-ISN-mappings X RABN-size)/device-blocksize
```

For example, assume the RABN size for the database is set to "3" (the ADADEF DEFINE RABNSIZE parameter) and that the block size of the device on which the Associator resides is 2544 bytes. If MAXISN=1000 and MINISN=1, ADALOD calculates that the actual number of ISNs to be mapped as (1000 -1)+1 (`MAXISN – MINISN +1`), or 1000. It then multiplies 1000 by three (the RABN size), to get 3000 bytes. Finally, it divides 3000 by 2544 (the block size of the device), resulting in a value of roughly 1.18, which it rounds up to two. So ADALOD determines that two AC blocks should be allocated for this ADALOD run. (Note that on the corresponding ADAREP report, the "MAX-ISN Expected" value would not be listed as 1000; instead it is listed as the actual number of ISNs that would fit into two AC blocks – in this case about 1694.)

If more than *(MAXISN - MINISN) + 1* records are to be loaded, and if NOACEXTENSION is *not* specified, ADALOD increases the MAXISN value and allocates an additional AC extent.

MAXISN does not specify the maximum number of records that can be loaded into the file. The maximum number of records that Adabas permits in a file depends on the ISNSIZE parameter, which specifies whether ISNs in the file are 3 bytes or 4 bytes long. (If ISNSIZE=3, Adabas permits up to 16,777,215 records. If ISNSIZE=4, Adabas permits up to 4,294,967,294 records.)

SORTSIZE: Sort Size

SORTSIZE specifies the space available for the sort data set or data sets R1/2 (SORTR2 is not supported under z/VSE). The value can be either cylinders (a numeric value only) or blocks (a numeric value followed by a "B"). If blocks are specified, they should be equivalent to a full number of cylinders. The SORTSIZE parameter must be specified. Refer to the *Adabas DBA Reference* documentation for more information on estimating the sort space.

TEMPSIZE: Temporary Storage Size

TEMPSIZE specifies the size of the temp data set for the file. The Temp size equals the total of TEMP space required for each descriptor in the file; see the section *LOAD File Space Allocation* in the *LOAD Data and Space Requirements* section for more information. The size can be either in cylinders or blocks (followed by a "B").

Optional Parameters and Subparameters

ACRABN/ AC2RABN/DSRABN/ NIRABN/ UIRABN: Starting RABN

Causes space allocation for the address converter (ACRABN), secondary address converter (AC2RABN), Data Storage (DSRABN), the normal index (NIRABN), or the upper index (UIRABN) to begin at the specified RABN.

ADAMFILE: File to Be Loaded with ADAM Option

ADAMFILE specifies the file is to be loaded using the ADAM option.

If this parameter is specified, the Data Storage RABN for each input record is calculated using a randomizing algorithm, the result of which is based on the value of the ADAM descriptor in each record. See the ADAMER utility description for additional information about using the ADAM option. If ADAMFILE is specified, ADAMDE must also be specified.

Note:
If a *LOB file* is being loaded (read about the FILE parameter), the ADAMFILE parameter cannot be specified in the ADALOD LOAD run.

ADAMDE: ADAM Key

ADAMDE specifies the field to be used as the ADAM key.

The ADAM descriptor must be defined in the field definition table (FDT). The descriptor must have been defined with the UQ option, and *cannot*

- be a sub-, super-, hyper-, collation, or phonetic descriptor;
- be a multiple-value field;
- be a field within a periodic group;
- be variable length;
- specify the null suppression (NU) option.

If the ISN of the record is to be used as the ADAM key, ADAMDE=ISN must be specified.

This parameter must be specified when the ADAM option has been selected for the file being loaded with the ADAMFILE parameter.

ADAMOFLOW: Overflow Area Size for ADAM File

ADAMOFLOW is the size of the Data Storage area to be used for ADAM file overflow. The ADAMOFLOW value applies only if the ADAM option has been selected for the file being loaded (see ADAMFILE parameter).

ADALOD will choose a prime number which is less than DSSIZE minus ADAMOFLOW (in blocks). This prime number is used to compute the Data Storage RABN for each record. If a record does not fit into the block with the computed RABN, it is written to the next free RABN in the overflow area.

ADAMPARM: Bit Truncation for ADAM File

ADAMPARM specifies the number of bits to be truncated from the ADAM descriptor value before it is used as input to the ADAM randomizing algorithm. A value in the range 1-255 may be specified. If this parameter is omitted, a value of 0 bits (no truncation) will be used.

This parameter achieves a type of record clustering, with nearly equal ADAM keys. ADAMPARM can be specified only when the ADAMFILE parameter has also been specified.

ALLOCATION: Action to Follow File Extent Allocation Failure

ALLOCATION specifies the action to be taken if file extent allocations cannot be obtained according to the placement parameters ACRABN, DSRABN, NIRABN, or UIRABN.

By default (that is, ALLOCATION=FORCE), the utility terminates with error if any file extent allocation cannot be met according to RABN placement parameters.

If ALLOCATION=NOFORCE is specified and any allocation with placement parameters fails, the utility retries the allocation without the placement parameter.

If insufficient space can be obtained according to the placement parameters DSRABN, NIRABN, or UIRABN, only the first extent will be made there and the rest (until the fifth extent) will be made elsewhere. But if the placement parameter ACRABN is used with ALLOCATION=FORCE, the complete space has to be available there; otherwise, the utility terminates with an error.

ANCHOR: Expanded Component/ Anchor File

ANCHOR defines the base (anchor) file for either an existing or a new expanded file. If the file defined by ANCHOR is the same as that defined by the FILE parameter, the loaded file becomes the physical base (anchor) file for a new expanded logical file. Otherwise, the FILE file becomes a new component of the expanded file defined by ANCHOR.

If ANCHOR specifies a file that is not part of an expanded file, the LOAD operation defines this file and the file specified by the FILE parameter as a new expanded file. It also sets the NOACEXTENSION indicator for the file specified by ANCHOR.

If ANCHOR specifies the anchor file of an already existing expanded file, the LOAD operation adds the file specified by FILE to the expanded file.

Note:
When loading a new file to an existing expanded file, you must have exclusive update use of the anchor file as well as the file being added. This can be achieved by locking the anchor file for utility use.

Both the file specified by ANCHOR and the file specified by FILE must have the same field definition table (FDT) structure. The maximum record length (MAXRECL parameter) and any file security definitions must also be the same.

If ANCHOR is specified, the MINISN and NOACEXTENSION parameters must also be specified. Coupled files or multiclient files cannot be part of expanded files.

Note:
If a *LOB file* is being loaded (read about the FILE parameter), the ANCHOR parameter cannot be specified in the ADALOD LOAD run.

ASSOPFAC: Associator Padding Factor

ASSOPFAC defines the padding factor to be used for each Associator block. If not specified, the default padding factor is 10.

The value specified represents the percentage of each Associator block (padding area) that is not to be used during the loading process. The padding area is reserved for use when additional entries must be added to the block for new descriptor values or new ISNs for existing values, thereby avoiding the overhead caused by relocating overflow entries into another block.

A value in the range 1-90 may be specified. The number of bytes contained in an Associator block, minus the number of bytes reserved for padding, must be larger than the largest descriptor value contained in the file, plus 10 bytes.

A small padding factor (1-10) should be specified if little or no descriptor updating is planned. A larger padding factor (10-50) should be specified if a large amount of updating including addition of new descriptor values (or new ISNs) is planned.

ASSOVOLUME: Associator Extent Volume

Note:
The value for ASSOVOLUME must be enclosed in apostrophes.

ASSOVOLUME specifies the volume on which the file's Associator space (that is, the AC, NI, and UI extents) is to be allocated. If the requested number of blocks cannot be found on the specified volume, ADALOD retries the allocation while disregarding the ASSOVOLUME parameter.

Note:
If there are five or more blocks of unused ASSO space on the specified volume, ADALOD allocates these blocks; if this is not enough space, it ends with ERROR-060. If there are no free blocks remaining on the specified volume, ADALOD tries to allocate space on another volume.

If ACRABN, UIRABN, or NIRABN is specified, ADALOD ignores the ASSOVOLUME value when allocating the corresponding extent type. If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADALOD's default allocation rules.

BASEFILE: Base File Number

BASEFILE specifies the file number of the *base file* associated with the *LOB file* you are loading. This parameter is only used when loading LOB files.

For more information, read *Large Object (LB) Files and Fields*.

DATAFRM: Overwrite ADAM Data Storage

DATAFRM controls overwriting of an ADAM file's Data Storage during loading. DATAFRM=YES (the default) forces ADALOD to reformat the Data Storage area when the file is loaded; DATAFRM=NO prevents reformatting, and is recommended when loading relatively few records because the load operation may run significantly faster. Specifying NO, however, assumes that the Data Storage area was previously formatted with the ADAFRM utility specifying FROMRABN.

Warning:
Specify DATAFRM=NO with care. If the primary Data Storage area was incorrectly formatted, later file processing could cause errors and unpredictable results.

DATAPFAC: Data Storage Padding Factor

DATAPFAC is the padding factor to be used for each Data Storage physical block. A percentage value in the range 1-90 may be specified. If not specified here, the default padding factor is 10.

