# STORE: Store Files

The STORE function loads one or more files into an existing database using output produced by the RESTRUCTURE functions. The Associator and Data Storage are reordered as part of RESTRUCTURE/STORE processing.

If the ALLFILES parameter is specified, all files contained on the input data set are stored. If ALLFILES is not specified, only those files specified by FILE parameters are stored.

One or more files may be specified with FILE parameter statements, even when ALLFILES is also specified. The STORE function loads each file specified with a FILE statement according to the definition contained in any subparameters immediately following that file's FILE statement. All other files are loaded according to their existing definitions.

If existing files in the database are to be overwritten, the OVERWRITE parameter must be supplied.

This function requires exclusive EXF control of the database files involved in the operation. In addition, parts of the database are overwritten during ADAORD execution, so we recommend that you back up the database (or file) using the ADASAV utility first, before running ADAORD functions.

If the file specified for this function was originally loaded with ISNREUSE=YES active, this store function will reset the first unused ISN value in that file's control block (FCB) to the actual first unused ISN found in the address converter.

**Notes:**

1. The STORE function does not reorder ADAM files. However, it, in conjunction with other ADAORD functions, can be used to relocate an ADAM file to different RABNs.
2. Storing restructured ADAM files on a device with a smaller DATA blocking factor than before can result in utility ERROR 103 if the ADAM file previously used the overflow area. To relocate an ADAM file to a different device, use the ADAULD and ADALOD utilities.
3. Checkpoint and security files from Adabas version 5.1 or 5.2 cannot be stored due to internal structure changes to the files in version 5.3.
4. If ALLFILES is requested in the STORE function, be sure that the Associator (ASSO) data set is large enough to accommodate the all the files. Use the calculation `(file-count * 3) + 110` to verify this. Otherwise, an ERROR-068 is possible.
5. You can restructure databases and files from an Adabas version prior to Adabas 8 and store them in an Adabas 8 database using ADAORD STORE. However, you cannot store the restructured output of an Adabas 8 database or file in a database running with any prior Adabas version (for example, Adabas 7). If you attempt this, the following warning will be generated and ADAORD will end with a CC=4: `*** Warning: The input data set is from V8 and will not be processed`
6. Logically deleted field data in the file is loaded by the ADAORD STORE utility function.

This is the syntax of the ADAORD STORE function:

```
ADAORD STORE  [ALLFILES]
              [CHECKPOINT]
              [FILE = file-number ]
                 [ACRABN = starting-rabn ]
                 [AC2RABN = starting-rabn ]
                 [ALLOCATION = { FORCE | NOFORCE } ]
                 [ASSOPFAC = padding-factor ]
                 [ASSOVOLUME = ' Associator-extent-volume ' ]
                 [DATAPFAC = padding-factor ]
                 [DATAVOLUME = ' Data-Storage-extent-volume ' ]
                 [DSDEV = device-type ]
                 [DSRABN = starting-rabn ]
                 [DSRELEASE]
                 [DSSIZE = size ]
                 [INDEXCOMPRESSION = { YES | NO } ]
                 [ISNSIZE = { 3 | 4 } ]
                 [MAXISN = highest-isn ]
                 [MAXISN2 = highest-isn ]
                 [MAXRECL = record-length ]
                 [NIRABN = starting-rabn ]
                 [NIRELEASE]
                 [NISIZE = size ]
                 [READONLY = { YES | NO } ]
                 [RPLDSBI = { YES | NO } ]
                 [RPLKEY = { primary-key | OFF } ]
                 [RPLTARGETID = { target-ID | OFF | 0 } ]
                 [RPLUPDATEONLY = { YES | NO } ]
                 [UIRABN = starting-rabn ]
                 [UIRELEASE]
                 [UISIZE = size ]
              [EXCLUDE = file-list ]
              [LIP = { isn-pool-size | 16384 } ]
              [NOUSERABEND]
              [OVERWRITE]
              [REPLICATOR]
              [SECURITY]
              [SLOG]
              [TEST]
              [TRIGGER]
```

This chapter covers the following topics:

- Optional Parameters and Subparameters
- Examples

---

# Optional Parameters and Subparameters

### ACRABN: Starting RABN for Address Converter

ACRABN specifies the beginning RABN for the file's address converter extent. If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent.

