COMPRESS: Compress an Adabas File

```
ADACMP  COMPRESS  {field-definition-statements | FDT = file-number}
                  [CODE = cipher-code ]
                  [DATADEVICE = device-type ]
                  [DEVICE = device-type-list ]
                  [FACODE = file-alpha-EBCDIC-key ]
                  [FILE = {file-number | 0 } ]
                  [FWCODE = file-wide-key ]
                  [FUWCODE = wide-key ]
                  [FORMAT = format ]
                  [HEADER = { YES | NO } ]
                  [LOBDEVICE = device-type-list ]
                  [LOBVALUES = { YES | NO } ]
                  [MAXLOGRECLEN = buffer-size ]
                  [MUPECOUNT = 1 | 2 ]
                  [MUPEX]
                  [NOUSERABEND]
                  [NUMREC = number-of-records]
                  [PASSWORD = "password" ]
                  [RECFM = { F | FB | V | VB | U } ]
                        [,LRECL = record-length } ]
                  [SPAN]
                  [TZ = ' timezone-name ' [,DST] ]
                  [ { USERISN | MINISN = { start-isn | 1 } } ]
                  [UACODE = userdata-alpha-key ]
                  [UARC = {userdata-architecture-key | 2 } ]
                  [UWCODE = userdata-wide-key ]
```

This chapter describes the syntax and parameters of the ADACMP COMPRESS function.

- Essential Parameters and Subparameters
- Optional Parameters and Subparameters
- ADACMP COMPRESS Examples

Essential Parameters and Subparameters

field-definition-statements

Field definition statements, when provided as input to ADACMP, are used to:

- provide the length and format of each field contained in the input record. This enables ADACMP to determine the correct field length and format during editing and compression.

- create the Field Definition Table (FDT) for the file. This table is used by Adabas during the execution of Adabas commands to determine the logical structure and characteristics of any given field (or group) in the file.

Either an FDT parameter or field definition statements must be supplied for ADACMP COMPRESS. If both are supplied, the field definition statements are ignored.

The field definition statements that can be included in this syntax:

```
FNDEF = 'field-definition'
[COLDE = 'collation-descriptor-definition' ]
[HYPDE = 'hyperdescriptor-definition' ]
[PHONDE = 'phoneticdescriptor-definition' ]
[SUBDE = 'subdescriptor-definition' ]
[SUBFN = 'subfield-definition' ]
[SUPDE = 'superdescriptor-definition' ]
[SUPFN = 'superfield-definition' ]
```

For complete information on field definition statements, including their syntax, read *Field Definition Statements*.

FDT: Use Existing Adabas Field Definition Table

Specifies an existing Adabas FDT to be used. The FDT may be that of an existing file or a file that has been deleted with the KEEPFDT option of the ADADBS utility.

Either an FDT parameter or field definition statements must be supplied for ADACMP COMPRESS. If both are supplied, the field definition statements are ignored.

If the FDT parameter is specified, the input data must be consistent with the structure as defined in the specified FDT, unless the FORMAT parameter is used. When the FDT defines multiple-value fields or periodic groups, length values must be defined or already included in the FDT. Read sections *Multiple-Value Field Count* and *Periodic Group Field Count*.

If the FDT parameter is used, any field definitions specified will be ignored.

Warning:
ADACMP COMPRESS utility runs that specify an FDT (via the FDT parameter) but do not specify a FORMAT parameter and that run against a file with logically deleted fields require that the data include the values for the logically deleted fields. Failure to include these values could lead to incorrectly compressed records.

Optional Parameters and Subparameters

CODE: Cipher Code

If the data is to be loaded into the database in ciphered form, the cipher code must be specified with this parameter. See the *Adabas Security* documentation for additional information on the use of ciphering.

Note:
You cannot specify the CODE parameter in a ADACMP COMPRESS function if the file contains LB fields.

DATADEVICE: Device Type

The DATADEVICE parameter specifies the Data Storage device type to be used for the segmentation of spanned records. If the SPAN parameter is specified, ADACMP will break long, spanned, compressed records into segments that are just a bit smaller than the Data Storage block size implied by the DATADEVICE parameter.

If the SPAN parameter is not specified, no value for DATADEVICE is required. However, it can be specified to limit the size of compressed records. In this case, all records that exceed the given storage device block size will be written to the DDFEHL error data set.

DEVICE: Device Type

If the DEVICE parameter is specified, ADACMP calculates and displays a report of this run's space requirements for each specified device type. This report includes an indication of whether or not the MUPEX parameter has been set for a file.

If no device types are listed on the DEVICE parameter, the ADARUN device type is used as the default.