A small padding factor (1-10) should be specified if little or no record expansion is expected. A larger padding factor (10-50) should be specified if a large amount of updating is planned that will expand the logical records.

The percentage value specified represents the portion of each Data Storage block (padding area) to be reserved during the loading process for later record expansion. The padding area is used when any given logical record within the block requires additional space as the result of record updating, thereby avoiding the overhead that would be needed to relocate the record to another block.

Since records loaded into a file can be different lengths, the padding factor cannot be exactly the percentage specified in each block. Adabas balances the size of the padding area for the different record lengths to the extent that at least 50 bytes remain in a block.

Example:

A block size is 1000 bytes; the padding factor is 10%. The space available for loading records (block size - padding-area) is therefore 900 bytes.

After loading some records, 800 bytes of the block have been used. The next record is 170 bytes long. This record cannot be loaded into the current block because less the 50 bytes would remain in the block after the record was loaded. Therefore, the record is loaded into the next block.

The current block remains filled to 800 bytes. The difference between 800 and 900 bytes (that is, -100 bytes) is used for balancing.

Suppose the next record had been 150 bytes instead of 170 bytes, and assume that the cumulative balancing value at that point in time is a negative number of bytes. The 150-byte record would be loaded because 50 bytes would remain in the block after the record was loaded (1000 - 950).

However, 50 bytes of the padding area would have been used (900 - 950) leaving +50 bytes for balancing.

For files loaded with the ADAM option, a new record is loaded into its calculated Data Storage block if space is available in the block (including the padding area). Records that cannot be stored in their calculated block are stored in another block (in this case, the padding area is not used).

DATAVOLUME: Data Storage Extent Volume

Note:
The value for DATAVOLUME must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file’s Data Storage space (DS extents) is to be allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADALOD retries the allocation while disregarding the DATAVOLUME value.

If DSRABN is specified, DATAVOLUME is ignored for the related file. If DATAVOLUME is not specified, the Data Storage space is allocated according to ADALOD’s default allocation rules.

DSDEV: Data Storage Device Type

DSDEV specifies the device type on which the file’s Data Storage is to be loaded. There is no default value; if DSDEV is not specified, an arbitrary device type is used.

DSREUSE: Data Storage Reusage

DSREUSE indicates whether Data Storage space which becomes available is to be reused. The default is YES.

ETID: Multiclient File Owner ID

The ETID parameter assigns a new owner ID to all records being loaded into a multiclient file. It specifies the user ID identifying the owner of the records being loaded. The owner ID assigned to the records is taken from the user profile of the specified user ID.

The ETID parameter must be specified if the file is to be loaded as a multiclient file (see the LOWNERID parameter discussion) and the input file contains no owner IDs; that is, the input file was not unloaded from a multiclient source file.

ETID is optional if the input file was unloaded from a multiclient source file. In this case, the loaded records keep their original owner IDs.

The ETID parameter must not be specified when loading a non-multiclient file.

Note:
If the ETID parameter is used, the ADALOD utility requires an active nucleus. The nucleus will translate the ETID value into the internal owner ID value.

IGNFDT: Ignore Old FDT

When a file is deleted using the ADADBS DELETE function with the KEEPFDT parameter, the field definition table (FDT) remains in the Associator. When the file is again reloaded and IGNFDT is not specified, ADALOD compares the file's old FDT with the new one (security information is not compared). If both FDTs are identical, ADALOD loads the file and replaces the old FDT with the new FDT. If the FDTs are not identical, the old FDT is kept and the ADALOD operation ends with an error message.

Specifying the IGNFDT parameter causes ADALOD to ignore any existing (old) FDT for the file; no comparison is made. The new FDT replaces the old FDT, and ADALOD loads the file.

INDEXCOMPRESSION: Compress File Index

INDEXCOMPRESSION indicates whether the index of the file is loaded in compressed or uncompressed form. A compressed index usually requires less index space and improves the efficiency of index operations in the Adabas nucleus.

If INDEXCOMPRESSION is not specified, ADALOD obtains the default value from the sequential input file. If the input file was created using

- ADACMP, the default value is NO.
- ADAULD, the value of the file at the time of the unload is taken as the default.

ISNREUSE: ISN Reusage

ISNREUSE indicates whether or not an ISN freed as the result of deleting records may be reassigned to a new record. The default is NO.

ISNSIZE: 3- or 4-Byte ISN

ISNSIZE indicates whether ISNs in the file are 3 or 4 bytes long. The default is 3 bytes.

LIP: ISN Buffer Pool Size

LIP specifies the size of the ISN pool for containing ISNs and their assigned Data Storage RABNs. The value may be specified in bytes as a numeric value ("2048") or in kilobytes as a value followed by a "K" ("2K"). The default for LIP is 2000 bytes.

LIP can be used to decrease the number of address converter I/Os during loading when the USERISN=YES and the user-supplied ISNs are unsorted. Optimum performance is obtained if LIP specifies a buffer size large enough to hold all ISNs to be processed.

The length of one input record is ISNSIZE + RABNSIZE + 1. Thus the entry length is at least 7 bytes (the ISNSIZE of the file is 3 and the RABNSIZE of the database is 3) and at most 9 bytes (the ISNSIZE is 4 and the RABNSIZE is 4).

LOBFILE: LOB File Number

LOBFILE specifies the file number of the *LOB file* associated with the *base file* you are loading. This parameter is only used when loading base files.

For more information, read *Large Object (LB) Files and Fields*.

LOWNERID: Internal Owner ID Length for Multiclient File

The LOWNERID parameter specifies the length of the internal owner ID values assigned to each record for multiclient files. Valid length values are 0-8. If the LOWNERID parameter is not specified, its default value is the length of the owner IDs in the input file.

The specified or default value of the LOWNERID parameter determine whether a file is to be loaded as a multiclient or a non-multiclient file. If the effective LOWNERID value is zero, the file is loaded as a normal, non-multiclient file; if it is nonzero, the file is loaded as a multiclient file.

In combination with the ETID parameter, the LOWNERID parameter can be used to

- reload a non-multiclient file as a multiclient file;
- increase/decrease the length of the owner ID for the file; or
- remove the owner ID from the records of a file.

The following table shows the possible combinations of the LOWNERID parameter and the owner ID length in the input file.

LOWNERID Parameter Setting	Owner ID Length Value in Input File	
	0	2
0	Keep as a non-multiclient file	Convert to a non-multiclient file
1	Set up multiclient file (ETID)	Decrease owner ID length
2	Set up multiclient file (ETID)	Keep owner ID length
3	Set up multiclient file (ETID)	Increase owner ID length
(not specified)	Keep as a non-multiclient file	Keep as a multiclient file

When loading a multiclient file (the specified or default value of LOWNERID is non-zero), the ETID parameter can be specified to assign a new owner ID to all records being loaded. If the input file already contains owner IDs and ETID is omitted, all records keep their original owner IDs.

Where the table indicates the ETID parameter in the "Owner ID Length...0" column, the ETID parameter is mandatory, as there are no owner IDs given in the input file.

Note:
If a *LOB file* is being loaded (read about the FILE parameter), the LOWNERID parameter cannot be specified in the ADALOD LOAD run.

LWP: Work Pool Size

LWP specifies the size of the work pool to be used for descriptor value sorting. The value can be specified in bytes or kilobytes followed by a "K". If no value is specified, the default is 1048576 bytes (or 1024K); however, to shorten ADALOD run time for files with very long descriptors or an unusually large number of descriptors, set LWP to a higher value. To avoid problems with the sort data set, a smaller LWP value should be specified when loading relatively small files.

The minimum work pool size depends on the sort data set’s device type:

Sort Device	Minimum LWP	Minimum LWP
	Bytes	Kilobytes
2000	106496	104K
2314	090112	88K
3375	131072	128K
3380	139264	136K
3390	159744	156K

MAXDS/ MAXNI/ MAXUI: Maximum Secondary Allocation

Specifies the maximum number of blocks per secondary extent allocation for Data Storage (MAXDS), normal index (MAXNI), or upper index (MAXUI). The value specified must be in blocks (for example, MAXNI=8000B) and cannot be more than 65535B. If no limit is specified, no limit is assumed (the default).

MAXISN2: Allocate Secondary Address Converter RABNs to Account for Maximum Secondary ISNs

The MAXISN2 parameter is optional, regardless of whether or not spanned records exist in the ADALOD input file or not. Use this parameter to specify the desired size of the secondary address converter (AC2) in ISNs. The secondary address converter is used to map secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADALOD determines the number of ISN mappings for which to allow space in the secondary AC using the following method:

1. If a secondary AC is not yet allocated, one secondary AC block is allocated.
2. If the current MAXISN2 setting is greater than or equal to four times the MAXISN value (the current maximum primary ISNs expected), the same algorithm is used to determine the number of secondary ISNs as is used to allocate additional primary ISNs.
