Adabas

User, Hyperdescriptor, Collation Descriptor, and SMF Exits

Version 8.6.2

October 2026

ADABAS & NATURAL

This document applies to Adabas Version 8.6.2 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 1971-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software GmbH.

Document ID: ADAMF-EXITS-862-20261013

Table of Contents

User, Hyperdescriptor, Collation Descriptor, and SMF Exits v
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
2 User Exit Calling Conventions 5
Standard MVS Calling Conventions 6
User Exit Invocation 6
User Exit Processing 6
Return from User Exits 7
3 User Exit 1 (General Processing) 9
4 User Exit 2 (Dual Log Processing) 11
User Exit 2 Calling Sequence 13
Input Parameters 14
Output Parameters 16
5 User Exit 3 (User-Defined Phonetic Processing) 17
Input Parameters 18
6 User Exit 4 (User-Generated Log Data) 19
Command Log Format 20
Invoking User Exit 4 23
Accessing Adabas Buffers via CQX 24
Accessing Adabas Buffers via LORECX 25
7 User Exit 5 (Adabas Review Hub Event Handler) 27
Input Parameters 28
Output Parameters 29
8 User Exit 6 (User Processing Before Data Compression) 31
ADACMP Header Processing 34
Input Parameters 35
Output Parameters 35
9 User Exit 8 (Operator Interface) 37
Input Parameters 39
10 User Exit 9 (ADAULD) 41
Processing 42
User Exit 9 Sample 44
11 User Exit 11 (General Processing) 45
Input and Output Parameters 47
12 User Exit 12 (Multiple Data Set Log Processing) 49
User Exit 12 Calling Sequence 52
User Exit Interface 52
Output Parameter 55
Using Sample User Exit 12 55
13 Hyperdescriptor Exits 01 - 31 59
Main Parameter Area 61

Input Parameter Area (Pointed to by Third Parameter Address) 62
Output Parameter Area 64
Null Value Option 65
Hyperdescriptor Exit Initialization Call 66
Hyperdescriptor Exit Sample 66
14 Collation Descriptor Exits 01 - 08 67
Collation Descriptor Exit Interface 68
15 SMF User Exit 71
Index 73

User, Hyperdescriptor, Collation Descriptor, and SMF Exits

This document refers to the user exits activated by the ADARUN parameters UEX*n*, HEX*nn*, and CDX*nn* (see the *Adabas Operations* documentation for descriptions of the ADARUN parameters).

Caution: All supplied sample user exits are sample user programs and are not supported under any maintenance contract agreement.

Note: Adabas user exits must adhere to the standard MVS calling conventions. They must return control in the same processor state they were called in. For more information, please see ***User Exit Calling Conventions***.

The user exits documented in this document are as follows:

User Exit	ADARUN	Use
User Exit 1	UEX1	Command processing (Adabas nucleus) -- being retired
User Exit 2	UEX2	Dual log processing
User Exit 3	UEX3	User-defined phonetic processing
User Exit 4	UEX4	User-generated log data
User Exit 5	UEX5	Adabas Review hub event handler
User Exit 6	UEX6	Data compression (ADACMP)
User Exit 8	UEX8	Operator interface
User Exit 9	UEX9	Data unload (ADAULD)
User Exit 11	UEX11	Command processing (Adabas nucleus)
User Exit 12	UEX12	Multiple log processing

Hyperdescriptor Exits	ADARUN	Use
1 . . 31	HEX01 . . HEX31	User-supplied algorithm to create hyperindex values

Collation Descriptor Exit	ADARUN	Use
1 . . 8	CDX01 . . CDX08	User-supplied algorithm to encode and decode values for the corresponding collation descriptors

SMF User Exit	ADARUN	Use
SMF User Exit	UEXSMF	User-supplied detail section to be included in the SMF record

Conventions

Notation `vrs`, `vr`, or `v`: When used in this documentation, the notation `vrs` or `vr` stands for the relevant version of a product. For further information on product versions, see *version* in the *Glossary*.

Related Literature

Other user exits supported by Adabas include the following:

Caution: These user exits are sample user programs and are not supported under any maintenance contract agreement.

Entry Name	Use
ADACDCUX	Allows you to obtain control at strategic points during ADACDC utility processing. See the *Adabas Utilities* documentation.
ADACSHUX	Allows you to obtain control at strategic points during Adabas Caching Facility processing. See the *Adabas Caching Facility* documentation.
ADASMXIT	Allows you to supply parameters to a PIN routine or examine a condition when it is encountered before the PIN routine is invoked so that recovery actions other than those provided by Adabas can be implemented. See *Adabas Online System Demo Version* in the DBA Tasks documentation..
DSFUEX1	Automatically submits the necessary job to prevent overflow of the DLOG area. See the *Adabas Delta Save Facility Facility* documentation.
LUEXIT1	Linked with the Adabas link globals table and receives control before a command is passed to a target. For more information regarding Adabas link module user exit 1, refer to Link Routine User Exit 1 (Pre-Command) and User Exit 2 (Post-Command)in the Preparing Adabas Link Routines section of the Installation for z/OS chapter.
LUEXIT2	Linked with the Adabas link globals table and receives control after a command is processed by a target, the Adabas SVC, or the link module itself. For more information regarding Adabas link module user exit 2, refer to Link Routine User Exit 1 (Pre-Command) and User Exit 2 (Post-Command)in the Preparing Adabas Link Routines section of the Installation for z/OS chapter.
UEXRAI	Allows you to change automatically generated ADARAI RECOVER JCL before it is written to DDJCLOUT. See the *Adabas Utilities* documentation.

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://tech-community.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

2 User Exit Calling Conventions

- Standard MVS Calling Conventions 6
- User Exit Invocation 6
- User Exit Processing 6
- Return from User Exits 7

Standard MVS Calling Conventions

Please refer to the IBM documentation, z/OS MVS Programming: Assembler Services Guide, Linkage conventions, Saving the calling program's registers. The following is a link to the documentation for z/OS 3.1:

https://www.ibm.com/docs/en/zos/3.1.0?topic=conventions-saving-calling-programs-registers

If a user exit routine uses AMODE64 or G-type instructions, it must preserve the high halves (bits 0-31) of general registers 2-14.

User Exit Invocation

User exit routines are invoked with the following attributes:

- 31-bit addressing mode (AMODE31)
- primary space mode
- TCB mode
- the same PSW key that Adabas was invoked with.

Additionally, user exit routines may be invoked in either problem state or supervisor state, depending on the configuration of Adabas.

User Exit Processing

While a user exit is processing, Adabas is waiting for the user exit to return control. During that time, Adabas cannot execute any commands it is sent. User exits should not perform complex operations that require a response from some other server, such as another Adabas database. In particular, user exits must not issue commands to the Adabas server that calls them, as this would directly lead to a deadlock.

Return from User Exits

All user exits must return the same program status word (PSW) fields to the calling program that were active when the user exit was called. In particular this applies to the following:

- addressing mode (AMODE)
- program mask
- problem state flag
- PSW key
- PSW key mask (PKM)
- address space control setting.

The condition code need not be preserved.

If any of these PSW fields is changed by the user exit, one way to ensure that their previous values are returned is to envelope the code where the change is in effect with a pair of the BAKR ... PR instructions. If BAKR ... PR instructions are not necessary, return using BSM 0,R14 after restoring all registers except for R15.

Refer to supplied sample exit UEX12 for more details on BAKR/PR.

3 User Exit 1 (General Processing)

With the introduction of user exit 11, support for user exit 1 is dropped. However, to ease the migration, a sample user exit, UEX11UX1, is supplied that you can insert in front of your existing user exit 1 to have it invoked as user exit 11. This sample will only work for direct calls made using the ACB direct call interface; it will not work for direct calls made using the Adabas 8 ACBX direct call interface. The exit is still subject to exit 11 constraints, as described in ***User Exit 11***, elsewhere in this guide. In particular, changes are allowed only to the file number (CQXFNR), Additions 2 (ACBADD2), Additions 3 (ACBADD3), and user area (ACBUSER) fields. The nucleus will ignore changes in any other ACB field and all other changes to the CQX. Please refer to comments in the sample user exit for more details, including how to link it with an existing user exit 1. Adabas nucleus support for this transition aid will be withdrawn in a future Adabas release.

Caution: UEX11UX1 is a sample user program and is not supported under any maintenance contract agreement.

4 User Exit 2 (Dual Log Processing)

- User Exit 2 Calling Sequence 13
- Input Parameters 14
- Output Parameters 16

This user exit is given control by the Adabas nucleus during a switch from one dual log to the alternate dual log for the purpose of copying the log before it is reused by Adabas. This switch occurs only if dual data protection logging or dual command logging is in effect for the session.

For more information regarding ***Standard MVS Calling Conventions*** and ***Return from User Exits*** please refer the respective sections in the *Overview*.

Note: UEX2 and UEX12 are mutually exclusive for an Adabas nucleus session: only one can be specified.

The user exit routine must invoke a procedure whereby the appropriate function of the ADARES utility (CLCOPY or PLCOPY) is executed.

User exit 2 is invoked:

- during Adabas nucleus startup if a PLOG/CLOG has to be copied;
- whenever a dual command or dual protection log switch occurs between two log data sets;
- at the end of a PLCOPY or CLCOPY job if ADARES determines there are more copies needed;
- during Adabas nucleus shutdown.

The user exit is provided with information about the status of the dual log data sets.

The user exit can decide which action is to be taken:

- Ignore the call;
- Submit a job to copy the log data set just filled up (ADARES utility);
- Wait for completion of the copy job just submitted.