When specifying the starting RABN for Associator extents, the space needed for the FCBs, FDTs, and DSST should also be considered.

### AC2RABN: Starting RABN for Secondary Address Converter

The beginning RABN for the file's secondary address converter extent. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

If this parameter is omitted, ADAORD assigns the starting RABN. The space requested must be available in one extent. If the file contains no secondary address converter extents, this parameter is ignored.

### ALLFILES: Select All Files for Storing

ALLFILES causes all files in the input data set to be stored in the database. If ALLFILES is not supplied, only those files are stored for which FILE parameters have been specified.

If the input data set contains files that are coupled or part of an expanded file and the related files are not in the data set, ERROR-138 is returned indicating an inconsistent file list. You must add the related files before the STORE function will execute successfully.

**Note:**
If ALLFILES is requested in the STORE function, be sure that the Associator (ASSO) data set is large enough to accommodate the all the files. Use the calculation `(file-count * 3) + 110` to verify this. Otherwise, an ERROR-068 is possible.

### 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.

### ASSOPFAC: Associator Padding Factor

ASSOPFAC specifies the new Associator block padding factor. The number specified represents the percentage of each Associator block not to be used during the reorder process. A value in the range 1-90 may be specified. The remaining number of bytes after padding must be greater than the largest descriptor value plus 10.

If this parameter is omitted, the current Associator padding factor in effect for the file is used.

**ASSOVOLUME: Associator Extent Volume**

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

ASSOVOLUME identifies the volume on which to allocate the file's Associator space (the AC, NI, and UI extents). If the requested number of blocks cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If ACRABN, UIRABN, or NIRABN is specified, ADAORD ignores the ASSOVOLUME value when allocating the corresponding extent type.

If ASSOVOLUME is not specified, the file's Associator space is allocated according to ADAORD's default allocation rules.

**CHECKPOINT: Store the Checkpoint File**

If either ALLFILES is specified or the FILE parameter specifies the checkpoint file, CHECKPOINT stores the checkpoint file from the DDFILEA/FILEA tape in the database, making that file the new checkpoint file. The new checkpoint file must have the same file number as the old checkpoint file.

If the CHECKPOINT parameter is not specified, the checkpoint file on the FILEA/DDFILEA tape is not stored in the database, even though the checkpoint file was specified by a FILE parameter or the ALLFILES parameter was specified.

**DATAPFAC: Data Storage Padding Factor**

DATAPFAC specifies the new Data Storage padding factor, which is the percentage of each Data Storage block reserved for record expansion when the file is reordered. A value in the range 1-90 may be specified (see the ADALOD LOAD DATAPFAC parameter discussion for more information about setting the padding factor). If this parameter is omitted, the current padding factor for the file is used.

**DATAVOLUME: Data Storage Extent Volume**

**Note:**
The value of the DATAVOLUME parameter must be enclosed in apostrophes.

DATAVOLUME specifies the volume on which the file's Data Storage space (DS extents) are allocated. If the number of blocks requested with DSSIZE cannot be found on the specified volume, ADAORD allocates the remaining blocks on other volumes according to its default allocation rules.

If DSRABN is specified, DATAVOLUME is ignored for the related file.

If DATAVOLUME is not specified, the Data Storage space is allocated according to ADAORD's default allocation rules.

**DSDEV: Data Storage Device Type**

DSDEV specifies the device type to be used for the file's Data Storage. The specified device type must already be defined to Adabas, normally when the database was created or by the ADADBS utility's ADD function.

If this parameter is not specified, ADAORD attempts to allocate the file on the device type used before restructuring.

**DSRABN: Data Storage Starting RABN**

The beginning RABN for the Data Storage extent for the specified file. If this parameter is omitted, ADAORD assigns the starting RABN.

**DSRELEASE: Release Unused Data Storage Blocks**

Specifying DSRELEASE releases unused Data Storage (DS) blocks belonging to the specified file. If DSRELEASE is not specified, ADAORD allocates *at least* the number of DS blocks that were allocated before the file was reordered.