3. If none of the above conditions is met, then secondary AC space is allocated as the smaller of the following two calculations:
   - The product of 10 times the old MAXISN2 setting (10 x oldMAXISN2)
   - The sum of the old MAXISN2 setting and the MAXISN setting (oldMAXISN2 + MAXISN)

There is no default value.

MAXRECL: Maximum Compressed Record Length

MAXRECL specifies the maximum compressed record length permitted for the file. The default is the maximum length supported by the device type being used.

MINISN: Lowest ISN Count

This parameter specifies the lowest number of ISNs that can be assigned in the file. The default is 1.

The main purpose of MINISN is to assign the low end of the ISN range for a component file of an Adabas expanded file. MINISN is required when ANCHOR is specified for an expanded file.

Use MINISN to avoid wasting Associator space in files where all records are assigned ISNs significantly greater than 1. For example, a savings bank uses account numbers as ISN numbers, and the lowest account number is 1,000,001. Specifying MINISN = 1 000 001 stops Adabas from allocating address converter space for ISNs 1-999 999, which would be unused. For more information, see the description of the MAXISN parameter.

MIXDSDEV: Data Storage Mixed Device Types

MIXDSDEV allows the allocation of secondary Data Storage extents on different device types, and therefore with different block lengths. If MIXDSDEV is not specified (the default), Data Storage extents for the specified file must all be on the same device type.

NAME: File Name

NAME is the name to be assigned to the file. This name appears, along with data pertaining to this file, on the Database Status Report produced by the ADAREP utility. The maximum number of characters permitted is 16. The default name assigned is TESTFILE.

If the file name contains special characters or embedded blanks, the name must be enclosed within apostrophes (’...’), which themselves must be doubled if one is included in the name; for example, ’JAN’’S FILE’.

NISIZE: Normal Index Size

NISIZE specifies the number of blocks or cylinders to be assigned to the normal index. A block value must be followed by a "B" (for example, "5500B").

If the specified extent size exceeds the largest free size, ADALOD allocates as many file extents as necessary (up to a total of 5) to satisfy the request.

If the NISIZE parameter is omitted:

- ADALOD determines the space allocation for the normal index based on a sampling of records taken from the input data set. Since this calculation requires additional CPU time and I/O operations, Software AG recommends setting this parameter if the size is known so that no estimation is performed.
- and INDEXCOMPRESSION=YES is set, the index size estimation made by ADALOD does not consider the index compression as it has no knowledge of the rate of compression to be expected. ADALOD may thus allocate a larger index than necessary.

If a small number of records is being loaded and a larger number of records is to be added later, the NISIZE parameter should be set to increase the Normal Index to accommodate the total record amount. For more information, see the section *LOAD File Space Allocation* in the *LOAD Data and Space Requirements* section.

NOACEXTENSION: Limit Address Converter Extents

If NOACEXTENSION is specified, the MAXISN defined for this file cannot be increased in the future. No additional address converter (AC) extents will be created. NOACEXTENSION applies mainly to component files comprising Adabas expanded files; if ANCHOR is specified, NOACEXTENSION must also be specified.

NOUSERABEND: Termination without Abend

When a parameter error or a functional error occurs while this utility function is running, the utility ordinarily prints an error message and terminates with user abend 34 (with a dump) or user abend 35 (without a dump). If NOUSERABEND is specified, the utility will *not* abend after printing the error message. Instead, the message "*utility* TERMINATED DUE TO ERROR CONDITION" is displayed and the utility terminates with condition code 20.

Note:
When NOUSERABEND is specified, we recommend that it be specified as the first parameter of the utility function (before all other parameters). This is necessary to ensure that its parameter error processing occurs properly.

NUMREC: Limit Number of Records to Be Loaded

NUMREC specifies the limit on the number of records to be loaded. If NUMREC is specified, ADALOD stops after processing the specified number of records (unless an end-of-file condition on the input data set ends ADALOD operation before that time). This option is most often used to create a subset of a file for test purposes. If this parameter is omitted, all input records are processed.

If the input data set contains more records than specified by NUMREC, ADALOD processes the number of records specified by NUMREC and then ends with condition code 4.

Note:
If a *LOB file* is being loaded (read about the FILE parameter), the NUMREC parameter cannot be specified in the ADALOD LOAD run.

PGMREFRESH: Program-Generated File Refresh

PGMREFRESH specifies whether a user program is allowed to perform a refresh operation on the file being loaded. If PGMREFRESH is specified, a refresh can be made using an E1 command, or an equivalent call to the nucleus.

READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off for the file. Valid values for this parameter are "YES" (read-only status is on) and "NO" (read-only status is off).

If READONLY is not specified, the default is "NO."

REPLICATOR: Load an Adabas Replicator System File

The REPLICATOR parameter is an Event Replicator for Adabas parameter for an Event Replicator Server. It is a file type parameter. Use this parameter to load the Replicator system file into the Event Replicator Server.

The Replicator system file stores the Event Replicator initialization parameters. When it is loaded into the Event Replicator Server, it can be read during Event Replicator Server startup. You can modify the initialization parameters in the Replicator system file using the Adabas Event Replicator Subsystem. If the Replicator system file cannot be found at startup, the Event Replicator initialization parameters are read from the DDKARTE statements of the Event Replicator Server startup job.

The REPLICATOR parameter may not be specified when loading a file on a database that is not an Event Replicator Server. The contents of DD/EBAND are ignored when loading a Replicator system file. For more information about Adabas system files, read about the FILE parameter of the ADALOD LOAD function, elsewhere in this section.

The REPLICATOR parameter may not be specified in the same ADALOD LOAD as the RPLTARGETID, or any of its associated parameters.

RESTART: Restart Interrupted ADALOD Execution

RESTART forces an interrupted ADALOD run to be restarted, beginning with the last *restart point* reached before the interruption. The *restart point* is the latest point of execution that can be restored from the Temp data set.

If ADALOD is interrupted by a defined error condition, ADALOD issues a message indicating whether or not a restart is possible.

When restarting the ADALOD operation, the following parameters may be changed:

- TEMPSIZE can be increased to make the temp data set larger. Note, however, that the temp data set content contains information necessary for the restart operation, and therefore *must not be changed* ;
- The SORTSIZE and SORTDEV parameters and the sort data set can be changed.

No other parameters can be changed. The DDEBAND/EBAND and DDFILEA/FILEA data sets must remain the same.

RPLDBSI: Before Image for Data Storage

The RPLDSBI parameter is an Event Replicator for Adabas parameter for an Adabas database that turns on the collection of before images of data storage during an update command to the file. Parameter RPLDSBI may only be specified if the RPLTARGETID parameter is also specified. Specify RPLDSBI to turn on the collection of data storage before images during an update. For more information about how this setting is used in Adabas database processing during replication, read *Adabas Nucleus Replication Setup* and *Detailed Adabas Nucleus Processing* in the *Event Replicator for Adabas Concepts* documentation.

The RPLDSBI parameter may not be specified in the same ADALOD LOAD run as the REPLICATOR parameter.

RPLERRORDEACTFILE: Deactivate Replication for File on Error

Use the RPLERRORDEACTFILE parameter to deactivate replication for the file for which replication errors occurred during ADALOD LOAD processing. RPLERRODEACTFILE controls ADALOD LOAD behavior for replication initialization errors as well as other replication errors (in contrast with the RPLINITERROR parameter which affects ADALOD LOAD behavior only for replication initialization errors).

Note:
The RPLERRORDEACTFILE parameter can only be specified if the RPLLOAD parameter is set to "YES" or "FILE".

Valid settings for RPLERRORDEACTFILE are "YES" and "NO":

- Specifying "YES" indicates that replication for the file should be deactivated when a replication error occurs.
- Specifying "NO" (the default) indicates that replication for the file should not be deactivated.

Warning:
If you elect to have ADALOD LOAD continue its processing after a replication error, you are also responsible for recovering your environment appropriately.

Once replication is deactivated for a file it can only be reactivated using Adabas Online System (AOS) or the ADADBS utility.

RPLINITERROR: Replication Control on Error

Use the RPLINITERROR parameter to indicate whether ADALOD processing should continue if replication initialization errors occur.

Note:
The RPLINITERROR parameter can only be specified if the RPLLOAD parameter is set to "YES" or "FILE".

Valid settings for RPLINITERROR are "FAIL" and "CONTINUE":

- Specifying "FAIL" (the default) indicates that ADALOD processing should fail if a replication initialization error occurs. This is the how ADALOD LOAD works now, before the enhancement described in this enhancement preview is applied.
- Specifying "CONTINUE" indicates that ADALOD LOAD processing should continue if a replication initialization error occurs.

