Processing

This chapter describes ADACMP utility processing.

- Segmented Record Considerations
- Logically Deleted Fields
- Data Verification
- Data Compression
- Representation of LOB Values and Value References in Uncompressed Data
- Identifying MU and PE Occurrences Greater Than 191 in Compressed Records
- Restart Considerations
- User Exit 6

Segmented Record Considerations

If a decompressed record from ADACMP is too long to fit into the longest record length allowed for a sequential data set (32KB or less), ADACMP can segment it into multiple physical records. A single logical decompressed record can span one or more physical decompressed records.

In addition, the ADACMP utility allows you to create special headers, ADAH and ADAC, in the decompressed output. These special headers are used only with ADACMP processing. They identify the position of the payload data in the logical record as well as the relation between the physical record and the other physical records in the same logical record. When created, an ADAH header is used for the first physical record of a logical record; ADAC headers are used for the second and subsequent physical records that comprise the logical record. If a decompressed logical record does not need to be segmented (if it fits into one physical record), only an ADAH header is created; there is no need for ADAC headers.

Do not confuse the record segmentation that occurs with ADACMP decompression logic and *record spanning*. Spanned records also consist of multiple physical records (one primary record and multiple secondary records), but they are compressed records. In addition, each spanned record is automatically assigned a standard spanned record header that is not the same as the ADAH and ADAC headers you can create for decompressed records using ADACMP; segmented records produced by ADACMP do not contain the standard spanned record header. For complete information about spanned records, read *Spanned Records*.

This section covers the following topics:

- Creating and Supporting ADACMP Headers
- ADAH and ADAC Header Descriptions

Creating and Supporting ADACMP Headers

The HEADER parameter of ADACMP DECOMPRESS controls whether the decompression logic produces the headers in its *output*. The HEADER parameter of ADACMP COMPRESS controls whether the compression logic will accept the ADACMP headers as part of the uncompressed *input*.

ADAH and ADAC Header Descriptions

This section covers the following topics:

- ADAH Headers
- ADAC Headers
- Example

ADAH Headers

When ADACMP headers are used, the first physical record of a logical record begins with an ADAH header containing the following information:

- The characters "ADAH".
- The length of the ADAH header.
- A continuation indicator that indicates whether this is the last physical record in the logical record or whether another physical record for the same logical record will follow this one.
- The total length of the record (with the headers). The value may be zero if the total record length is not known when the first physical record is written.
- The length of the payload data (a segment of the logical record) in this physical record. This refers to the length of the payload data; it does not include the length of the ADAH header. The length must be less than or equal to the length of the physical record minus the length of the header. If it is less than this value, any extra data in the physical record (not covered by the payload data length) is ignored.

The payload data follows the ADAH header.

The ADAH DSECT can be found in the ADAH member of the distributed Adabas 8 SRCE library.

ADAC Headers

When ADACMP decompressed record segmenting occurs and when ADACMP headers are requested, the second and every subsequent physical record for a logical record begins with an ADAC header containing the following information:

- The characters "ADAC".
- The length of the ADAC header.
- A continuation indicator that indicates whether this is the last physical record in the logical record or whether another physical record for the same logical record will follow this one.

- The sequence number of this secondary record in the logical record. The second physical record of a logical record is the first secondary record and therefore has a sequence number of "1". The sequence numbers are in ascending order, without gaps.
- The offset within the logical record of the payload data (segment) contained in this physical record. This offset is the sum of the payload data lengths of each prior physical record in the logical record.
- The length of the payload data (segment) in this physical record. This refers to the length of the payload data; it does not include the length of the ADAC header. The length must be less than or equal to the length of the physical record minus the length of the header. If it is less than this value, any extra data in the physical record (not covered by the payload data length) is ignored.

The payload data follows the ADAC header.

The ADAC DSECT can be found in the ADAC member of the distributed Adabas 8 SRCE library.

Example

The following table depicts three logical records spanning seven physical records of uncompressed data.

Note:
DSECTs for the ADAH and ADAC headers can be found in members ADAH and ADAC in the distributed Adabas 8 SRCE library.