DST: Daylight Savings Indicator

The DST parameter can be specified to indicate that date-time data includes a daylight savings time indicator. If a time zone uses daylight savings time, you must be sure to store and retrieve the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time. The two-byte daylight savings indicator directly follows the date-time value in uncompressed input and specifies the hexadecimal value of the daylight saving time offset from standard time in seconds.

The DST parameter requires that the TZ parameter be set in the same ADACMP run. However, it should *not* be specified when the FORMAT parameter is specified in the run.

You must specify a DST parameter for files containing date-time fields defined with option TZ, when the time zone includes a daylight savings time indicator. If the DST parameter is not specified, date-time data is stored without a daylight savings time indicator. The default is store the date-time data without a daylight savings time indicator.

FACODE: Alphanumeric Field Encoding

FACODE must be specified if you want to define UES file encoding for alphanumeric fields in the file. The alphanumeric encoding must belong to the EBCDIC encoding family; that is, the space character is X'40'.

FILE: File Number

If the FDT contains a hyperdescriptor, this parameter must be specified. The specified file number becomes input for the related hyperdescriptor exit. For more information about hyperdescriptor exits, refer to the *Adabas DBA Reference* documentation.

User exit 6 is always supplied with this file number. If FILE is not specified, a value of zero is assumed.

FORMAT: Input Record Format Definition

Use this parameter to provide a format definition that indicates the location, format, and length of fields in the input record. The format provided must follow the rules for format buffer entries for update commands as described in the *Adabas Command Reference Guide* documentation.

Conversion rules are those described for Adabas update commands in the *Adabas Command Reference Guide* documentation. For conversion of SQL null (NC option) field values, see *NC: SQL Null Value Option*. If a field is omitted in the FORMAT parameter, that field is assigned no value.

If the FORMAT parameter is omitted, the input record is processed in the order of the field definition statements provided or, if the FDT parameter is used, according to an existing Adabas field definition table.

If the LOBVALUES parameter is set to NO, LB fields cannot be used in the definition supplied in the FORMAT parameter.

FUWCODE: Wide-Character Field Default User Encoding

FUWCODE defines the default user encoding for wide-character fields for the file when loaded in the database. If this parameter is omitted, the encoding is taken from the UWCODE definition of the database.

FWCODE: Wide-Character Field Encoding

If fields with format W (wide-character) exist in the compressed file, you *must* specify FWCODE to define the file encoding for them.

FWCODE also determines the maximum byte length of the wide-character field.

HEADER

This optional parameter indicates whether or not the ADACMP compression logic should expect segmented ADACMP record headers in the uncompressed input records. Valid values are YES or NO; the default is NO.

HEADER=NO is the format accepted and produced by ADACMP in Adabas versions prior to Adabas 8. When it is specified, the input records must contain only the uncompressed data for the fields of the file being processed. Each data record must fit into one physical record of the sequential input data set (less than 32 KB). If the data exceeds this size, the records in error are written to the DDFEHL error data set.

HEADER=YES can only be specified if you are running Adabas 8. If HEADER=YES is specified, each input record must begin with either an ADAH or ADAC header, relating the physical segmented record with a logical record to be processed by ADACMP. Each logical record can be larger than 32 KB. The header in each physical record defines the position of the data following it within the logical record. DSECTs for the ADAH and ADAC headers can be found in members ADAH and ADAC of the distributed Adabas SRCE data set.

If HEADER=YES is specified, an error may occur while segmented uncompressed records are being assembled into a logical record. If the ADAH header is in error, the ADAH record is written and subsequent ADAC records are not written until the next ADAH record is processed. If an ADAC header is in error, the preceding ADAH header will be written without its payload data. The ADAC record in error will be written in its entirety. Subsequent ADAC records are not written until the next ADAH record is processed. For more information about rejected records and possible response codes resulting from them, read *COMPRESS Function Output*.

Do not confuse the HEADER parameter with the SPAN parameter. The SPAN parameter controls whether the compressed records themselves should be spanned if they exceed the Data Storage block size of the device. Spanned records contain a standard spanned record header that differs from the ADAH or ADAC headers expected by the HEADER parameter. For complete information about spanned records, read *Spanned Records*.

LOBDEVICE: Device Type for LOB File

This optional parameter specifies the data storage device type that will be used for loading the *LOB file* produced by the ADACMP COMPRESS function. ADACMP will divide the LB field values into segments based on the block size of the specified device. This parameter is only valid if the FDT includes one or more large object (LB option) fields.