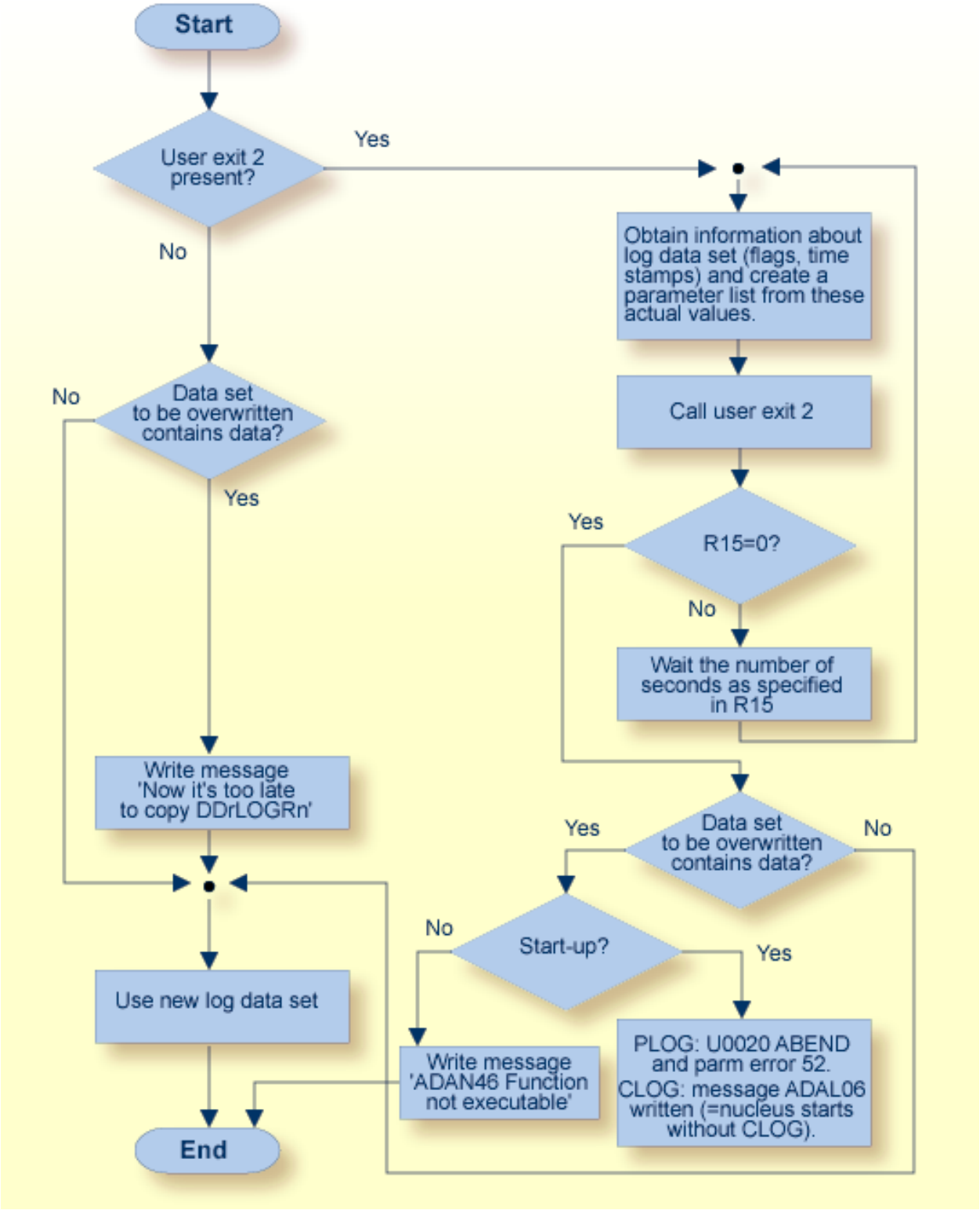
Note: If automated CLOG merge is being used in a cluster environment, it is critical that the exit 2 is used in the suggested manner to copy the CLOGs in a timely fashion as illustrated in the sample exit. Invoking the CLCOPY process in a different manner can result in time stamp inconsistencies between the CLOG datasets in a cluster environment causing CLOG merge issues. The PLOG merge is always automatic and also requires that the PLOGs are copied in a timely manner.

If the data set to be overwritten contains data, console message ADAN46 Function not executable is issued.

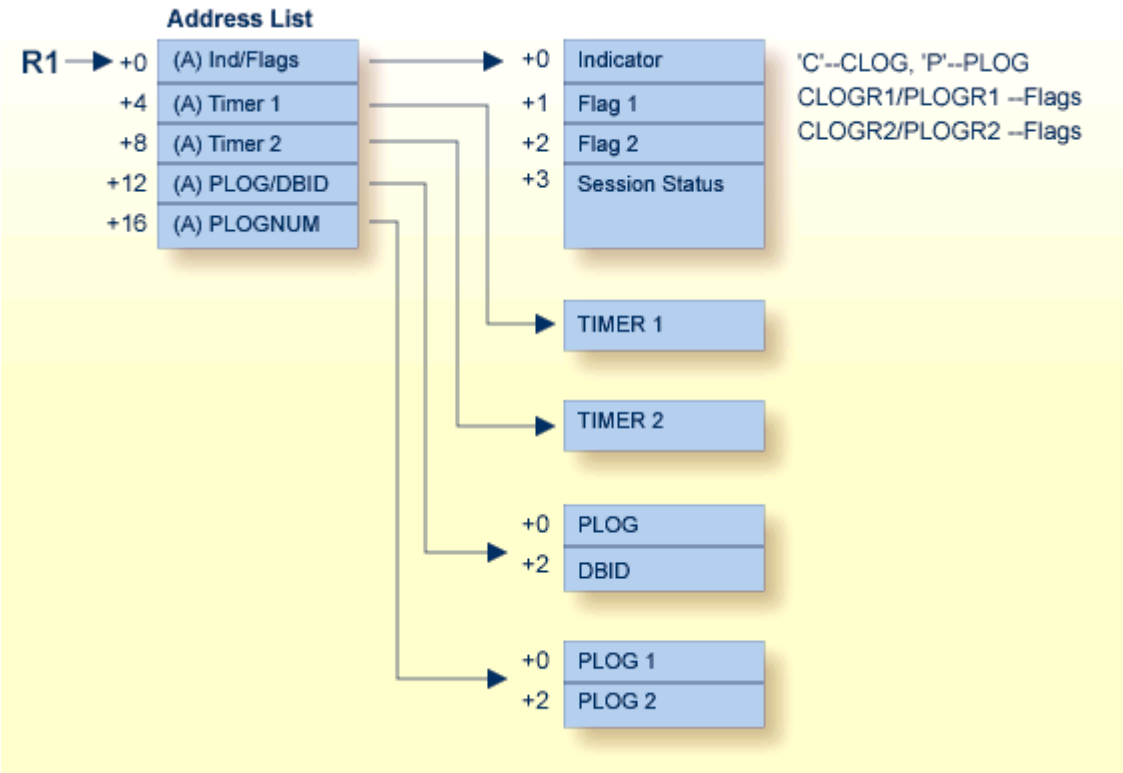
An example of user exit 2 is supplied with the Adabas installation procedure. Refer to the *Adabas Installation* documentation for more information.

The call to the user exit is made using a standard BASR 14,15 Assembler instruction. All registers must be saved when control is received and restored immediately prior to returning control to Adabas. Register 15 contains an action code as described in ***Output Parameter***, elsewhere in this chapter.

User Exit 2 Calling Sequence



Input Parameters



The input parameters for the address list are as follows:

Parameter	A fullword address of . . .
0 (R1)	the C/PLOG indicators with Flag 1 and Flag 2.
4 (R1)	the four-byte timer 1 field.
8 (R1)	the four-byte timer 2 field.
12 (R1)	the current session's PLOG number, followed by the database ID.
16 (R1)	a four-byte area where the first two bytes contain the number of PLOG1, and the second two bytes hold the number of PLOG2.

Other input parameters are explained in the following table:

Parameter	Usage	
Flag 1	Status flags for DDPLOGR1 and DDCLOGR1; and	
Flag 2	Status flags for DDPLOGR2 and DDCLOGR2:	
	B'1... ....' :	Data set being written by nucleus
	B'.1.. ....' :	Data set has been completed by nucleus
	B'.11. ....' :	Being copied by ADARES
	B'…. .1..':	Records include time stamp (PLOG) CLOG not merged (Cluster)
	B'0000 0000' :	Data set is empty (or copied) and reusable for the nucleus.
	All other flag field values are reserved. For DDCLOGR1/2 only: X`08' for CLOGLAYOUT=5. Flag bit settings can be combined (X`40' and X`20' as X`60', for example). If OPENOUT is specified, these flags are set after OPEN is issued for the output data set; otherwise, the flags are set before the OPEN is issued.	
Session Status	Contains information about the status of the nucleus when the exit was called:	
	X'S'	Called during nucleus session startup.
	X'T'	Called while terminating the nucleus session.
	X'W'	Called following a dual protection log switch.
TIMERn	Time-stamp (highest four bytes of a STCK instruction) for the time the first block of the log data set has been written. TIMER1 for DDPLOGR1 and DDCLOGR1, and TIMER2 for DDPLOGR2 and DDCLOGR2	
PLOG	Current session protection log number (two bytes). This value is set for PLOG only; the field contains X`00' for CLOG.	
DBID	Database ID (two bytes).	
PLOG1/2	Two 2-byte PLOG numbers found on PLOG 1 and PLOG 2. If the previous nucleus session ended abnormally, these four bytes contain that session`s PLOGNUM value, which can be used in the initial user exit 2 call to copy that session's PLOG. During any subsequent session, these bytes contain the current PLOGNUM value. If the preceding session ends abnormally, these four bytes contain the ended session's PLOG numbers during the nucleus start phase. This PLOG information is needed during the start phase to assign the correct PLOG numbers to the PLOG areas to be copied. During subsequent exit calls, the current PLOG values are in these fields.	

Output Parameters

Parameter	Usage
R15 = 0	Nucleus continues processing.
R15 > 0	R15 is treated as the number of seconds to wait before calling user exit 2 again. During this time, the nucleus is in a "hard" wait. No commands are processed during the wait.