Logical Record	Physical Record Headers	Header Fields		Description
		Field	Value	
1	ADAH	ADAHEYE	ADAH	ADAH header eyecatcher
		ADAHLEN	32	ADAH header length
		ADAHIND	C	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Must contain binary zeros.
		ADAHTLEN	50000	Total length of logical record. This value may be zero if the total length is not known when the first segment is written.
		Reserved	0	Reserved
		ADAHSLEN	27962	Length of this segment (length of the payload data). The sum of the values of ADAHLEN and ADAHSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADAHDATA	'Record 1 - payload data part 1'	Payload data
	ADAC	ADACEYE	ADAC	ADAC header eyecatcher
		ADACLEN	32	ADAC header length
		ADACIND	E	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADACSEQ	1	Continuation record sequence number within the logical record (the first ADAC record has a sequence number of "1").
		ADACOFFS	27962	Segment offset within the logical record (the first payload data byte is at offset "0").
		Reserved	0	Reserved
		ADACSLEN	22038	Length of this segment (length of the payload data). The sum of the values of ADACLEN and ADACSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADACDATA	'Record 1 - payload data part 2'	Continuation record payload data

Logical Record	Physical Record Headers	Header Fields		Description
		Field	Value	
2	ADAH	ADAHEYE	ADAH	ADAH header eyecatcher
		ADAHLEN	32	ADAH header length
		ADAHIND	E	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Must contain binary zeros.
		ADAHTLEN	25000	Total length of logical record. This value may be zero if the total length is not known when the first segment is written.
		Reserved	0	Reserved
		ADAHSLEN	25000	Length of this segment (length of the payload data). The sum of the values of ADAHLEN and ADAHSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADAHDATA	'Record 2 - payload data'	Payload data

Logical Record	Physical Record Headers	Header Fields		Description
		Field	Value	
3	ADAH	ADAHEYE	ADAH	ADAH header eyecatcher
		ADAHLEN	32	ADAH header length
		ADAHIND	C	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Must contain binary zeros.
		ADAHTLEN	0	Total length of logical record. This value may be zero if the total length is not known when the first segment is written (as is the case for logical record 3).
		Reserved	0	Reserved
		ADAHSLEN	27962	Length of this segment (length of the payload data). The sum of the values of ADAHLEN and ADAHSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADAHDATA	'Record 3 - payload data part 1'	Payload data
	ADAC	ADACEYE	ADAC	ADAC header eyecatcher
		ADACLEN	32	ADAC header length
		ADACIND	C	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADACSEQ	1	Continuation record sequence number within the logical record (the first ADAC record has a sequence number of "1").
		ADACOFFS	27962	Segment offset within the logical record (the first payload data byte is at offset "0").
		Reserved	0	Reserved
		ADACSLEN	27962	Length of this segment (length of the payload data). The sum of the values of ADACLEN and ADACSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADACDATA	'Record 3 - payload data part 2'	Continuation record payload data
	ADAC	ADACEYE	ADAC	ADAC header eyecatcher
		ADACLEN	32	ADAC header length
		ADACIND	C	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADACSEQ	2	Continuation record sequence number within the logical record (the first ADAC record has a sequence number of "1").
		ADACOFFS	55924	Segment offset within the logical record (the first payload data byte is at offset "0").
		Reserved	0	Reserved
		ADACSLEN	27962	Length of this segment (length of the payload data). The sum of the values of ADACLEN and ADACSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADACDATA	'Record 3 - payload data part 3'	Continuation record payload data
	ADAC	ADACEYE	ADAC	ADAC header eyecatcher
		ADACLEN	32	ADAC header length
		ADACIND	E	Continuation indicator. Valid values are: C: Continuation record segment to follow E: End of logical record (last segment)
		Reserved	0	Reserved
		ADACSEQ	3	Continuation record sequence number within the logical record (the first ADAC record has a sequence number of "1").
		ADACOFFS	83886	Segment offset within the logical record (the first payload data byte is at offset "0").
		Reserved	0	Reserved
		ADACSLEN	16114	Length of this segment (length of the payload data). The sum of the values of ADACLEN and ADACSLEN is the minimum length of the physical record. The physical record can be longer than this; in this case, the excess data has no meaning and is ignored.
		ADACDATA	'Record 3 - payload data part 4'	Continuation record payload data

Logically Deleted Fields

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 (see the ADADBS DELFN utility function) require that the data include the values for the logically deleted fields. Failure to include these values could lead to incorrectly compressed records.

Data Verification

ADACMP checks each field defined with format P (packed) or U (unpacked) to ensure that the field value is numeric and in the correct format. If a value is empty, the null characters must correspond to the format specified for the field (see *Representing SQL Null Values* in the *Field Definition Statements* section.