If LOBDEVICE is not specified, the device specified for the ADACMP COMPRESS DEVICE parameter is used. If no device is specified for the DEVICE parameter either, the value of the ADARUN DEVICE parameter is used.

LOBVALUES: LB Field Size Indicator

This optional parameter indicates whether long LB field values (larger than 253 bytes) or short LB field values (up to 253 bytes) are expected in the ADACMP COMPRESS input data. Valid values for this parameter are "YES" and "NO"; the default is "NO".

If "YES" is specified for this parameter, the uncompressed input data may contain LB field values larger than 253 bytes. In this case, a second sequential output data set must also be supplied in the JCL for the run. This second data set is identified in the JCL using the DD control statement DDAUSB1. It is used to store the compressed LB segment records for LB field values that are larger than 253 bytes.

If "NO" is specified for this parameter, the uncompressed input data may contain only LB fields up to 253 bytes long and references to LB field values stored in a *LOB file*. In this case, you cannot specify an LB field in the ADACMP COMPRESS FORMAT parameter. During processing, ADACMP writes any short LB field values and LB field value references contained in the input to the output.

Note:
An ADACMP DECOMPRESS function with LOBVALUES=NO followed by an ADACMP COMPRESS function with LOBVALUES=NO can be used to modify the FDT of the *base file* in the *LOB file group*.

LRECL: Input Record Length (z/VSE Only)

If RECFM=F or RECFM=FB is specified, this parameter must also be specified to provide the record length (in bytes) of the input data; otherwise, do not specify LRECL.

For z/OS, the record length is taken from the input data set label or DD statement.

For BS2000, the record length is taken from the catalog entry or /FILE statement.

MAXLOGRECLEN: Buffer Size

This optional parameter can be used to specify the size, in bytes, of a buffer used by ADACMP to assemble any segmented, uncompressed, physical records into a compressed logical record. This buffer is allocated only if HEADER=YES is also specified. Otherwise, the setting of MAXLOGRECLEN is ignored. The default value of MAXLOGRECLEN is 1,048,576 bytes (1 MB).

If the value specified by MAXLOGRECLEN is appended by the letter "K", it is multiplied by 1024. The minimum value is 32768 bytes.

Note:
MAXLOGRECLEN pertains to the spanning of uncompressed input data and should not be confused with the SPAN parameter which pertains to the spanning of compressed records.

MINISN: Starting ISN

For automatic ISN assignment, MINISN defines the lowest ISN to be used. If MINISN is not specified, the default is 1. If USERISN is specified, MINISN cannot be specified.

MUPECOUNT: Specify Value Count Field Size

The MUPECOUNT parameter specifies the size of the value count field in the input record for the COMPRESS function . Its syntax is:

```
MUPECOUNT={1 | 2}
```

If "1" is specified, each value count field preceding the MU or PE values in the input data must be one byte with a value of no more than "191". If "2" is specified, each value count field preceding the MU or PE values in the input data must be two bytes. A value count may exceed 191 only if the MUPEX parameter is also specified. When MUPEX is specified, the maximum count is "65,534".

If the MUPEX parameter has been set, the default for MUPECOUNT is "2"; if the MUPEX parameter has not been set, the default for MUPECOUNT is "1".

Note:
This option is not compatible with releases prior to Adabas 8; therefore, backward conversion to prior versions is not possible once records with more than 191 PE group occurrences have been loaded. However, ADACMP data sets created by versions of Adabas prior to Adabas 8 will load successfully using the Version 8 ADALOD utility.

For information on how to identify MU and PE occurrences greater than 191 in the compressed record, read *Identifying MU and PE Occurrences Greater Than 191 in Compressed Records*.

MUPEX: Enable Extended Periodic Group Count

The MUPEX parameter indicates whether extended MU/PE limits (greater than 191) are allowed for the file. If this option is *not* specified, the maximum number of MU fields and the maximum number of PE fields that can be specified is 191. Otherwise, the maximum is 65,534.

If you set the MUPEX parameter, consider setting the SPAN parameter as well to avoid compression errors if the compressed record size is exceeded when compressing the additional MU and PE fields.

Note:
This option is not compatible with releases prior to Adabas 8; therefore, backward conversion to prior versions is not possible once records with more than 191 PE group occurrences have been loaded. However, ADACMP data sets created by versions of Adabas prior to Adabas 8 will load successfully using the Version 8 ADALOD utility.

For information on how to identify MU and PE occurrences greater than 191 in the compressed record, read *Identifying MU and PE Occurrences Greater Than 191 in Compressed Records*.

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: Number of Records to Be Processed

Specifies the number of input records to be processed. If this parameter is omitted, all input records contained on the input data set are processed.