5 User Exit 3 (User-Defined Phonetic Processing)

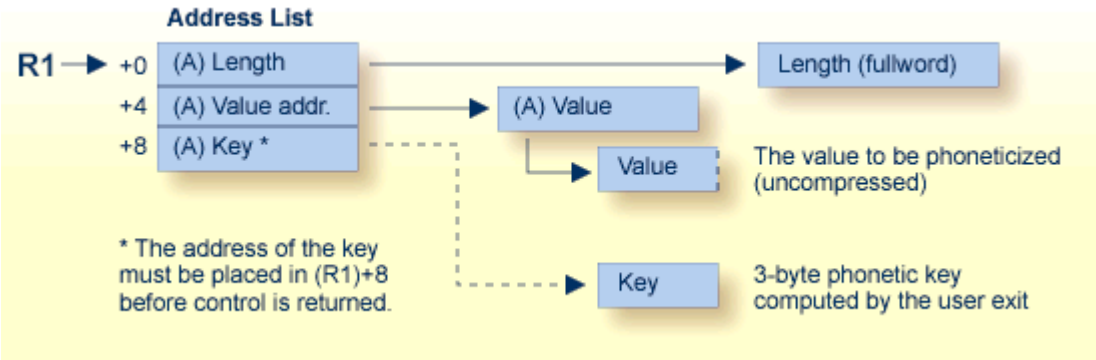
- Input Parameters 18

This user exit may be used to perform user-defined phonetic processing. It is given control by the ADACMP utility or the Adabas nucleus whenever phonetic processing is required.

The user exit must develop a three-byte phonetic key using the value supplied. The address of the resulting phonetic key must be placed at 8(R1) before control is returned.

Input Parameters

Register 1 contains the address of the following parameter list:



Parameter	A fullword address of . . .
0(R1)	the four-byte length for the value to be phonetically processed.
4(R1)	the address of the value to be phonetically processed.
8(R1)	a three-byte location to contain the phonetic key. This address is set to zero before the user exit and must be set to the actual address during the user exit.

The call to the user exit is made using a standard BASR 14,15 assembler instruction. All registers must be saved when control is received and restored immediately prior to returning control to Adabas. The content of R15 is ignored.

6 User Exit 4 (User-Generated Log Data)

- Command Log Format 20
- Invoking User Exit 4 23
- Accessing Adabas Buffers via CQX 24
- Accessing Adabas Buffers via LORECX 25

User exit 4 is called immediately before an Adabas command log record is to be written. It may be used to generate any required user log data (SMF records) special statistics, or to suppress writing a log record.

For more information regarding ***Standard MVS Calling Conventions*** and ***Return from User Exits*** please refer the respective sections in the *Overview*.

Note: User exit 4 is still called even if ADARUN LOGGING=NO and REVIEW is specified. User exit 4 will not be invoked if LOGGING=NO and REVIEW is not active. If REVIEW is specified, the only way to disable user exit 4 is to remove the ADARUN UEX4 parameter from the Adabas run.

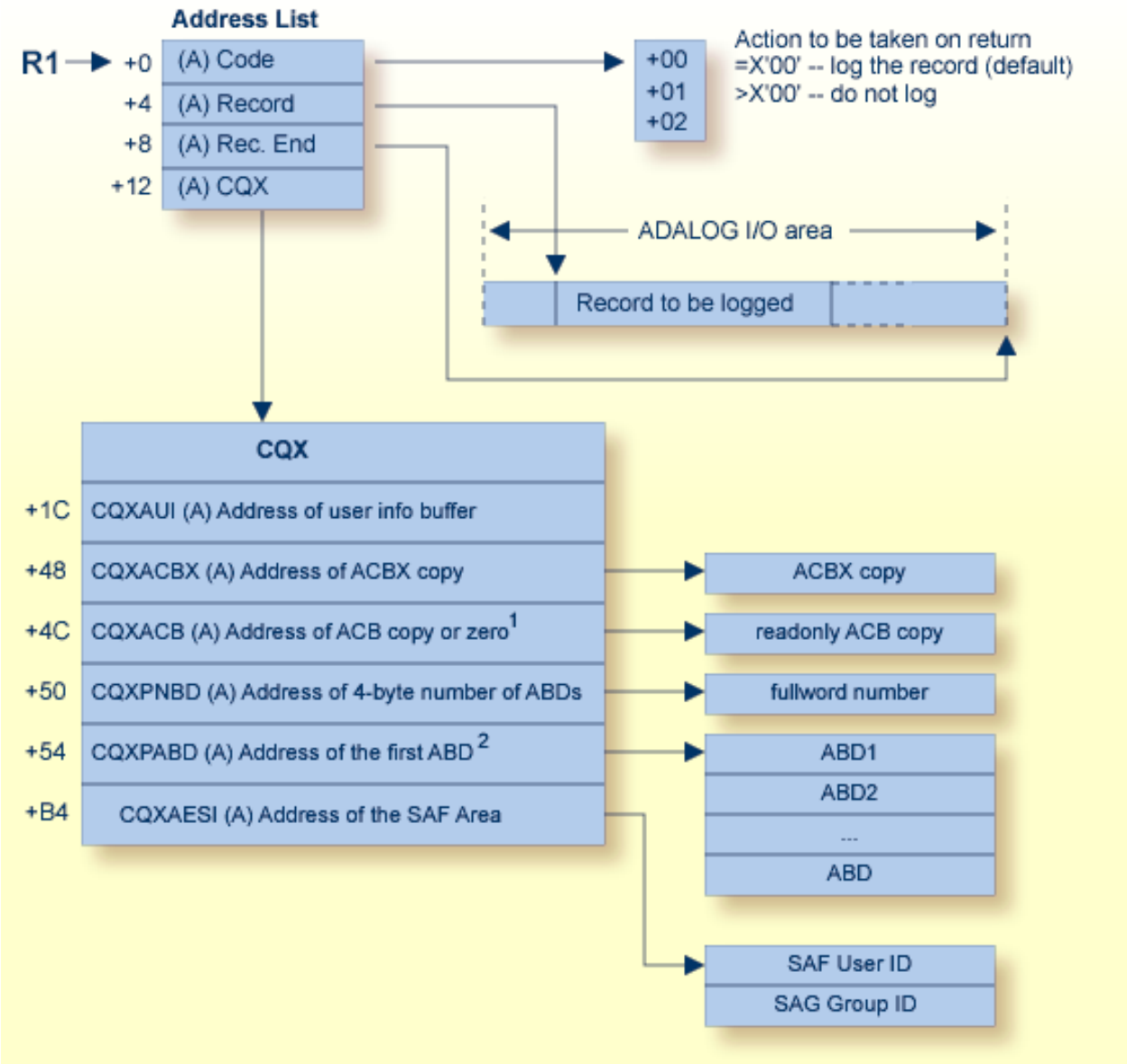
Command Log Format

Adabas supports two different command log formats. The ADARUN CLOGLAYOUT parameter determines which format is used:

- CLOGLAYOUT=5 (the default) is supported only in Adabas versions 5.2 and above.
- CLOGLAYOUT=8 specifies the new format, which is supported only in Adabas versions 8 and above.

Both formats are described in *Command Log Formats*, in *Adabas DBA Tasks Manual*.

Ensure that your user exit and command log evaluation programs recognize the format in use before switching to it.



[1]*Address of ACB Copy*: This address should be set to zero if the command is using an ACBX interface direct call.

[2]*Address of the first ABD*: The Adabas buffer descriptions (ABDs) are in a contiguous array. For complete information about locating ABDs in this array, read ***Locating the Correct ABD***, elsewhere in this section.

Parameter	Address of . . .
0(R1)	a byte containing a logging action code. This byte contains: ■ +00 -- action code to log the record upon each call. If changed to a nonzero value, this record will not be written to DDLOG. ■ +01 -- reserved for future use ■ +02 -- two-byte database ID.
4(R1)	the record to be logged. This address is zero if the exit is called at the end of the nucleus session.
8(R1)	the end of the Adabas I/O area. This address is zero if the exit is called at the end of the nucleus session.
12(R1)	the command queue element (CQX). This address is zero if the exit is called at the end of the nucleus session.

The record to be logged may be modified by the user exit. The record's address in 4(R1) may also be modified. The logging action code must always be specified before returning to the Adabas nucleus.

Caution: When modifying the record, do not exceed the end address of the ADALOG I/O area contained in 8(R1).

Locating the Correct ABD

Internally, Adabas 8 only uses extended Adabas control blocks (ACBX) and Adabas buffer descriptions (ABDs). Direct calls made using the classic Adabas control block (ACB) and buffer definitions have their data structures converted to ACBX calls and ABDs by ADASVC before the nucleus sees the call. Thus, the protocol for locating and accessing buffers in user exits, such as this one, has changed as of Adabas 8.

The Adabas buffer descriptions (ABDs) are now in a contiguous array. However, the internal representation of the ABD may not have the same length as the base ABD, as defined by the value of the ABDXQLL symbol in the ADABDX DSECT, although the first ABDXQLL bytes continue to be mapped by ADABDX. This means that you should not use the ABDXQLL value in the ADABDX DSECT to locate the next ABD in the ABD array. Instead, you should use the value of the two-byte ABDXLEN field at offset `+x'00'` of the ABD to determine the end of that ABD and the start of the next ABD in the array. Do not assume that all internal ABD representations have the same length: each must be located in turn by applying its predecessor's ABDXLEN value.