**Note:**
Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

**DSSIZE: Data Storage Size**

DSSIZE specifies the number of blocks or cylinders to be allocated for the Data Storage. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD will compute the file extent size (in blocks) in proportion to an increase or decrease in the DATAPFAC padding factor used.

**EXCLUDE: Exclude Specified Files from Store**

EXCLUDE lists the numbers of the files to be excluded from STORE processing; that is, the files that are not to be stored.

The parameter is optional: if not specified, no files are excluded. A file number may be listed only once.

The EXCLUDE parameter may be specified only if ALLFILES is also specified.

The EXCLUDE parameter is provided for use in recovery jobs built by the Adabas Recovery Aid (ADARAI).

**FILE: File Number**

FILE specifies the file to be stored. A separate statement must be provided for each file to be processed, followed by ADAORD statements containing the relevant parameters for that file.

If you specify a file that is coupled or part of an expanded file and you do not also specify the related files, ERROR-138 is returned indicating an inconsistent file list. You must add the related files before the STORE function will execute successfully.

### INDEXCOMPRESSION: Compress File Index

INDEXCOMPRESSION indicates whether the index for the file is rebuilt 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, the default is the form of the file at the time of the corresponding restructure operation.

### ISNSIZE: 3- or 4-Byte ISN

ISNSIZE specifies whether ISNs in the file are to be 3 or 4 bytes long. The default is the value currently used for the file; this value is stored in the file control block (FCB).

**Note:**
It is not possible to change the ISNSIZE of a physically coupled file using ADAORD.

### LIP: ISN Buffer Pool Size

The LIP parameter can be used to decrease the number of Associator I/O operations when recreating the address converter. For best performance, specify a size that accepts all ISNs of the largest file to be processed.

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 16384 bytes (or 16K).

The length of one input record is ISNSIZE + RABNSIZE. Thus the entry length is at least 6 bytes (the ISNSIZE of the file is 3 and the RABNSIZE of the database is 3) and at most 8 bytes (the ISNSIZE is 4 and the RABNSIZE is 4).

**Note:**
When ADAORD is processing files that contain spanned records with secondary ISNs, a second LIP will be allocated to contain these ISNs.

### MAXISN: Highest ISN Permitted for the File

MAXISN specifies the highest ISN which may be allocated for the file. This value must be greater than the current TOPISN value displayed in the ADAREP database report.

ADAORD uses the specified value to calculate the address converter space required. If this parameter is omitted, the current MAXISN value for the file is retained.

### MAXISN2: Highest Secondary ISN Permitted for the File

MAXISN specifies the desired size of the secondary address converter (AC2) in ISNs. This value must be greater than the current TOP AC2 ISN value displayed in the ADAREP database report. The secondary address converter is used to map the secondary ISNs of secondary spanned records to the RABNs of the Data Storage blocks where the secondary records are stored.

ADAORD uses the specified value to calculate the space required in the secondary address converter for the file. If this parameter is omitted, the current MAXISN2 value for the file is retained. If the file contains no secondary address converter extents, this parameter is ignored.

### MAXRECL: Maximum Compressed Record Length

Use the MAXRECL parameter to change the maximum record length, after compression, permitted in the file. Specifying MAXRECL has two effects:

- The DATA data set for the file can be allocated only to devices that support the specified length.
- If the file contains Data Storage records that exceed the specified length, ADAORD abends and prints ERROR-126 (Data Storage record too long).

If MAXRECL is not specified, there are two possibilities for the default value:

- If the maximum compressed record length before the file was restructured was the default ADALOD MAXRECL value, then DATA is allocated to a device arbitrarily, and the new maximum record length is derived from the device type;
- Otherwise, the maximum compressed record length does not change.

### NIRABN: Starting RABN for Normal Index

The beginning RABN for the file's normal index extent. If this parameter is omitted, ADAORD assigns the starting RABN.

### NIRELEASE: Release Unused Normal Index Blocks

Specifying NIRELEASE releases unused normal index (NI) blocks belonging to the file. If NIRELEASE is not specified, ADAORD allocates *at least* the number of NI blocks that were allocated before the file was reordered.

