Modifying File Parameters

To modify parameters for a file:

1. List the files in a database, as described in *Listing Database Files*.

2. Select the file for which you want to modify the parameters by clicking on the filename.

   The property table for the file appears in detail-view.

3. Click on **Modify**.

   A property table for the file appears in detail-view, which allows you to modify the padding factor, the maximum compressed record length, file number, file name, extent allocation for NI/UI/AC/DS, ISN reusage, and DS reusage.

4. Modify the parameter settings according to your requirements and click **OK** to save your modifications and return to the previous panel.

   Or:
   For Adabas version 8 databases, the following additional parameters may be specified:

   - Record Spanning
   - MUPEX
   - Has LOB Fields
   - LOB File

   For detailed information on record spanning, MUPEX, LOB fields and files, see the *Adabas for Mainframes* documentation.

- Changing Padding Factors, Maximum Allocation, and Record Length
- Selecting File Options
- Record Spanning, MUPEX, LOB Fields/File (Adabas V8 only)

Changing Padding Factors, Maximum Allocation, and Record Length

You can use the Modify File Parameters function to change the Associator and Data Storage padding factors for the file, the maximum number of blocks that can be allocated (for a new Data Storage, normal index, or upper index extent), and the maximum compressed record length allowed.

The "padding factor" is the percentage (%) of each Associator or Data Storage block that is reserved; that is, not loaded. This area is used to create new records later. The range is from 3 to 90 percent. The factor size allocated should depend on the amount of updating that is expected. The number of bytes left in the Associator after padding must exceed the largest descriptor value by at least 10.

Selecting File Options

You can also turn off or on several file options in this table:

- Data Space Reusage ...with RESET / in Parallel
- ISN Reusage ...with RESET / in Parallel
- Mixed Device Types for DS Extents Allowed
- User Refresh Allowed
- User ISN

ISN Reusage and Data Storage Reusage determine whether ISNs and Data Storage blocks for deleted records are reused as new records are added to the file.

When setting either of these two options to "ON", you can also set the RESET option "ON" to start the search for an unused ISN and/or Data Storage block at the beginning of the file.

Record Spanning, MUPEX, LOB Fields/File (Adabas V8 only)

Record Spanning

With Adabas 8, records can be spanned in a database. In the database, the logical record is split into a number of physical records, each part fitting into a single Data Storage (DS) block. The resulting physical records are each assigned individual ISNs. The first physical record is called the primary record and contains the beginning of the compressed record and is assigned a primary ISN. The remaining physical records are called secondary records and contain the rest of the data of the logical record. Secondary records are assigned secondary ISNs. These ISNs do not affect the user ISNs assigned when using the N2 command or the ISNs used when using the I option of the L1 command. If spanned records are used, a secondary address converter is used to map the secondary ISNs to the RABNs of the Data Storage blocks where the secondary records are stored.

A spanned record is comprised of one primary record and one or more secondary records. However, the number of segments in a spanned record is limited. The Adabas nucleus allows up to five physical records (one primary record and four secondary records) in a spanned record.

Spanned records are not directly visible to application programs. Applications always address spanned records via the primary ISN.

Spanned records are also supported in expanded Adabas files and in multi-client files.

To count spanned records:

1. If the parameter has been specified, click on Record Spanning in the file property table to count the spanned records.

The Count Spanned Records table is displayed in detail-view.

2. Set the maximum running time for the count (default is 60 seconds) and supply the password if required.
3. Click **OK** to start the count.

MUPEX

With Adabas version 8 the number of occurrences of each MU field or each PE group in a record has been increased from 191 to about 65,534. However, the actual number of occurrences is limited to the size of the data-block size, the device type and the file type (spanned or not spanned). All MU fields and PE groups and other fields must fit into one compressed record. If you are using spanned records (introduced with Adabas 8), more MU fields and PE groups can be stored.

In addition, subdescriptors and superdescriptor definitions can affect the number of MU fields or PE groups in the record. For example, if a superdescriptor is created as a combination of a PE group and one or more MU fields and the number of occurrences is high, performance and resource problems can occur.

Note:
Excessive use of extended MU and PE fields might cause performance and resource problems. These can result in a work storage overflow, resulting in Response code 9. If this should happen, increase the ADARUN LP size for the database.

The use of more than 191 MU or PE fields in a record must be explicitly allowed for a file (it is not allowed by default).

LOB Fields/Files

Adabas 8 introduces large object fields (LB fields). These are fields that may contain much more data than the 253 bytes of normal alphanumeric fields or the 16,381 bytes of LA fields. Fields containing large objects are defined with the new LB option. The theoretical maximum size of the value of an LB field is just short of 2 GB; practical usable sizes are smaller.

Adabas version 8 stores large object field values in a separate file, called a LOB file, that is tightly associated with the file containing the LB fields, which is called the base file. Behind the scenes, Adabas will store LB field values (except for very short ones) in the LOB file, but this is transparent to your application. Commands in application programs should always be directed against the base file; application programs need neither know nor care about the existence of a LOB file.

For more information, see section *Creating a New LOB File*, and *Getting Started with Large Object (LB) Fields* in the *Adabas 8 for Mainframes* documentation.