In addition, the order of the ABDs is not defined and my change over time or from command to command, although within the array all ABDs of a given type (format buffer, record buffer, etc.) are contiguous. There will be an ABD for every buffer provided by the user that is documented as an input or output buffer for the specific command. There may also be additional buffers created by other components. When there are multiple instances of format, record and (optional) multifetch buffers, they are related based on their position: the first format buffer is associated with the first record (and optional multifetch) buffer, the second with the second, and so forth. If the caller

provides an unequal number of format, record and (optional) multifetch buffers, dummy descriptors with a zero buffer length are created to bring about equal quantities. When multifetch is used with a classic ACB call, certain commands (L1/2/3/4/9) will have their ISN buffer converted into a multifetch buffer. Here are some examples:

- If a caller (using either an ACB or ACBX call) issues an OP command and provides a record buffer and search buffer, the array of ABDs will have one record buffer ABD and one dummy format buffer ABD (to satisfy the internal requirement that there be equal numbers of format and record buffers). There is no ABD for the search buffer because that is not a documented input or output buffer for the OP command.
- If a caller uses an ACBX call to issue an L1 command and provides two format buffers and three record buffers, the array of ABDs will have three record ABDs and three format ABDs, the last one of which is a dummy format ABD. The first record buffer is associated with the first format buffer; the second record buffer is associated with the second format buffer; and the third record buffer is associated with the third (dummy) format buffer.
- Suppose a caller uses an ACB call to issue an L3 command with Command Option 1 set to "M" (multifetch) and Command Option 2 set to "A" (ascending retrieval from a specified value). In addition, the caller provides a format buffer, a record buffer, an ISN buffer, a search buffer and a value buffer. In this case, the array of ABDs will have one format buffer ABD, one record buffer ABD, one multifetch buffer ABD, one search buffer ABD, and one value buffer ABD. The caller's ISN buffer will have been converted to a multifetch buffer.

Invoking User Exit 4

User exit 4 is invoked for every record which is written to the CLOG. The kind of record can be determined by field LOXTYPE of structure LORECX (`<R1> + 4` => LORECX).

The set of records written to CLOG depends on the Adabas command (valid buffer types) and the corresponding ADARUN parameters (LOGGING, LOGABDX, LOGCB, LOGFB, LOGIB, etc. referred to as LOG*xxx* later).

Sample

```
ADARUN LOGGING=YES,LOGCB=YES,LOGRB=YES,LOGSB=YES
OP command with RB="ACC=10."
```

Will result in invocations of UEX4 with:

1. Basic record including control block (`LOXTYPE= x'0001'`)
2. ABDX + record buffer (`LOXTYPE = x'0008'`)

Accessing Adabas Buffers via CQX

<R1> + x'0C'	CQX structure (Command Queue Element info for Adabas user exits)
CQX + x'48'	ACBX structure
CQX + x'50'	number of ABDs
CQX + x'54'	array of ABDs, length of array element is ABDXLEN (i.e. step forward to next array element by adding its length ABDXLEN)

Depending on the value of ADBXLOC, you get to the buffer itself. In this scenario the value of ABDXLOC is 'I' and ABDXADR contains the address of the buffer. It points directly to the value; there is no preceding length field.

The length of the buffer can be obtained from

- ABDXSIZE the allocated length of the buffer
- ABDXSEND the length of data send to Adabas
- ABDXRECV the length of data received from Adabas

The simplest way is to use `"LENGTH = max(ABDXSEND,ABDXRECV)"` but make sure this does not exceed ABDXSIZE.

Sample accessing Adabas buffers via CQX

Assuming the LOG*xxx* ADARUN parameters are set accordingly.

```
If LORECX.LOXTYPE eq LOXTBAS then                    /*? Adabas basic record
  "process ACBX pointed by CQX.CQXACBX"
  #n := pointed by CQX.CQXPNBD                       /* number ABDX
  If #n gt 0 then                                    /*? Any  ABDX present
    #s := pointed by CQX.CQXPABD                     /* address 1st ABDX
    For #i := 1 to #n                                /* iterate for all ABDX
      #l := max (#s.ABDXSEND, #s.ABDXRECV)           /* l'data
      If #l > 0
        Then "process  ABDX pointed by CQX.CQXPABD"
      End-if
      #s := #s + #s.ABDXLEN                          /* step to next ABDX
    End-for
  End-if
End-if
```

Accessing Adabas Buffers via LORECX

When accessing the Adabas buffers via LORECX structure one has to "listen" for records with LOXTYPE:

- **x'0001'**
  the basic record containing the ACBX (optional) and the fields LOX1TYP1 and LOX1TYP2 which define and illustrate which "data records" (LOXTYPE= x'0008' and x'0009') will follow.
- **x'0008'**
  ABDX and buffer
- **x'0009'**
  buffer continued (in case the data overflows the record length)

7 User Exit 5 (Adabas Review Hub Event Handler)

- Input Parameters 28
- Output Parameters 29

User exit 5 is called by the Adabas nucleus when an *event* occurs with the Adabas Review hub.

An event is defined as

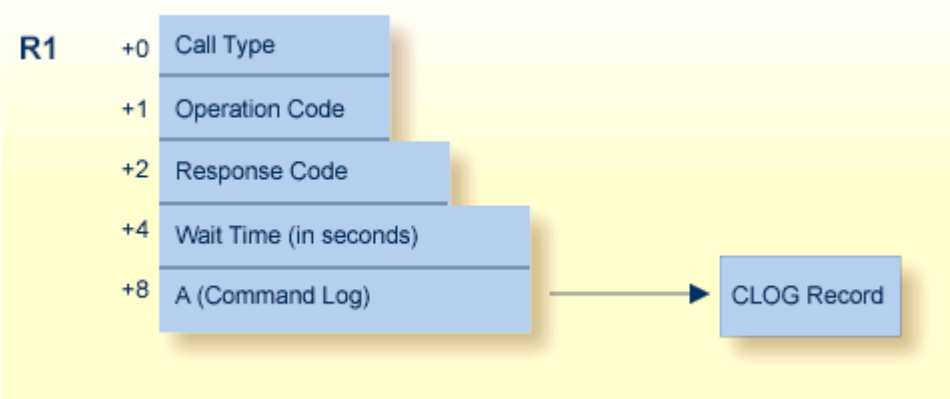
- a connection made with the Adabas Review hub during Adabas session open;
- a connection broken with the Adabas Review hub during Adabas session close; or
- a non-zero return code received from the send operation for a command log record.

The exit is invoked with AMODE=31 and should return control in the same state.

The exit is required to process logging errors. It determines how the failure is handled. The record that was not logged and the response code received from the Adabas Review hub logging request are provided to assist in making the determination.

Input Parameters

On entry, the register 1 points to the following parameter list:



Parameter	Usage	
0(R1)	Exit call indication. The value of this byte can be:	
	O	connection with Adabas Review hub opened;
	C	connection with Adabas Review hub closed; or
	L	sending logging error to Adabas Review hub.
1(R1)	Action to handle a logging error (ignored for open and close). The exit must provide one of the following values for this field in the parameter list for a logging error:	
	W	wait a specified time and then retry;
	R	retry logging operation immediately; or
	I	ignore the logging failure and continue without consequence.

Parameter	Usage
2(R1)	Response code for logging errors. This response code is the same as the Adabas response code found in the *Adabas Messages and Codes*.
4(R1)	Fullword where the exit must provide a wait time (in seconds) for the logging failures that are to be retried after waiting.
8(R1)	Address of the command log record that the Adabas nucleus was attempting to send to the Adabas Review hub.

Other Register Values at Entry

R13	save area of calling Adabas nucleus routine
R14	return address in Adabas nucleus
R15	entry point address for exit

Output Parameters

- For logging errors, the exit is required to set a value in the `operation' field. If the wait value (W) is chosen, the exit is also required to provide a non-zero time value.
- Register 15 should be set to zero. All other registers should be returned intact.

8 User Exit 6 (User Processing Before Data Compression)

- ADACMP Header Processing 34
- Input Parameters 35
- Output Parameters 35

This user exit can be used to perform user processing on a record before it is processed by the ADACMP COMPRESS utility. It can also be used to control the sequence and contents of the decompressed records that are output from the ADACMP DECOMPRESS utility; when used in this way, the user exit controls which decompressed records ADACMP writes to the DDAUSBA data set.

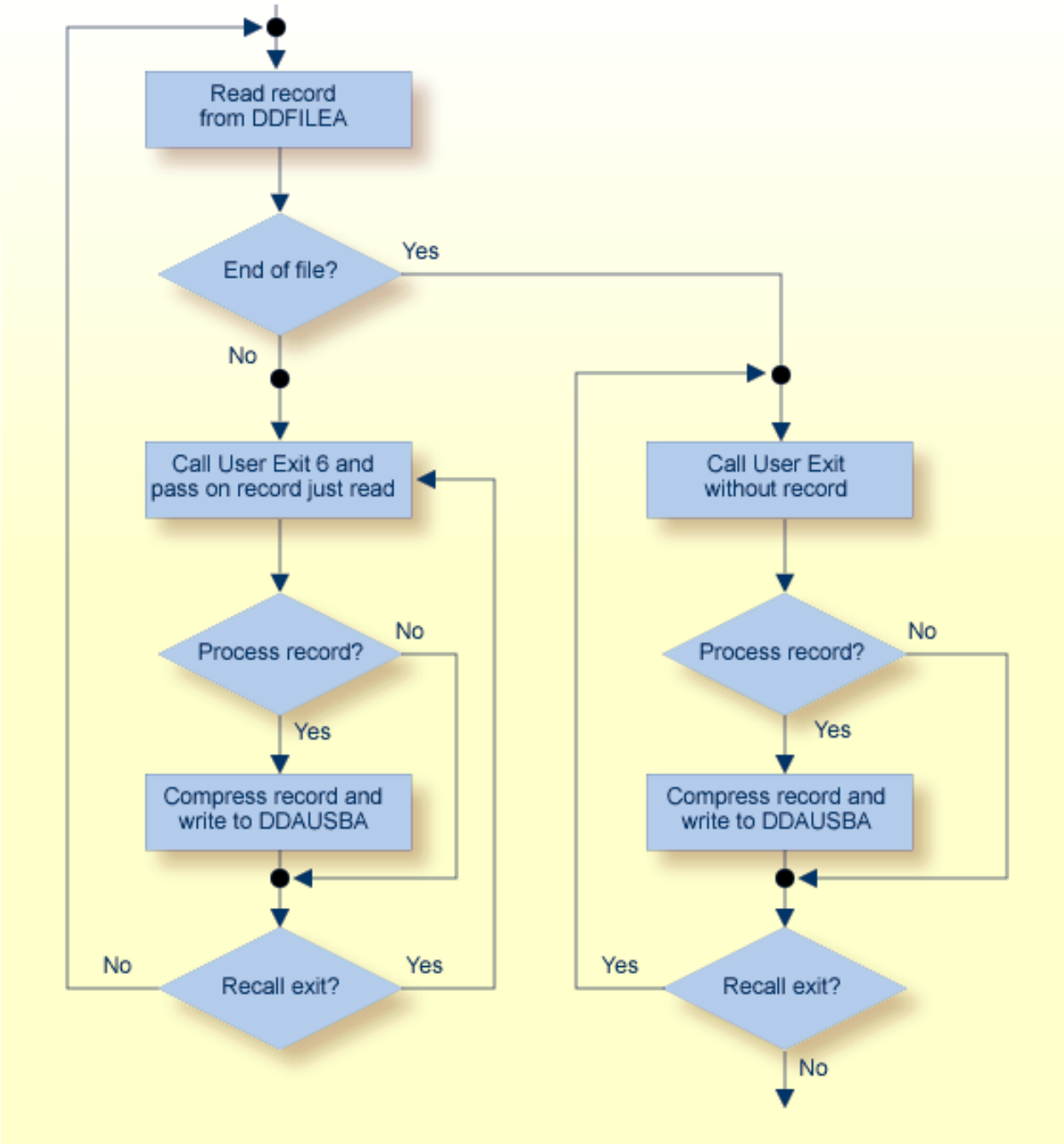
Sample user exit 6 assembler language source is supplied in the Adabas source library in member USEREX6A (Assembler). A sample job to assemble and link the user exit is supplied in member ASMUEX6 of the Adabas sample job library.