**Note:**
Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

### NISIZE: Normal Index Size

The number of blocks or cylinders to be allocated for the normal index. The specified value cannot be larger than the largest single contiguous RABN area available; specifying blocks (a number of blocks followed by a "B") is therefore recommended.

If this parameter is omitted, ADAORD computes the file extent size (in blocks) in proportion to an increase or decrease in the ASSOPFAC padding factor used.

If this parameter is omitted and the INDEXCOMPRESSION parameter is specified, the ADAORD index size calculation does not consider the change in index size because ADAORD has no knowledge of the compression rate to be expected. Thus, ADAORD may allocate an index smaller than required causing secondary index extent allocations; or larger than necessary.

### 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.

### OVERWRITE: Overwrite Existing File

If a file to be stored already exists in the database, ADAORD terminates with an error message unless the OVERWRITE parameter has been specified.

### READONLY: Read-only Status Indicator

READONLY indicates whether the read-only status is on or off. 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 status of the file at the time of the corresponding restructure operation is used.

### REPLICATOR: Store the Replicator System File

Use this parameter to store the Replicator system file from the DDFILEA/FILEA tape as the new Replicator system file for the Event Replicator Server. The new Replicator system file must have the same file number as the old Replicator system file. This parameter can only be specified if the database is an Event Replicator Server.

When the REPLICATOR parameter is not specified, the Replicator system file on the DDFILEA/FILEA tape is not stored in the database, even if it is specified by a FILE or ALLFILES parameter.

### RPLDSBI: Before Image for Data Storage

The RPLDSBI parameter indicates whether the collection of before images of data storage during an update command to the file should be done if replication is turned on for the file.

Parameter RPLDSBI may only be specified if replication is turned on for the file.

If RPLDSBI is not specified, the default value depends on the target database ID (DBID) of the RESTORE and its replication state; if the ADAORD STORE DBID is the same as the original RESTRUCTURE DBID and REPLICATION=YES in the target DBID, the original RPLDSBI value is used. Otherwise, RPLDSBI is set to NO.

### RPLKEY: Primary Key for Replication

The RPLKEY parameter specifies the primary key for replication. The format is RPLKEY=’*primary-key-for-replication*’. When a valid descriptor name is specified for RPLKEY, the specified descriptor is used as the primary replication key for the corresponding file. When OFF is specified for RPLKEY, no primary key is used for replication (RPLKEY is turned off) for the corresponding file.

This parameter may only be specified if replication is turned on for the file and if the provided descriptor name is a valid Adabas descriptor.

If RPLKEY is not specified, the default value depends on the target database ID (DBID) of the RESTORE and its replication state; if the ADAORD STORE DBID is the same as the original RESTRUCTURE DBID and REPLICATION=YES in the target DBID, the original RPLKEY value is used. Otherwise, RPLDSBI is set to OFF.

**RPLTARGETID: Replication Target ID**

The RPLTARGETID parameter specifies the Event Replicator target ID used when the Adabas file data is replicated. The format is RPLTARGETID='*reptor-target-id*'. When a valid target ID is specified for RPLTARGETID, the specified target is used as the target for replicated data for the corresponding file. When OFF or 0 (zero) are specified for RPLTARGETID, no target is used for replication for the corresponding file.

If RPLTARGETID is not specified, the default value depends on the target database ID (DBID) of the RESTORE and its replication state; if the ADAORD STORE DBID is the same as the original RESTRUCTURE DBID and REPLICATION=YES in the target DBID, the original RPLTARGETID value is used. Otherwise, RPLTARGETID is set to OFF.

**RPLUPDATEONLY: Allow Only Event Replicator Processing Updates**

The RPLUPDATEONLY parameter can be used in the ADAORD STORE 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 ADAORD STORE function is an existing file, there is no default for this parameter. If no value is specified for the RPLUPDATEONLY parameter in the ADAORD STORE function for an existing file, the value used previously for the file is used.

**SECURITY: Store the Security File**

SECURITY stores the security file from the DDFILEA/ FILEA tape, making that file the new security file for the database. The new file must have the same number as the old security file.