Software AG recommends using this parameter for the initial ADACMP execution if a large number of records are contained on the input data set. This avoids unneeded processing of all records when a field definition error or invalid input data results in a large number of rejected records. This parameter is also useful for creating small files for test purposes.

Setting NUMREC to zero (0) prevents the input data set from being opened.

PASSWORD: Password for FDT File

If the FDT parameter is specified and the file is password-protected, this parameter must be used to provide a valid password for that file.

RECFM: Input Record Format (z/VSE Only)

You *must* specify the input record format with this parameter as follows:

F	fixed length, unblocked (requires that you also specify the LRECL parameter)
FB	fixed length, blocked (requires that you also specify the LRECL parameter)
V	variable length, unblocked
VB	variable length, blocked
U	undefined

Under z/OS, the record format is taken from the input data set label or DD statement.

Under BS2000, the record format is taken from the catalog entry or FILE statement.

SPAN: Enabling Spanned Records

The SPAN parameter allows the record to be spanned after compression if its compressed size exceeds the Data Storage block size of the device.

TZ: Time Zone

The TZ parameter can be used to specify the local time zone. As records are compressed and loaded into the file, the date-time data is converted from the specified time zone to UTC time (Coordinated Universal Time, also known as Greenwich Mean Time). Date-time field data is always stored in UTC time in an Adabas file. Valid time zone names are listed in the TZINFO member of the Adabas source library and are specified in single quotes. The following is an example of a valid TZ specification:

```
TZ='America/New_York'
```

Note:
Time zone names are case-sensitive.

This example sets the time zone to Eastern US. When compressing input, local time is assumed to be Eastern US and will be converted to UTC time for storage on the database.

Adabas uses the time zone data taken from the tz database, which is also called the *zoneinfo* or *Olson* database. The specific list of time zone names that Adabas supports in any given release can be found in the TZINFO member of the Adabas source library (ADA*vrs*.SRC in BS2000 environments, ADA*vrs*.LIBR in VSE environments, and ADA*vrs*.SRCE in z/OS environments.). For more information about the TZINFO member of the time zone library, read *Supported Time Zones*.

Note:
Review important information about the daylight savings time (DST) parameter. If a time zone uses daylight savings time, you must be sure to store the daylight savings indicator with your date-time data or there will be no way to distinguish date-time values in the hour before the time is switched back to standard time.

UACODE: User Encoding for Input Alphanumeric Fields

UACODE defines the user encoding of the sequential input of alphanumeric fields. If you specify UACODE, you *must* also specify FACODE.

UARC: Architecture for Input Uncompressed User Data

The UARC parameter specifies the architecture of the sequential input of the uncompressed user data. The "userdata-architecture-key" is an integer which is of the sum of the following numbers:

byte order	b=0	high-order byte first
	b=1	low-order byte first
encoding family	e=0	ASCII encoding family
	e=2	EBCDIC encoding family (default)
floating-point format	f=0	IBM370 floating-point format
	f=4	VAX floating-point format
	f=8	IEEE floating-point format

The default is ARC = b + e + f = 2; that is, high-order byte first; EBCDIC encoding family; and IBM370 floating-point format (b=0; e=2; f=0).

User data from an Intel386 PC provides the example: b=1; e=0; f=8; or ARC=9.

USERISN: User ISN Assignment

The ISN for each record is to be user-defined. If this parameter is omitted, the ISN for each record is assigned by Adabas.

If USERISN is specified, you must provide the ISN to be assigned to each record as a four-byte binary number immediately preceding each data record. If the MINISN parameter is specified, USERISN cannot be specified.

If USERISN is specified with HEADER=YES, the ISN immediately follows the ADAH header as part of the logical record.

The format for fixed or undefined length input records with user-defined ISNs is:

```
userisn/data
```

The format for variable-length input records with user-defined ISNs is

```
length/xx/userisn/data
```

where

length	is a two-byte binary physical record length (length of record data, plus 8 bytes).
xx	is a two-byte field containing binary zeros.
userisn	is a four-byte binary ISN to be assigned to the record.
data	is input record data.

ISNs may be assigned in any order, must be unique (for the file), and must not exceed the MAXISN setting specified for the file (see the ADALOD utility documentation).

ADACMP does not check for unique ISNs or for ISNs that exceed MAXISN. These checks are performed by the ADALOD utility.

UWCODE: User Encoding for Input Wide-Character Fields

UWCODE defines the user encoding of the sequential input of wide-character fields. If you specify UWCODE, you *must* also specify FWCODE.

For user input, all wide-character fields are encoded in the same code page. It is not possible to select different encodings for different fields in the same ADACMP run.