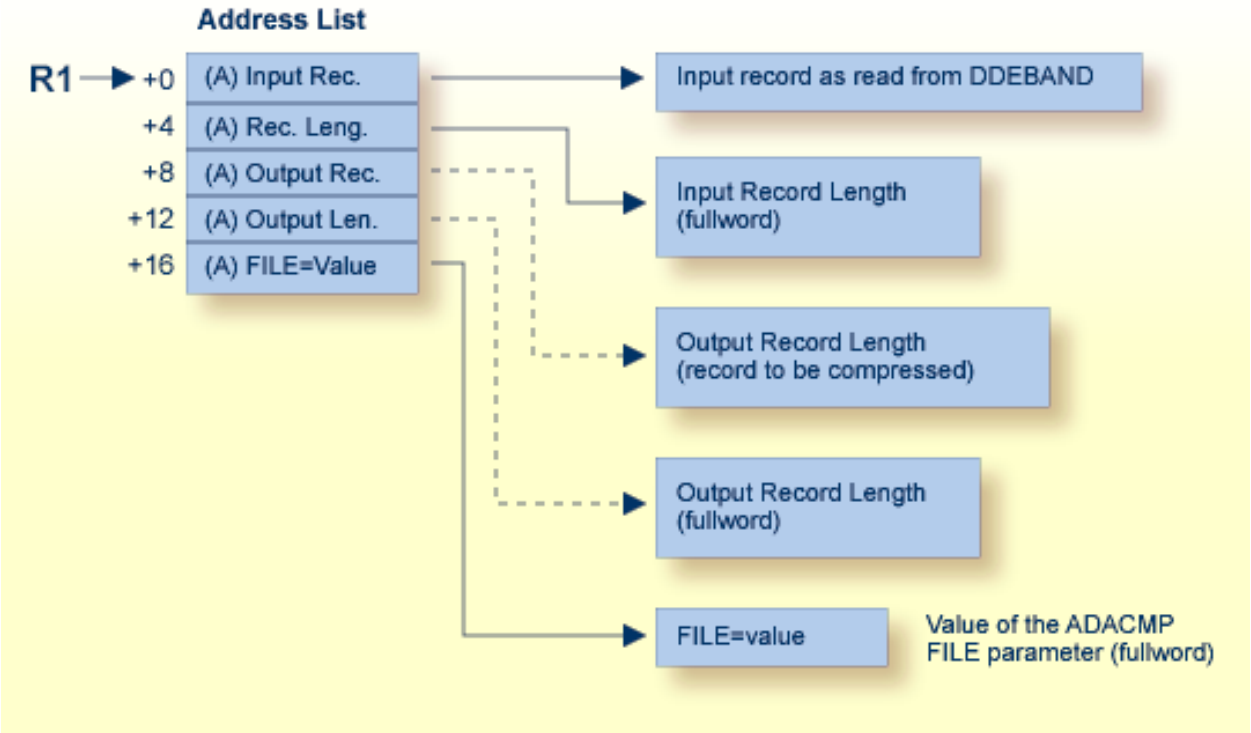
Caution: Sample user exits and programs and are not supported under any maintenance contract agreement.

The ADACMP utility job must specify `ADARUN UEX6=`*`program parameter`*, where *`program`* is the name of the user program and *`parameter`* is a parameter passed to that program.

User exit 6 is called by the ADACMP COMPRESS utility function immediately after one of the following occurs so that it can append records to the input:

- A record has been read from DDEBAND.
- An end-of-file condition has occurred on DDEBAND.


Read record
from DDFILEA
End of file?
Yes
No
Call User Exit 6 and
pass on record just read
Process record?
No
Yes
Compress record and
write to DDAUSBA
Recall exit?
No
Yes
Call User Exit
without record
Process record?
No
Yes
Compress record and
write to DDAUSBA
Recall exit?
Yes
No



ADACMP Header Processing

When ADACMP is run with the parameter HEADER=YES, all input records for ADACMP COMPRESS and output records for ADACMP DECOMPRESS are preceded by 32-byte ADAH and ADAC headers that describe the grouping of physical records into logical records that may be larger than 32 KB. DSECTs for the ADAH and ADAC headers can be found in members ADAH and ADAC in the distributed Adabas 8 SRCE library. These headers identify how one logical record containing uncompressed data is composed of one or more physical records.

When ADACMP is run with user exit 6 and HEADER=YES, ADACMP will pass each physical record to user exit 6. The user exit application may need to use these headers to determine the relationship between the physical and logical records.

For more information about the ADACMP headers and record segmenting, read *Segmented Record Considerations*, in *Adabas Utilities Manual*.

Input Parameters

Parameter	Usage
0 (R1)	Address of an input record. The length field preceding the variable record is skipped. The address is of a fullword containing -1 (X`FFFF FFFF') if the user exit is called after ADACMP detects end-of-file in DDEBAND.
4 (R1)	Address of the field containing the input record length. For fixed records, this is a logical record length. For variable records, this is the length of the actual data only (excluding the length field itself). The address points to a fullword containing minus 1 (X`FFFFFFFF') if the user exit is called after ADACMP detects end-of-file in DDEBAND.
8 (R1)	Contains binary zeros on entry to the user exit (see ***Output Parameters***).
12 (R1)	Contains binary zeros on entry to the user exit (see ***Output Parameters***).
16 (R1)	Address of the FILE parameter value specified by the ADACMP COMPRESS utility job. The address is in the rightmost/low-order two bytes. The location and content of this fullword must remain unchanged during the time of the user exit. If ADACMP COMPRESS did not specify the FILE parameter, the fullword is X`00000000'.

Output Parameters

Parameter	Usage
8 (R1)	Address of the user exit output record. This record will be used as input to the ADACMP compression algorithm. The address of this record must be placed into 8 (R1) each time the user exit is called. If this field contains binary zeros on return, ADACMP will ignore the input record and will continue processing.
12 (R1)	Address of a 4-byte field containing the length of the returned record. The address of this field must be placed into 12 (R1) each time the exit is called. If this field contains binary zeros on return, ADACMP will ignore the record and will continue processing. Though the length field pointed to by 12 (R1) has a length of 4 bytes, only the low-order/rightmost halfword is used (bytes 3 and 4). If byte 2 contains a X`01' on return, the exit is recalled before the next record is read from DDEBAND. This enables the user to return more than one record to ADACMP for each record read from DDEBAND.

9 User Exit 8 (Operator Interface)

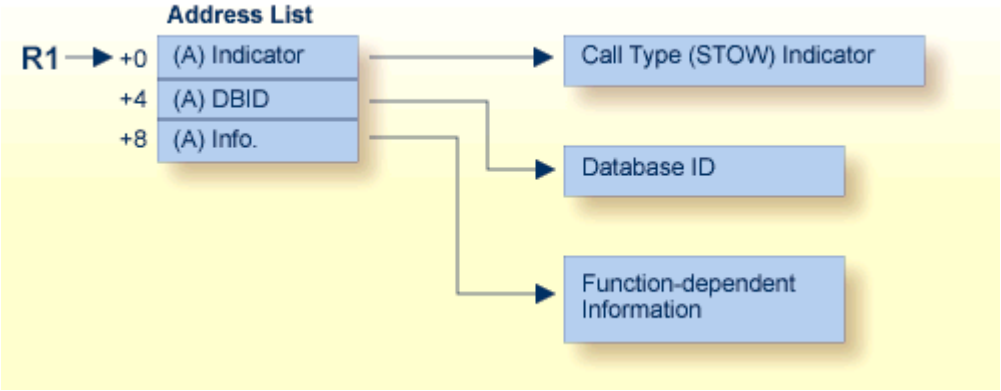
- Input Parameters 39

This user exit receives control from the Adabas nucleus whenever the nucleus starts or stops, or whenever the nucleus or an Adabas utility receives a message from or sends a message to the operator. User exit 8 can be used to provide specific instructions to the operator

- when the nucleus starts and (normally) stops operation;
- as added information when Adabas sends console messages to the operator;
- to confirm commands entered by the operator.