If SECURITY is omitted, the security file on the FILEA/ DDFILEA tape is not stored in the database, even if it is specified by a FILE parameter or the ALLFILES parameter is specified.

**SLOG: Store the SLOG System File**

Use this parameter to store the SLOG system file from the DDFILEA/FILEA tape as the new SLOG file for the Event Replicator Server. The new SLOG file must have the same file number as the old SLOG file. This parameter can only be specified if the database is an Event Replicator Server.

When the SLOG parameter is not specified, the SLOG system file on the DDFILEA/FILEA tape is not stored in the database, even if it is specified by a FILE or ALLFILES parameter.

**TEST: Test Syntax**

This 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.

**TRIGGER: Store the Trigger File**

Specify the TRIGGER parameter to store the trigger file from the DDFILEA/FILEA tape as the new trigger file for the database. The new trigger file must have the same file number as the old trigger file.

When the TRIGGER parameter is *not* specified, the trigger file on the DDFILEA/FILEA tape is *not* stored in the database, even if it is specified by the FILE or ALLFILES parameter.

**UIRABN: Starting RABN for Upper Index**

UIRABN specifies the beginning RABN for the upper index extent of the file. If this parameter is omitted, ADAORD assigns the starting RABN.

**UIRELEASE: Release Unused Upper Index Blocks**

Specifying UIRELEASE releases unused upper index (UI) blocks belonging to the file. If UIRELEASE is not specified, ADAORD allocates *at least* the number of UI blocks that were allocated before the file was reordered.

**Note:**
Adabas calculates the file extent size using any changed padding factor or block size values *before* the file is reordered.

**UISIZE: Upper Index Size**

UISIZE specifies the number of blocks or cylinders to be allocated for the upper index. A block count must be followed by a "B" (for example, "2000B").

If this parameter is omitted, ADAORD computes the file extent size (in blocks) in proportion to an increase or decrease in the ASSOPFAC padding factor used.

If this parameter is omitted and the INDEXCOMPRESSION parameter is specified, the ADAORD index size calculation does not consider the change in index size because ADAORD has no knowledge of the compression rate to be expected. Thus, ADAORD may allocate an index smaller than required causing secondary index extent allocations; or larger than necessary.

# Examples

### Example 1:

```
ADAORD STORE FILE=14,OVERWRITE
```

File 14, as unloaded by one of the RESTRUCTURE or the REORDB functions, is to be stored into an existing database. If the file already exists, it is deleted before being stored.

### Example 2:

```
ADAORD   STORE   FILE=1,OVERWRITE
ADAORD           FILE=2,OVERWRITE
ADAORD           FILE=3,OVERWRITE
```

Files 1, 2 and 3 are written to the existing database. Old files 1, 2 and 3 are deleted.

### Example 3:

```
ADAORD   STORE   OVERWRITE,ALLFILES
ADAORD           FILE=1,ACRABN=1000,NIRABN=2200
ADAORD           FILE=2,MAXISN=500000
ADAORD
FILE=4,ASSOPFAC=5,DATAPFAC=20,DSSIZE=5B,DSRABN=1
```

All files unloaded by the RESTRUCTURE function are to be stored into an existing database. The address converter for file 1 is to begin with RABN 1000. The normal index for file 1 is to begin with RABN 2200. The MAXISN for file 2 is to be set to 500,000. The following assignments are made for file 4: the Associator block padding factor is set to 5 percent; the Data Storage block padding factor is set to 20 per cent; a new DSSIZE of 5 cylinders is assigned starting at RABN 1.

All other files contained in the input data set are restored with their default values. If a file already exists, it is deleted before the new file is stored.

### Example 4:

```
ADAORD STORE ALLFILES,CHECKPOINT
ADAORD EXCLUDE=20,10
```

All files from the input data set (including the checkpoint file) are stored. However, files 10 and 20 are excluded; that is, not stored.

### Example 5:

```
ADAORD STORE ALLOCATION=NOFORCE
ADAORD FILE=10
ADAORD DSRABN=12345
```

File 10 is stored in the database. Its data storage is allocated beginning at RABN 12,345. If this allocation is not possible because the space is occupied by another file, ADAORD retries the allocation anywhere in the database’s data storage.