Warning:
If you elect to have ADALOD LOAD continue its processing after a replication error, you are also responsible for recovering your environment appropriately.

RPLKEY: Primary Key for Replication

The RPLKEY parameter is an Event Replicator for Adabas parameter for an Adabas database that specifies the primary key for replication. This parameter may only be specified if the RPLTARGETID parameter is also specified. For more information about how this primary key is used in Adabas database processing during replication, read *Adabas Nucleus Replication Setup* and *Detailed Adabas Nucleus Processing* in the *Event Replicator for Adabas Concepts* documentation.

The RPLKEY parameter may not be specified in the same ADALOD LOAD run as the REPLICATOR parameter.

RPLLOAD: Replicate Load Data

The ADALOD LOAD RPLLOAD parameter is an Event Replicator for Adabas parameter that can be specified for the Adabas database files when using replication. It is only allowed for ADALOD LOAD if replication is already turned on for a database.

Note:
The version of Event Replicator specified in an ADALOD LOAD utility job that specifies RPLLOAD=YES must match the version of Event Replicator used by the Event Replicator Server. In addition, the Adabas version used by the Event Replicator Server must be greater than or equal to the

Adabas version used in an ADALOD LOAD utility job that specifies RPLLOAD=YES.

The RPLLOAD parameter indicates whether or not data, and possibly the FCB/FDT, loaded to the Adabas database via the ADALOD LOAD utility will be replicated to the Event Replicator Server. Valid values are "YES", "FILE", and "NO"; the default is "NO":

- When RPLLOAD=YES is specified, ADALOD LOAD replicates data to the Event Replicator Server that it loads to the Adabas database.
- When RPLLOAD=FILE is specified, ADALOD LOAD replicates the FCB/FDT and data to the Event Replicator Server. When the data is replicated to an Adabas destination, the file is first allocated on the target database. The DESTINATION parameters DREPLICATEUTI for the target destination and DAREPLICATEUTI for the target file must be set to YES to use this option.
- When RPLLOAD=NO is specified, ADALOD LOAD does not replicate its load data to the Event Replicator Server.

Warning:
If ADALOD LOAD ends abnormally (due to insufficient space, for example), updates made to the file before the abnormal ending cannot be backed out; there is no automated recovery for the updated data or for the replicated data. Software AG therefore recommends that you perform an ADASAV SAVE on the file before you run ADALOD LOAD.

RPLTARGETID: Replication Target ID

The RPLTARGETID parameter is an Event Replicator for Adabas parameter for an Adabas database that specifies the Event Replicator target ID used when the Adabas file data is replicated.

The RPLTARGETID parameter may not be specified in the same ADALOD LOAD as the REPLICATOR parameter. For more information about how this target ID is used in Adabas database processing during replication, read *Adabas Nucleus Replication Setup* in the *Event Replicator for Adabas Concepts* documentation.

Note:
Replication may not be turned on for an Adabas system file, a ciphered file, or a file that allows spanned data storage records (i.e. a file compressed with the ADACMP COMPRESS option SPAN).

RPLUPDATEONLY: Allow Only Event Replicator Processing Updates

The RPLUPDATEONLY parameter is an Event Replicator for Adabas parameter for an Adabas database that can be used in the ADALOD LOAD function to indicate whether an Adabas database file may be updated only by the Event Replicator Server as part of Adabas-to-Adabas replication or by other means as well. This parameter is optional. Valid values are "YES" or "NO". A value of "YES" indicates that the file can only be updated via Event Replicator processing; a value of NO indicates that the file can be updated by any normal means, including Event Replicator processing.

If the file is a new file, the default for this parameter is "NO".

However, if the file specified in the ADALOD LOAD function is an existing file, there is no default for this parameter. If no value is specified for the RPLUPDATEONLY parameter in the ADALOD LOAD function for an existing file, the value used previously for the file is used.

SKIPREC: Number of Records to Be Skipped

SKIPREC specifies the number of input records to be skipped before beginning load processing. The default is 0 (no records are skipped).

Note:
If a *LOB file* is being loaded (read about the FILE parameter), the SKIPREC parameter cannot be specified in the ADALOD LOAD run.

SLOG: Load an Adabas SLOG File

The SLOG parameter is an Event Replicator for Adabas parameter for an Event Replicator Server. It is a file type parameter. Use this parameter to load the SLOG system file into the Event Replicator Server.

An SLOG file is an Adabas system file used only by the Event Replicator Server to store subscription logging data. The SLOG parameter may not be specified when loading a file on a database that is not an Event Replicator Server. The contents of DD/EBAND are ignored when loading an SLOG file.

SORTDEV: Sort Device Type

ADALOD uses the sort data set to sort descriptor values. The SORTDEV parameter indicates the device type to be used for this data set. This parameter is required only if the device type to be used is different from that specified by the ADARUN DEVICE parameter.

SYFMAXUV: Maximum MU System Field Values

The SYFMAXUV parameter can be used to specify the maximum number of values kept for a system field with the MU option during the execution of an update (A1) command (in other words, the maximum number of occurrences allowed for MU system fields during the execution of an update command). The value set for SYFMAXUV applies to all system fields in the file with the MU option. Valid values are integers from 1 through 20. The maximum value for SYFMAXUV is 20.

The internal default, if SYFMAXUV is not specified, is zero (0), which Adabas interprets to mean that there is no setting for this parameter at the file level. In this case, Adabas will assume a default of 1.

TEMPDEV: Temporary Storage Device Type

ADALOD uses the temp data set to store intermediate data. The TEMPDEV parameter indicates the device type to be used for this data set. This parameter is required only if the device type to be used is different from that specified by the ADARUN DEVICE parameter.

TEST: Test Syntax

The TEST parameter tests the operation syntax without actually performing the operation. Only the syntax of the specified parameters can be tested; not the validity of values and variables.

UISIZE: Upper Index Size

UISIZE specifies the number of blocks or cylinders to be assigned to the upper index. A block value must be followed by a "B" (for example, "5500B").

If the specified extent size exceeds the largest free size, ADALOD allocates as many file extents as necessary (up to a total of 5) to satisfy the request.

If the UISIZE parameter is omitted:

- ADALOD determines the space allocation for the upper index based on a sampling of records taken from the input data set. Since this calculation requires additional CPU time and I/O operations, Software AG recommends setting this parameter if the size is known so that no estimation is performed.
- and INDEXCOMPRESSION=YES is set, the index size estimation made by ADALOD does not consider the index compression as it has no knowledge of the rate of compression to be expected. ADALOD may thus allocate a larger index than necessary.

If a small number of records is being loaded and a larger number of records are to be added later, the UISIZE parameter should be set to increase the upper index to accommodate the total record amount. For more information, see the section *LOAD File Space Allocation* in the *LOAD Data and Space Requirements* section.

UQDE: Unique Descriptors

UQDE defines one or more descriptors as unique. Each descriptor specified must contain a different value in each input record. If a non-unique value is detected during ADALOD processing, ADALOD terminates with an error message.

If the unique descriptor (UQ) option was specified with the ADACMP utility, the UQDE parameter here is not required.

Adabas prevents a descriptor defined with the unique descriptor (UQ) option from being updated with an add or update command if the update would cause a duplicate value for the descriptor.

Note:
For Adabas expanded files, ADALOD can only detect unique descriptor violations within the component file. If an identical value exists for a unique descriptor in one of the other component files, ADALOD cannot detect it. You must therefore ensure that unique descriptor values remain unique throughout an expanded file.

Note:
If a *LOB file* is being loaded (read about the FILE parameter), the UQDE parameter cannot be specified in the ADALOD LOAD run.

USERISN: User ISN Assignment

USERISN=YES indicates that the USERISN option for the loaded file is to be in effect, and that the ISN for each new record is being supplied by the user in the input data. If USERISN=NO, Adabas assigns the ISN for each new record.

If USERISN is not specified, a default setting is assumed that depends on the input file itself. If the input file was created by ADACMP with the USERISN option or by ADAULD from a file having the USERISN option, the default for ADALOD operation is USERISN=YES; otherwise, the default is USERISN=NO. Specifying USERISN here overrides the existing default value.

Note:
Adabas 5.2 files initially loaded with the USERISN option do not require USERISN=YES to again be specified when the files are reloaded; ADALOD assumes the default as described above. However, Adabas 5.1 files initially loaded with the USERISN option *must* have USERISN=YES specified whenever they are reloaded.

VERSION: Input Data Format

Originally, this parameter specified the Adabas version of the output (ADACMP) data sets to be loaded into Adabas.

Because ADALOD determines the version of the sequential input data set itself, this parameter is ignored. It is available only for compatibility with old ADALOD jobs.

Examples

Example 1:

```
ADALOD  LOAD FILE=6,MAXISN=20000,DSSIZE=20,ASSOPFAC=15,
ADALOD       DATAPFAC=15,TEMPSIZE=20,SORTSIZE=10
```

File 6 is to be loaded. The number of records initially permitted for the file is 20,000. 20 cylinders are to be allocated for Data Storage. The Associator and Data Storage block padding factors are both 15 percent. The temp and sort data sets are 20 and 10 cylinders, respectively.

Example 2:

```
ADALOD  LOAD FILE=7,MAXISN=350000,ASSOPFAC=5,MINISN=100001
ADALOD       DATAPFAC=15,DSSIZE=100,USERISN=YES
ADALOD       TEMPSIZE=200,SORTSIZE=100
```

File 7 is to be loaded. The number of records initially allocated for the file is 250,000, and the minimum is 100,001. The Associator padding factor is 5 percent. The Data Storage padding factor is 15 percent. 100 cylinders are to be allocated for Data Storage. ISNs are contained in the input. The temp and sort data sets are equal to 200 and 100 cylinders, respectively.

Example 3:

```
ADALOD  LOAD FILE=8,ADAMFILE,ADAMDE='AK'
ADALOD       ADAMPARM=4,ADAMOFLOW=5,UQDE='AK',MINISN=1
ADALOD       MAXISN=10000,DSSIZE=20,ASSOPFAC=5,DATAPFAC=5
ADALOD       TEMPSIZE=10,SORTSIZE=5
```

File 8 is to be loaded as an ADAM file. Field AK is the ADAM key. 4 bits are to be truncated from each value of AK before using the value to calculate the Data Storage RABN for the record. The size of the ADAM overflow area is 5 cylinders. The field AK is defined as a unique descriptor. The maximum number of records initially allocated for the file is 10,000. Twenty (20) cylinders are to be allocated to Data Storage, from which the five ADAM overflow cylinders are taken. The padding factor for both the Associator and Data Storage is five percent. The sizes of the Temp and Sort data sets are ten and five cylinders, respectively.

Example 4:

```
ADALOD  LOAD FILE=9,NAME=INVENTORY,MAXISN=5000
ADALOD       DSSIZE=2000B,DSRABN=30629,NISIZE=300B,UISIZE=50B
ADALOD       MAXDS=1000B,MAXNI=50B,MAXUI=1B
ADALOD       INDEXCOMPRESSION=YES
ADALOD       ASSOPFAC=20,DATAPFAC=10
ADALOD       TEMPSIZE=10,SORTSIZE=5,UQDE='U1,U2'
```

File 9 is to be loaded. The text name for the file is INVENTORY. The initial space allocation for the file is for 5,000 records. 2,000 blocks are to be allocated for Data Storage, beginning with RABN 30,629. 300 blocks are to be allocated for the normal index. 50 blocks are to be allocated to the upper index. The maximum allocations per secondary extent for Data Storage, normal index and upper index are 1000 blocks, 50 blocks, and 1 block respectively. The index is to be compressed. The padding factor for the Associator is 20 percent. The padding factor for Data Storage is 10 percent. The sizes of the temp and sort data sets are 10 and 5 cylinders respectively. Descriptors U1 and U2 are defined as unique descriptors.

Example 5:

```
ADALOD  LOAD FILE=2,SECURITY
ADALOD       DSSIZE=20B,MAXISN=2000,NISIZE=20B,UISIZE=5B
ADALOD       TEMPSIZE=10,SORTSIZE=5
```

File 2 is to be loaded as an Adabas security file. The DDEBAND contents are ignored. Space is allocated for Data Storage (20 blocks), for the address converter (2000 ISNs), the normal index (20 blocks), and the upper index (5 blocks). The temp size is 10 cylinders, and the sort area size is 5 cylinders.

LOAD Data and Space Requirements

The following general information describes data requirements for LOAD operation, and how ADALOD LOAD allocates space. For more information about space allocation, refer to the *Adabas DBA Reference* documentation.

Input Data for LOAD Operations

Compressed data records produced by the ADACMP or ADAULD utility may be used as input to ADALOD. If output from an ADAULD utility run made with the MODE=SHORT option is used as ADALOD input, any descriptor information will be removed from the FDT, and no index will exist for the file.

LOAD File Space Allocation

ADALOD allocates space for the normal index (NI), upper index (UI), address converter (AC), Data Storage, and the temp area for the file being loaded.

Index Space Allocation

If the NISIZE and/or the UISIZE parameters are supplied, allocation is made using the user-supplied values. If these parameters are not supplied, ADALOD allocates space for these indexes based on a sampling of the values present for each descriptor.

Descriptor values are sampled as follows:

1. ADALOD reads the compressed input, stores the records into Data Storage, extracts each value for each descriptor and writes these values to the temp data set. Each temp block contains values for one descriptor only. At the end of this processing phase, the following information is present:

   - number of values for each descriptor
   - number of bytes required for each descriptor
   - temp RABNs used for each descriptor

   For unique descriptors, the NI space requirement is equal to the temp size used. For non-unique descriptors, the number of duplicate values must be determined. Each duplicate value's space requirement must be estimated and then subtracted from the number of bytes required. The result is the NI size required for the duplicate descriptor.

   The number of duplicate values is determined by reading up to 16 temp blocks containing values for a single descriptor. These values are sorted to determine how many are duplicates. The resulting count of duplicate values is multiplied by the factor:

   $$\frac{\text{total number of values}}{\text{number of values in the sample}}$$

   The result is the estimated number of identical descriptor values present in the entire file for this descriptor. This space requirement is subtracted from the temp size estimate.

2. The upper index (UI) size is computed after all normal index (NI) and temp sizes are available.

3. The NI and UI sizes are each multiplied by the result of:

   $$\frac{\text{(MAXISN - MINISN) + 1}}{\text{number of records being loaded}}$$

For example, if 10000 records require 10 blocks of UI space and 500 blocks of NI space with MINISN = 1 (the default), the specification of MAXISN = 60000 causes 60 UI blocks and 3000 NI blocks to be allocated:

$$10 \times \frac{60000}{10000} = 60 \text{ UI Blocks}$$

$$500 \times \frac{60000}{10000} = 3000 \text{ NI Blocks}$$

However, this calculation is not made if USERISN=YES is in effect.

By setting MAXISN appropriately, it is therefore possible to increase the size allocation for files in which a small number of records are being loaded and for which a much larger number of records are to be added subsequently.

If the NISIZE and UISIZE parameters have been specified, the space allocation is made using unassigned Associator RABNs. If the NIRABN and/or the UIRABN parameters are supplied, space allocation is made at the user-specified RABN.

Address Converter Space Allocation

The address converter allocation is based on the MAXISN and MINISN values for the file. ADALOD allocates the blocks needed to contain the number of bytes calculated by the formula:

```
RABNSIZE x((MAXISN - MINISN)+ 1)
```

If the ACRABN parameter has been specified, ADALOD allocates the address converter beginning with the user-specified block number; otherwise, it uses unassigned Associator RABNs.

Data Storage Space Allocation

Data Storage allocation is based upon the value specified with the DSSIZE parameter. If the DSRABN parameter has been specified, the allocation is made beginning with the user-specified block number; otherwise, unassigned Data Storage RABNs are used.

If there are different device types in the database, Data Storage allocation can be forced on a specified device type by specifying DSDEV. The MIXDSDEV parameter permits Data Storage allocation on different device types, assuming the device types can store records with the length specified by MAXRECL.

Temp Area Space Allocation

For each descriptor, ADALOD generates a list of the values and ISNs of the records containing the value, and writes this information to the Temp data set. The space required for descriptor information is equal to the sum of the space required for each descriptor. The space needed for each descriptor can be calculated using the following formula:

```
SP = N x  NPE x  NMU x  (L + 4)
```

where

SP	is the space required for the descriptor (in bytes).
N	is the number of records being loaded.
NPE	is the average number of occurrences, if the descriptor is contained in a periodic group. If not in a periodic group, NPE equals 1.
NMU	is the average number of occurrences, if the descriptor is a multiple-value field. If not a multiple-value field, NMU equals 1.
L	is the average length (after compression) of each value for the descriptor.

Example:

A file containing 20,000 records is being loaded. The file contains two descriptors (AA and CC). Descriptor AA has 1 value in each record and the average compressed value length is 3 bytes. Descriptor CC has an average of 10 values in each record and the average compressed value length is equal to 4 bytes.

Field Definitions:

```
01,AA,5,U,DE
01,CC,12,A,DE,MU
```

- Space requirement for AA.