User exit 8 is invoked:

- (MODE=MULTI only) after Adabas startup, as soon as the nucleus is able to answer calls from user programs. At this point, the nucleus is now active.
- immediately after the Adabas nucleus or utility issues a console operator message. The user exit call is in addition to the standard message processing; the message itself cannot be changed during the user exit.
- after the Adabas nucleus or utility receives an operator command. The exit is called before the command is actually processed, and can reject or replace the command. The command cannot be modified in its original area.
- before a normal Adabas nucleus stop. At this point, the nucleus is no longer active; any more nucleus calls result in response code 148 (ADARSP148). This exit is not called if the nucleus ends abnormally.



Input Parameters

Parameter	Usage	
0 (R1)	Address of the byte containing the call type ("STOW") indicator:	
	S	called at nucleus start
	T	called at normal nucleus termination
	O	called with an operator message to the nucleus/utility
	W	called with a nucleus/utility message to the operator
4 (R1)	Address of the fullword containing the database ID.	
8 (R1)	Address of variable-length message-related information for "O" and "W" type calls. The information at this address has the following format: Call format:	
	O	The one-byte message length, followed by the alphanumeric message. The length excludes the length byte itself. If the message is to be changed, location 8(R1) must point to the new message on return. This message is structured as described above. If the message is to be suppressed, location 8(R1) must point to a two-byte field containing X'0140'.

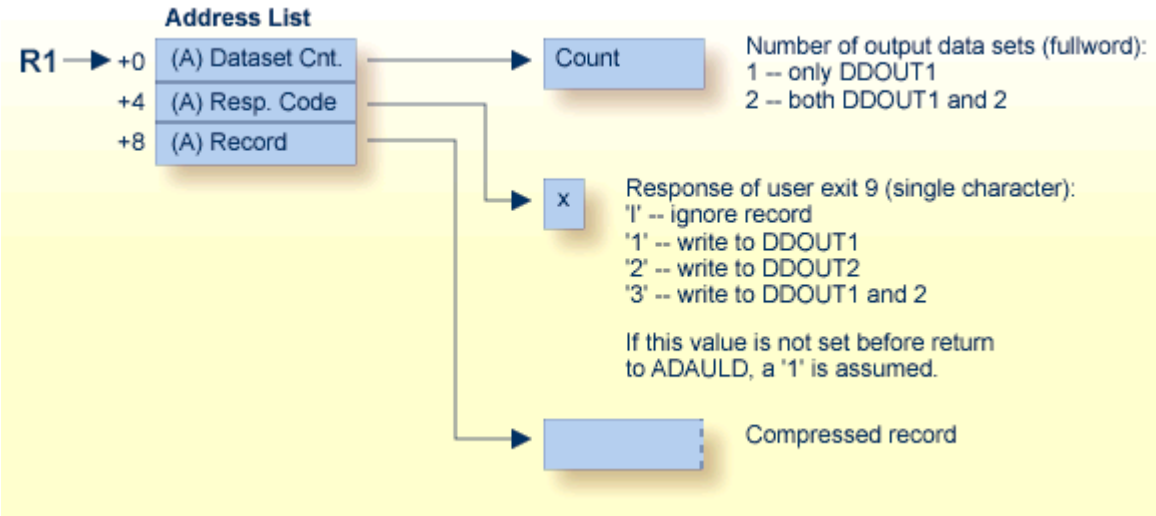
	W	8(R1) points to the message, which has the following structure:
		`DC H'message-length+4'`
		`DC H'0'`
		`DC C'message-text'`

10 User Exit 9 (ADAULD)

- Processing 42
- User Exit 9 Sample 44

User exit 9 is called by ADAULD whenever a compressed record is ready to be written. The user exit decides whether a record is written to DDOUT1, DDOUT2, both, or neither.

Processing

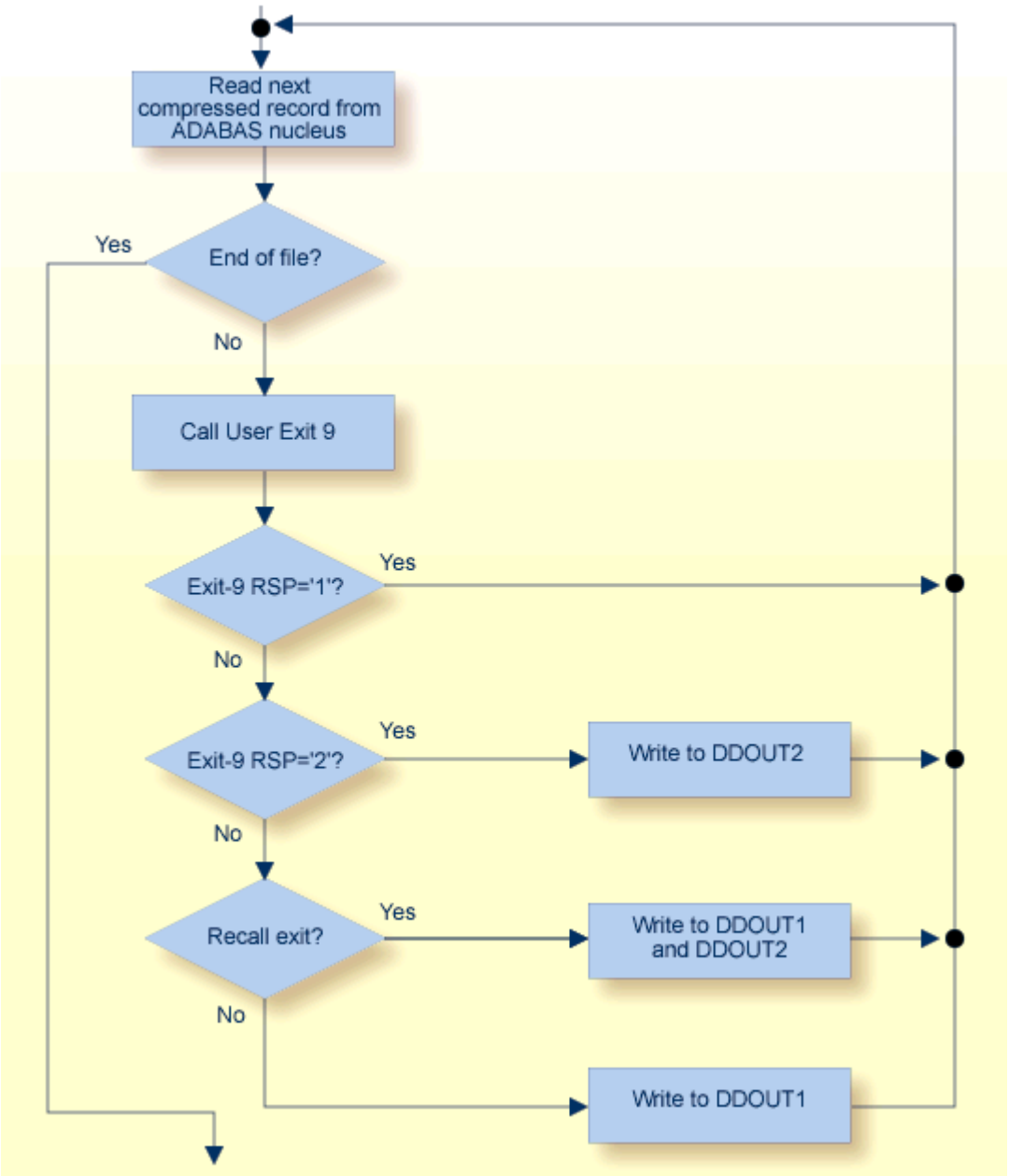


Notes:

1. DDOUT1 & 2 must have the same block size, or an ADAULD error occurs.

The compressed record pointed to by the third address has the following structure:

```
AL2 (L1) total length (inclusive)
AL2 (L2) record length (inclusive)
AL4 (ISN)
XL (L2 - 8) '...compressed fields...'
XL (L1 - L2 - 2) '...DVT entries...'
```



The call to the user exit is made via a standard BASR 14,15 assembler instruction. All registers must be saved when control is received and restored immediately prior to returning control to ADAULD. The content of R15 is ignored.

User Exit 9 Sample

Sample user exit 9 source is supplied in the Adabas source library in member USEREX9. A sample job to assemble and link the user exit is supplied in member ASMUEX9 of the Adabas sample job library.

Caution: Sample user exits and programs and are not supported under any maintenance contract agreement.

11 User Exit 11 (General Processing)

- Input and Output Parameters 47

This user exit is given control by Adabas immediately after a command is received by the Adabas nucleus. The command itself has yet to be processed except for the determination of the type of command (simple access, complex access, update).

One of the most common applications of this user exit is to insert a security password or a cipher code into the ACBX.

This user exit functionality largely matches that of the classic user exit 1, except for the fact that edited copies of the CQX and ACBX data structures are used during user exit 11 processing, rather than the actual structures used by user exit 1. In addition, support for user exit 1 is dropped in Adabas 8 (or later).

Only certain fields in the ACBX may be changed by the exit: ACBXFNR (file number), ACBXADD3 (Additions 3), ACBXADD4 (Additions 4), ACBXCOP1 through ACBXCOP8 (command options 1-8) and ACBXUSER (user area). The nucleus will ignore changes in any other ACBX fields and all changes to the CQX. DSECT EX11PARM maps the user exit 11 parameter list. In addition, a sample user exit 11 skeleton called UEX11 is provided. Both the DSECT and the user exit skeleton are provided in the Adabas source library.

Caution: UEX11 is a sample user program and is not supported under any maintenance contract agreement.