ADACMP COMPRESS Examples

Example 1:

```
ADACMP   COMPRESS
ADACMP   FNDEF='01,AA,7,A,DE,FI'                                  Field AA
ADACMP   FNDEF='01,AB,15,A,DE,MU,NU'                          Field AB
ADACMP   FNDEF='01,GA'                                              Group GA
ADACMP   FNDEF='02,AC,15,A,NU'                                   Field AC
ADACMP   FNDEF='02,AD,2,P,FI'                                       Field AD
ADACMP   FNDEF='02,AE,5,P,NU'                                    Field AE
ADACMP   FNDEF='02,AF,6,W'                                          Field AF
ADACMP   FNDEF='01,DT,8,P,DT=E(DATETIME),TZ'     Field DT
ADACMP   COLDE='7,Y1=AF'                    Collation descriptor Y1
ADACMP   SUBDE='BB=AA(1,4)'                 Subdescriptor BB
ADACMP   SUPDE='CC=AA(1,4),AD(1,1)'         Superdescriptor CC
ADACMP   HYPDE='1,DD,4,A,MU=AB,AC,AD'       Hyperdescriptor DD
ADACMP   PHONDE='EE(AA)'                    Phonetic descriptor EE
ADACMP   SUBFN='FF=AA(1,2)'                 Subfield FF
ADACMP   SUPFN='GG=AA(1,4),AD(1,1)'         Superfield GG
ADACMP   TZ='Europe/Berlin',DST         Time zone specification
```

The time zone in this example is set to "Europe/Berlin", with a day light savings indicator. The following fields are defined:

Field	Is defined as ...
AA	Level 1, seven bytes alphanumeric, descriptor, fixed storage option.
AB	Level 1, 15 bytes alphanumeric, descriptor, multiple value field, null value suppression.
GA	A group containing fields AC, AD, AE, and AF.
AC	Level 2, 15 bytes alphanumeric, null value suppression.
AD	Level 2, two bytes packed, fixed storage option.
AE	Level 2, five bytes packed, null value suppression.
AF	Level 2, six bytes wide-character format.
DT	Level 1, 8 bytes, packed, using a DATETIME edit mask and for which time zone conversion is requested. **Note:** The time zone for compression is specified as "Europe/Berlin" time and includes a daylight savings indicator.
Y1	A collation descriptor for AF and is assigned to collation descriptor user exit 7 (CDX07).
BB	A subdescriptor (positions 1-4 of field AA).
CC	A superdescriptor (positions 1-4 of field AA and position 1 of field AD).
DD	A hyperdescriptor consisting of fields AB, AC and AD. DD is assigned hyperdescriptor exit 1.
EE	A phonetic descriptor derived from field AA.
FF	A subfield (positions 1-2 of field AA).
GG	A superfield (positions 1-4 of AA and position 1 of AD).

Example 2:

```
ADACMP   COMPRESS
ADACMP   FORMAT='AG,6,U,AF,4X,AA,'    input record format
ADACMP   FORMAT='AB,AC'               continuation of FORMAT statement
ADACMP   FNDEF='01,AA,10,A,NU'        field definitions
ADACMP   FNDEF='01,AB,7,U,NU'
ADACMP   FNDEF='01,AF,5,P,NU'
ADACMP   FNDEF='01,AG,12,P,NU,DE'
ADACMP   FNDEF='01,AC,3,A,NU,DE'
```

The input record format is provided explicitly using the FORMAT parameter. ADACMP uses this format as the basis for processing fields from the input record. The FDT for the file corresponds to the structure specified in the FNDEF statements.

Example 3:

```
ADACMP COMPRESS
ADACMP FORMAT='AG,AF,4X,AA,AB,AC'     input record format
ADACMP FDT=8                          FDT same as file 8
```

The input record format is provided explicitly using the FORMAT parameter. The FDT to be used is the same as that currently defined for Adabas file 8.

Example 4:

```
ADACMP   COMPRESS NUMREC=2000,USERISN
ADACMP   FNDEF='01,AA,7,A,DE,FI'            Field AA
ADACMP   FNDEF='01,AB,15,A,DE,MU,NU'        Field AB
```

The number of input records to be processed is limited to 2,000. The ISN for each record is to be provided by the user.

Example 5:

```
ADACMP   COMPRESS RECFM=FB,LRECL=100
ADACMP   FNDEF='01,AA,7,A,DE,FI'            Field AA
ADACMP   FNDEF='01,AB,15,A,DE,MU,NU'        Field AB
```

A z/VSE input file contains fixed length (blocked) records. The record length is 100 bytes.