Alphanumeric (A)	blanks (hex ’40’)
Binary (B)	binary zeros (hex ’00’)
Fixed (F)	binary zeros (hex ’00’)
Floating Point (G)	binary zeros (hex ’00’)
Packed (P)	decimal packed zeros with sign (hex ’00’ followed by ’0F’, ’0C’, or ’0D’ in the rightmost, low-order byte)
Unpacked (U)	decimal unpacked zeros with sign (hex ’F0’ followed by ’C0’ or ’D0’ in the rightmost, low-order byte)

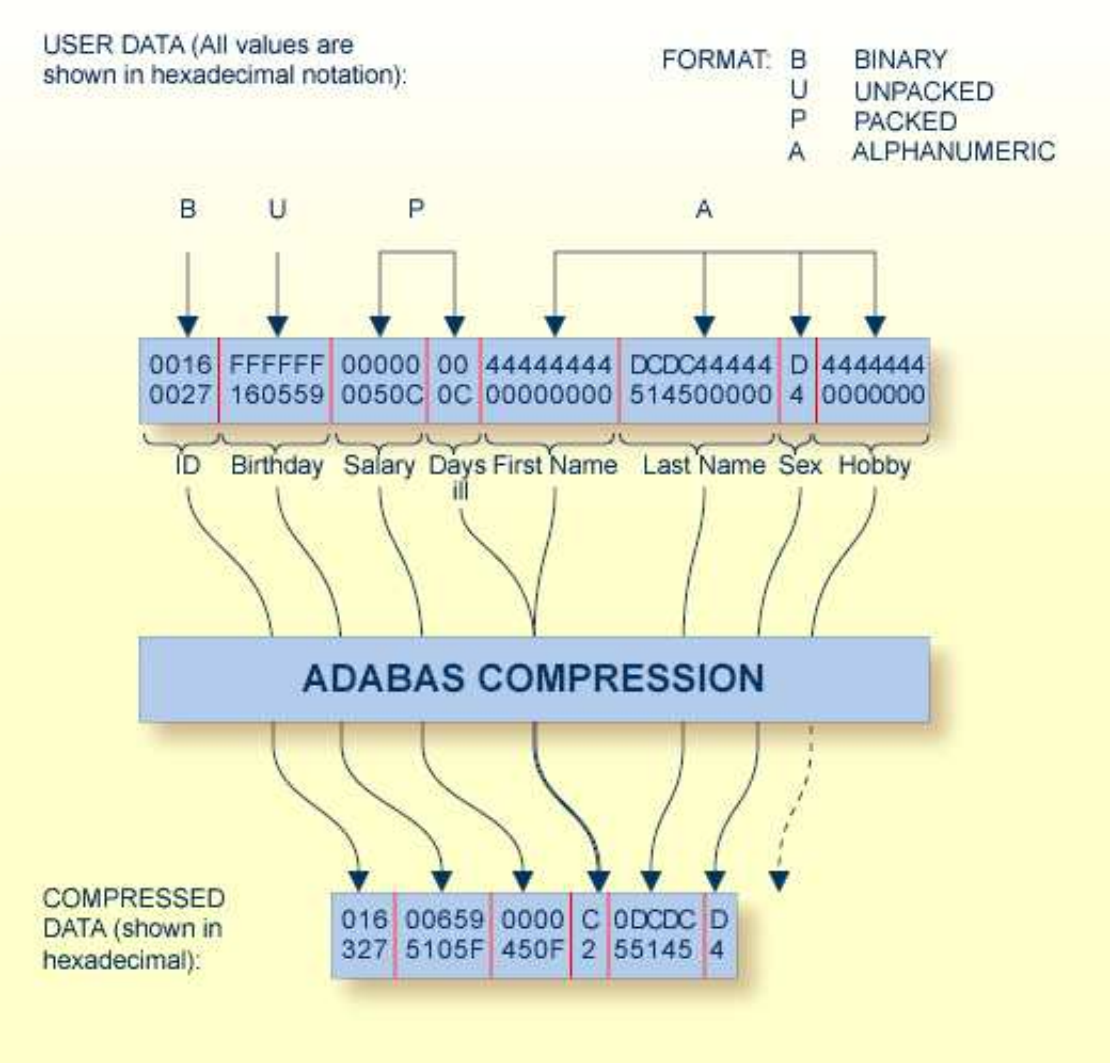
Any record that contains invalid data is written to the ADACMP error (DDFEHL) data set and is not written to the compressed data set.