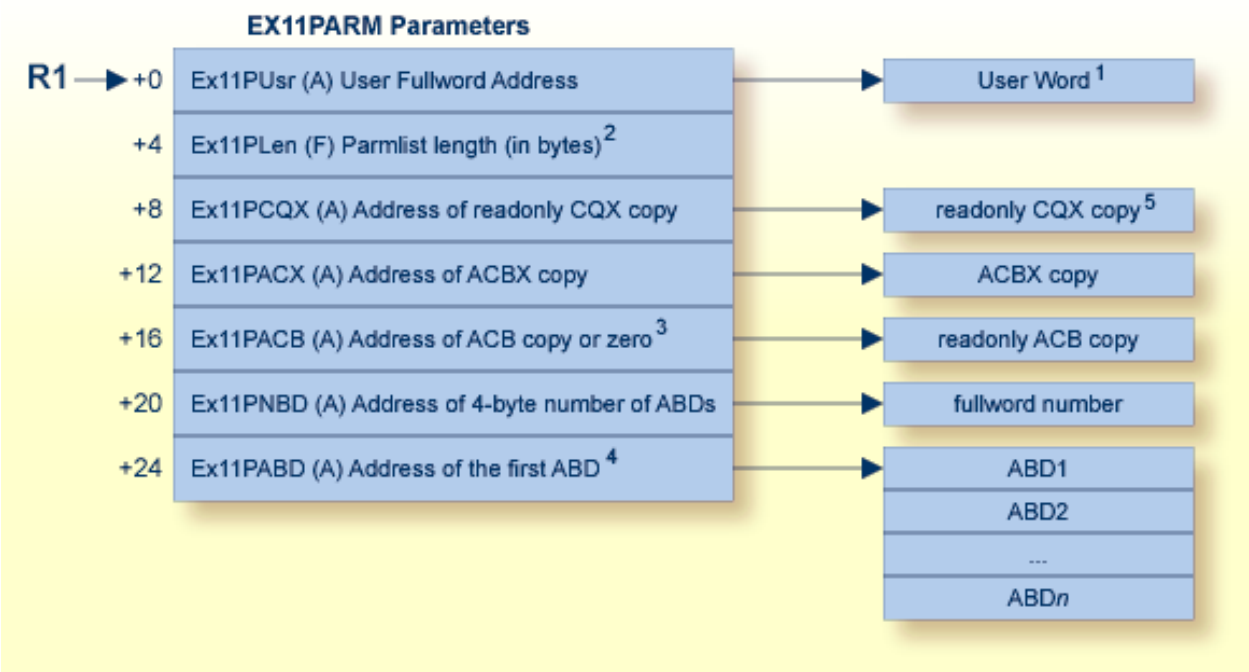
The call to the user exit is made using a standard BASSM R14,R15 assembler instruction. Register 1 contains the address of a parameter list. All registers must be saved when control is received and restored immediately prior to returning control to Adabas, with the exception of Register 15 which contains the return code. A non-zero value means that the command should not be executed and returns response code 22 (ADARSP022) subcode 6. However, the exit may set a response code in the user-defined range 231-239 in field ACBXRSP of the exit's ADACBX copy. When that is the case the response code is taken from ACBXRSP and the subcode from ACBXErrC.

For more information regarding ***Standard MVS Calling Conventions*** and ***Return from User Exits*** please refer the respective sections in the *Overview*.

Notes:

1. The length of an Adabas buffer in any Adabas buffer description (ABD) used by the call cannot be changed.

Input and Output Parameters



[1]*User Word*: Before calling user exit 11, the fullword reserved for the user is set to zero. It is not altered by Adabas between UEX11 invocations. It may be used for any purpose, typically to retain the address of storage acquired for the exit's workarea.

[2]*Parmlist length*: The EX11PARM parameter list length is at least 28 bytes.

[3]*Address of ACB Copy*: This address will be set to zero if the command originated using an ACBX direct interface call.

[4]*Address of the first ABD*: The Adabas buffer descriptions (ABDs) are in a contiguous array. For complete information about locating ABDs in this array, read *Locating the Correct ABD*, next in this section.

[5]*Readonly CQX copy*: For a more detailed diagram of the readonly CQX copy, see the diagram for user exit 4 in ***User Exit 4 (User-Generated Log Data)***, elsewhere in this guide.

Locating the Correct ABD

Internally, Adabas 8 only uses extended Adabas control blocks (ACBX) and Adabas buffer descriptions (ABDs). Direct calls made using the classic Adabas control block (ACB) and buffer definitions have their data structures converted to ACBX calls and ABDs by ADASVC before the nucleus sees the call. Thus, the protocol for locating and accessing buffers in user exits, such as this one, has changed as of Adabas 8.

The Adabas buffer descriptions (ABDs) are now in a contiguous array. However, the internal representation of the ABD may not have the same length as the base ABD, as defined by the value of the ABDXQLL symbol in the ADABDX DSECT, although the first ABDXQLL bytes continue to be mapped by ADABDX. This means that you should not use the ABDXQLL value in the AD-ABDX DSECT to locate the next ABD in the ABD array. Instead, you should use the value of the two-byte ABDXLEN field at offset `+x'00'` of the ABD to determine the end of that ABD and the start of the next ABD in the array. Do not assume that all internal ABD representations have the same length: each must be located in turn by applying its predecessor's ABDXLEN value.

In addition, the order of the ABDs is not defined and my change over time or from command to command, although within the array all ABDs of a given type (format buffer, record buffer, etc.) are contiguous. There will be an ABD for every buffer provided by the user that is documented as an input or output buffer for the specific command. There may also be additional buffers created by other components. When there are multiple instances of format, record and (optional) multifetch buffers, they are related based on their position: the first format buffer is associated with the first record (and optional multifetch) buffer, the second with the second, and so forth. If the caller provides an unequal number of format, record and (optional) multifetch buffers, dummy descriptors with a zero buffer length are created to bring about equal quantities. When multifetch is used with a classic ACB call, certain commands (L1/2/3/4/9) will have their ISN buffer converted into a multifetch buffer. Here are some examples:

- If a caller (using either an ACB or ACBX call) issues an OP command and provides a record buffer and search buffer, the array of ABDs will have one record buffer ABD and one dummy format buffer ABD (to satisfy the internal requirement that there be equal numbers of format and record buffers). There is no ABD for the search buffer because that is not a documented input or output buffer for the OP command.
- If a caller uses an ACBX call to issue an L1 command and provides two format buffers and three record buffers, the array of ABDs will have three record ABDs and three format ABDs, the last one of which is a dummy format ABD. The first record buffer is associated with the first format buffer; the second record buffer is associated with the second format buffer; and the third record buffer is associated with the third (dummy) format buffer.
- Suppose a caller uses an ACB call to issue an L3 command with Command Option 1 set to "M" (multifetch) and Command Option 2 set to "A" (ascending retrieval from a specified value). In addition, the caller provides a format buffer, a record buffer, an ISN buffer, a search buffer and a value buffer. In this case, the array of ABDs will have one format buffer ABD, one record buffer ABD, one multifetch buffer ABD, one search buffer ABD, and one value buffer ABD. The caller's ISN buffer will have been converted to a multifetch buffer.

12 User Exit 12 (Multiple Data Set Log Processing)

- User Exit 12 Calling Sequence 52
- User Exit Interface 52
- Output Parameter 55
- Using Sample User Exit 12 55

Note: UEX2 and UEX12 are mutually exclusive for an Adabas nucleus session: only one can be specified.

This user exit is given control by the Adabas nucleus during a switch from one multiple log data set to another for the purpose of copying the log data set before it is reused by Adabas. This switch occurs only if multiple data set data protection logging and/or multiple data set command logging is in effect for the session.

The user exit routine is designed to invoke a procedure that will execute the appropriate function (ALCOPY, CLCOPY or PLCOPY for Audit Log, Command Log or Protection Log) of the ADARES utility.

User exit 12 is invoked

- during Adabas nucleus startup if a multiple PLOG/CLOG/ALOG data set has to be copied;
- whenever a switch to another log data set occurs;
- at the end of a PLCOPY, CLCOPY or ALCOPY job if ADARES determines there are more copies needed;
- during Adabas nucleus shutdown.

The user exit is provided with information about the type of log (PLOG, CLOG or ALOG) and the status of the multiple log data sets.

The user exit can decide which action is to be taken:

- Ignore the call and allow Adabas to proceed;
- Submit a job to copy and mark as empty the log data set just filled (ADARES utility);
- Direct Adabas to wait for a specified interval, then call the user exit again with updated PLOG/CLOG/ALOG data set status information. During the wait interval, no commands that may produce log records for the log type being processed are allowed to proceed.

Note: If automated CLOG merge is being used in a cluster environment, it is critical that the exit 12 is used in the suggested manner to copy the CLOGs in a timely fashion as illustrated in the sample exit. Invoking the CLCOPY process in a different manner can result in time stamp inconsistencies between the CLOG datasets in a cluster environment causing CLOG merge issues. The PLOG merge is always automatic and also requires that the PLOGs are copied in a timely manner.

An example of user exit 12 is supplied with the Adabas installation procedure. Refer to the *Adabas Installation* documentation for more information.