```
SP = 20,000 •  1 •  (3 + 4)
SP = 140,000 bytes
```

- Space requirement for CC.

```
SP = 20,000 •  10 •  (4 + 4)
SP = 1,600,000 bytes
```

- Total space requirement = *1,740,000 bytes* .

The number of cylinders required may be calculated by dividing the number of blocks required by the number of blocks per cylinder.

For a model 3380 device type:

$$\text{Blocks required} = \frac{1{,}740{,}000}{7476 \text{ bytes per block}} = 232 +, \text{ or } 233 \text{ blocks}$$

$$\text{Cylinders required} = \frac{233 \text{ blocks}}{90 \text{ blocks per cylinder}} = 2 +, \text{ or } 3 \text{ cylinders}$$

Associator Updating by LOAD

ADALOD then sorts the descriptor values collected in the input phase and enters the sorted values into the normal index and upper index. If the allocated index space is not enough for the normal index or upper index, ADALOD allocates up to four additional extents.

Each additional extent allocated is equal to about 25 percent of the total current space allocated to the index. If insufficient space is available for the additional extent or if the maximum number of allocated extents has been reached, ADALOD terminates with an error message.

Loading Expanded Files

An expanded file is made up of a series of normal Adabas physical files. The number sequence of the files within the expanded file is arbitrary. The first file may be file 53; the second, file 127; the third, 13, and so on. ISNs assigned to each component file must be unique; no two files can contain the same ISN. The ISN range over all files must be in ascending order; however, there can be gaps in the sequence.

The total number of records in an expanded-file chain cannot exceed 4,294,967,294.

The sequence of physical component files that build an expanded logical file is defined by the ANCHOR parameter, which defines the first component file (anchor) in the sequence. The anchor file is loaded just as any other Adabas file; each additional component file must be loaded with the ANCHOR parameter referring to the anchor file. ADALOD inserts the new physical file into the existing expanded file chain according to the range of ISNs assigned to the added file. Each added component file must also specify the NOACEXTENSION parameter when being loaded to prevent Adabas from assigning new ISNs to a component file.

ADALOD processes only the anchor file and the single physical (component) files that compose an expanded file, and not the complete expanded file itself.

Loading Data into an Expanded File

To load data (for example, several million records) into different physical files, the input data must first be divided into several DDEBAND/EBAND input files. The DDEBAND/EBAND file data may be mapped into the component files using the SKIPREC and NUMREC parameters; however, one-to-one mapping without skipping or limits is recommended. This avoids the need to read records that will not be used later, and thus improves performance.

Examples:

The following examples, which show parts of one or more jobs for loading an expanded file, illustrate the mapping of DDEBAND/EBAND file data into component files:

```
//DDEBAND    DD DSN=LOAD.DATA.FILE1,...
//DDKARTE    DD *
ADALOD LOAD FILE=40,NAME='XXX_Part1'
ADALOD      MINISN=1,MAXISN=10000000,NOACEXTENSION
ADALOD      NUMREC=10000000
ADALOD        DSSIZE=...,NISIZE=...,UISIZE...
ADALOD        SORTSIZE=...,TEMPSIZE=...
            .
            .

//DDEBAND    DD DSN=LOAD.DATA.FILE1,...
//DDKARTE    DD *
ADALOD LOAD FILE=41,NAME='XXX_Part2',ANCHOR=40
ADALOD      MINISN=10000001,MAXISN=20000000,NOACEXTENSION
ADALOD      NUMREC=10000000,SKIPREC=10000000
ADALOD        DSSIZE=...,NISIZE=...,UISIZE...
ADALOD        SORTSIZE=...,TEMPSIZE=...
            .
            .

//DDEBAND    DD DSN=LOAD.DATA.FILE2,...
//DDKARTE    DD *
ADALOD LOAD FILE=35,NAME='XXX_Part2',ANCHOR=40
ADALOD      MINISN=20000001,MAXISN=30000000,NOACEXTENSION
ADALOD      NUMREC=10000000
ADALOD        DSSIZE=...,NISIZE=...,UISIZE...
ADALOD        SORTSIZE=...,TEMPSIZE=...
            .
            .
```

Loading Multiclient Files

Note:
A multiclient file cannot be made part of an expanded file, and an expanded file cannot be converted to a multiclient file.

A multiclient file stores records for multiple users or groups of users. It divides the physical file into multiple logical files by attaching an owner ID to each record. Each user can access only the subset of records that is associated with the user's owner ID.

For any installed external security package such as RACF or CA-Top Secret, a user is still identified by either Natural ETID or LOGON ID. The owner ID is assigned to a user ID. A user ID can have only one owner ID, but an owner ID can belong to more than one user.

The ADALOD LOAD function uses the LOWNERID and ETID parameters to support the migration of an application from a standard to a multiclient environment. The parameters work together to define owner IDs and determine whether a file is a multiclient file.

LOWNERID specifies the length of the internal owner ID values assigned to each record for multiclient files. In combination with the ETID parameter, the LOWNERID parameter can be used to reload a standard file as a multiclient file, change the length of the owner ID for the file, or remove the owner ID from the records of a file.

If the LOWNERID parameter is not specified, the length of the owner ID for the input file (if any) remains the same.

ETID assigns a new owner ID to all records being loaded into a multiclient file, and must be specified if the input file contains no owner IDs; that is, the input file was not unloaded from a multiclient source file.

Examples of Loading/Updating Multiclient Files

```
ADALOD LOAD FILE=20,LOWNERID=2,NUMREC=0
```

Creates file 20 as a multiclient file. The length of the internal owner ID is two bytes, but no actual owner ID (ETID) is specified. No records are actually loaded in the file (NUMREC=0).

```
ADALOD LOAD FILE=20,LOWNERID=2,ETID=USER1
```

Creates file 20 as a multiclient file, load all supplied records, and assign them to user USER1. The length of the internal owner ID is two bytes.

```
ADALOD UPDATE FILE=20,ETID=USER2
```

Performs a mass update to add records to file 20, a multiclient file. Load all the new records and assign them to USER2.