Data Compression

The value for each field is compressed (unless the FI option is specified) as follows:

- Trailing blanks are removed for fields defined with A format.
- Leading zeros are removed for numeric fields (fields defined with B, F, P or U format).
- If the field is defined with U (unpacked) format, the value is converted to packed (P) format.
- Trailing zeros in floating-point (G format) fields are removed.
- If the field is defined with the NU option and the value is a null value, a one-byte indicator is stored. Hexadecimal ’C1’ indicates one empty field follows, ’C2’ indicates that two empty fields follow, and so on, up to a maximum of 63 before the indicator byte is repeated. For SQL null value (NC option field) compression, see *Representing SQL Null Values* in the *Field Definition Statements* section.
- Empty fields located at the end of the record are not stored, and therefore not compressed.

Example of Data Compression



ADACMP Compression

The graphic shows how the following field definitions and corresponding values would be processed by ADACMP:

```
FNDEF='01,ID,4,B,DE'
FNDEF='01,BD,6,U,DE,NU'
FNDEF='01,SA,5,P'
FNDEF='01,DI,2,P,NU'
FNDEF='01,FN,9,A,NU'
FNDEF='01,LN,10,A,NU'
FNDEF='01,SE,1,A,FI'
FNDEF='01,HO,7,A,NU'
```

Representation of LOB Values and Value References in Uncompressed Data

This section describes how large object (LB) field values, LB field value references, and logical records that are longer than 32 KB must be represented in the input data set for the ADACMP COMPRESS function and how these items are represented in the output data set of the ADACMP DECOMPRESS function.

- Large Object (LB) Field Values
- Large Object (LB) Field Value References

Large Object (LB) Field Values

If ADACMP is run without the FORMAT parameter, each large object (LB) field value in the uncompressed data is preceded by a 4-byte length field. The length value includes the length of the LB field value proper plus four bytes for the length field itself. An empty LB field value for a field defined *without* the NB option consists of the length field with a value of 5 and a single blank; for a field defined *with* the NB option, an empty LB field value consists only of the length field with a value of 4.

If ADACMP COMPRESS is used to define an FDT with LB fields, each LB field value in the uncompressed input must be less than or equal to 253 bytes.

Large Object (LB) Field Value References

When the ADACMP DECOMPRESS function is run with LOBVALUES=NO to decompress only the records from the *base file* of a *LOB file group*, omitting all LB field values stored in the associated *LOB file*, each reference in a base file record to a LB field value in the LOB file is represented in the uncompressed output as follows:

- The four-byte length field for the LB field value contains X’FFFFFFFF’ (high value) to indicate the presence of the reference to an LB field value.
- The indicator is followed by a two-byte inclusive length field for the LB field value reference. The length value includes the length of the LB field value reference proper plus two bytes for the length field itself.
- The length field is followed by the LB field value reference proper.

The same structure is expected by the ADACMP COMPRESS function with LOBVALUES=NO in the place of an LB field value that is stored in the *LOB file* associated with the *base file* that is being compressed.

LB field value references that are input to ADACMP COMPRESS must originate from ADACMP DECOMPRESS. There is no sensible way to introduce new LB field value references using COMPRESS, as they would not properly refer to existing LB field values in a *LOB file*.

Identifying MU and PE Occurrences Greater Than 191 in Compressed Records

MU and PE occurrences greater than 191 are indicated in compressed records by a x'C0' byte at the beginning of the occurrence count. This byte is set by the ADACMP utility or the nucleus when the records are compressed. The x'C0' indicator byte is followed by a byte indicating the number of count bytes used for the MU or PE occurrence count that follows. For example, consider the following indicator:

```
X'C0020204'
```

In this example, x'C0' indicates this is an extended count; x'02' indicates that there are two count bytes, and x'0204' indicates that there are 516 occurrences of the field.

Restart Considerations

ADACMP has no restart capability. An interrupted ADACMP execution must be reexecuted from the beginning.

User Exit 6

A user-written routine called user exit 6 can be used for editing during ADACMP COMPRESS processing. The routine may be written in Assembler or COBOL. It must be assembled or compiled and then linked into the Adabas load library (or any library concatenated with it).

User exit 6 is invoked by specifying:

```
ADARUN UEX6=program
```

where *program* is the routine name in the load library.

For specific information about the user exit 6 structure and parameters, read *User Exits and Hyperdescriptor Exits*.