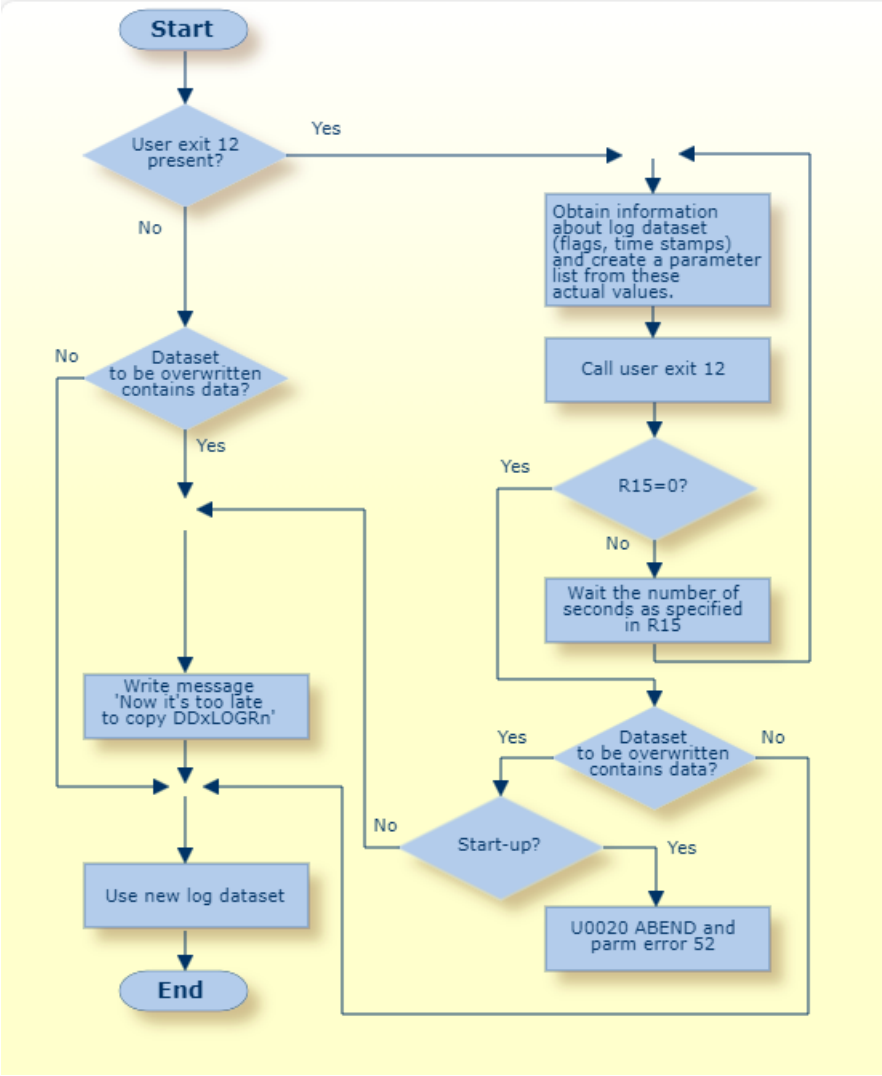
The call to the user exit is made using a standard BASSM R14,R15 Assembler instruction. All registers must be saved when control is received and restored immediately prior to returning control

to Adabas. Register 15 contains an action code as described in the user exit 2 section ***Output Parameter***, elsewhere in this guide.

For more information regarding ***Standard MVS Calling Conventions*** and ***Return from User Exits*** please refer the respective sections in the *Overview*.

Note: User exit 12 must return the same AMODE value to the calling program that was active when user exit 12 was called. The recommended Assembler instruction to return is BSM 0,R14.

User Exit 12 Calling Sequence

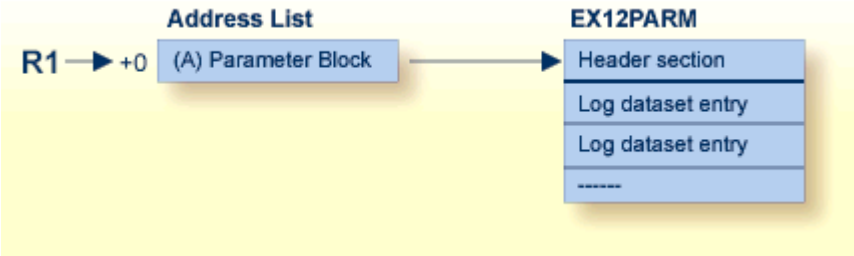


User Exit Interface

Optionally, the user exit may initialize its operation. It may store any value in field EX12USER of the EX12PARM parameter block to keep track of its resources. This field is considered as "owned" by the user exit and is supplied again for all subsequent executions of the exit. It is set to zero when the exit is first called and is not modified by Adabas thereafter.

The user exit is called again during termination to do any necessary finishing or 'cleanup' work.

Parameters



DSECT of the EX12PARM Parameter Block

```
         MACRO
         EX12PARM
.*********************************************************************
.*                                                                   *
.* Name          Ex12Parm                                            *
.*                                                                   *
.* Component     ADABAS User Exit                                    *
.*                                                                   *
.* Function      Parameter list for User Exit 12                     *
.*               (replacement for User Exit 2 for use when there are *
.*                two or more PLOGs, CLOGs or ALOGs)                 *
.*                                                                   *
.* Parameters    None                                                *
.*                                                                   *
.* Restrictions  None                                                *
.*                                                                   *
.* Notes         None                                                *
.*                                                                   *
.*********************************************************************
.*
EX12PARM DSECT ,                   User Exit 12 Parameter List
*
EX12HDR  DS    OF                  Common header section
*
EX12USER DS    F                   Reserved for the user.  This field  +
                                   is initialized to zero before the   +
                                   exit is called the first time, and  +
                                   will not be altered by ADABAS after +
                                   that.  It can be used to maintain   +
                                   information across invocations.
*
EX12LOGT DS    X                   Log type
EX12PLOG EQU     C'P'                PLOG
EX12CLOG EQU     C'C'                CLOG
EX12ALOG EQU     C'A'                ALOG
*
EX12TYPE DS    X                   Call type
```

```
EX12TBEG EQU     C'S'                Nucleus start
EX12TSW  EQU     C'W'                Log switch
EX12TEND EQU     C'T'                Nucleus termination
EX12TFRC EQU     C'F'                Forced switch
         DS    XL2                 Reserved
*
EX12NLOG DS    F                   Number of logs
EX12DBID DS    F                   Database ID
EX12NUCI DS    F                   Nucleus ID
*
EX12PLGN DS    F                   Current session PLOG number         +
                                     (zero for CLOGs)
EX12NCMP DS    F                   Log just completed                  +
                                     (zero during initialization and   +
                                      for forced CLOG switch)
EX12STAT DS    X                   Flags of next log in sequence
EX12WNUC EQU     X'80'               Being written by the nucleus
EX12FULL EQU     X'40'               Completed by the nucleus
EX12RES  EQU     X'20'               Being copied by ADARES
EX12CL5  EQU     X'08'               CLOGLAYOUT=5
EX12NMRG EQU     X'04'               Records include time stamp (PLOG)
*                                     CLOG not merged (cluster)
EX12PMRG EQU     X'02'               Log partially merged (cluster)
EX12UNUS EQU     X'00'               Unused and/or copied
         DS    XL3                 Reserved
         DS    4F                  Reserved
*
EX12HDRL EQU   *-EX12HDR           Length of header section
*
EX12ENT  DSECT ,
EX12LOG  DS    0F                  Start of individual log dataset     +
                                   entries.  This section is repeated  +
                                   for the number of logs specified in +
                                   field EX12NLOG
*
EX12LTIM DS    XL8                 Time stamp of write to log dataset
EX12LNUM DS    F                   Number of log dataset
EX12LFLG DS    X                   Flags (mapped as in EX12STAT)
         DS    XL3                 Reserved
         DS    4F                  Reserved
*
EX12LOGL EQU   *-EX12LOG           Length of a log dataset entry
*
*
         MEND
```

Output Parameter

Parameter	Usage
R15 = 0	Nucleus continues processing.
R15 > 0	R15 is treated as the number of seconds to wait before calling user exit 12 again with updated status for all log data sets. During this time, no commands that may create log entries are processed.

Using Sample User Exit 12

Caution: Sample user exit routines are not supported under any maintenance contract agreement.

Adabas is delivered with a sample user exit 12 routine. This routine is written in the Assembler language and performs the following functions:

- Issues a message identifying the reason and the type of log for which it was called.
- Issues a message with the status and timestamp of all log data sets that are not empty.
- If any log data set is full and the status of at least one log data set changed since the last time the exit was called, the exit reads 80-byte records from the input data set with DD-name COPYJOB and processes them either as a job to be submitted or a started task to be initiated.
  - If the first two characters in the first record are equal to "//", the exit processes the input as a job to be submitted (the model job). This is expected to be an ADARES PLCOPY, CLCOPY, or ALCOPY job.
  - If the first two characters in the first record are not equal to "//", the exit processes the input as an MVS operator command to be issued (the model command). This is expected to be a START command for an ADARES PLCOPY, CLCOPY or ALCOPY started task.
- Replaces all occurrences of the character "?" with either "P", C", or "A", depending on whether the exit is invoked to process a PLOG, CLOG, or ALOG event, respectively. This allows a single input model to accommodate an event for any of the log types.
- When the exit processes a job to be submitted, such as job control statements to be directed to a job execution queue:
  - If a record in the model job starts with the special stop mark "//*?", the exit:
    - replaces the stop mark with "//* ", in other words it changes the "?" to a blank.
    - stops copying further records to the output, when it is a PLOG or CLOG.
    - continues copying records and replacing "?" with "A", when the output is ALOG.

  - The exit writes the processed records to the output data set with DD-name INTRDR2 (normally, the JES internal reader).
- When processing an MVS operator command to be issued, the exit:
  - Picks up the contents of the first record as the default command to be issued, independent of event type, replacing "?" with "P", "C" or "A".
  - Reads any subsequent record and – if it starts with "P:", "C:" or "A:" and the event type is PLOG, CLOG, or ALOG, respectively – picks up the contents of the record starting at the third column (directly following the ":"), again replacing "?" with "P", "C" or "A".
  - Issues the last record picked up as an MVS command. This requires Adabas to run APF-authorized (the standard for Adabas installations).
- If at least one log data set is not full, it returns to the caller with R15 equal to zero. This allows Adabas to proceed.
- If all log data sets are full, it returns to the caller with R15 nonzero. This directs Adabas to wait for the number of seconds in R15, then call the exit again with an updated status of all log data sets. The default wait time is 30 seconds.

Activating Sample User Exit 12

Important: This sample user exit routine submits a job or issues an MVS operator command under the user ID and privileges given to the Adabas nucleus. It reads the job or operator command from the data set or member specified in the Adabas JCL under DD-name COPYJOB. This data set or member must be protected against unauthorized access, for example the APF-authorized load library used for Adabas. If a user can modify the COPYJOB data set or member to induce Adabas to perform an action that the user is otherwise unauthorized to perform, this can open the way to a privilege elevation attack.

The sample user exit 12 routine is delivered as a source (member UEX12) and as a load module (member UX12SAMP) that can be used without change or reassembly.

The source and load forms of the user exit are delivered in the Adabas source and load libraries, respectively. The job to assemble the user exit UX12ASML is located in the Adabas jobs library. The jobs library also contains a sample job UX12CJOB to be customized for submitting ADARES PLCOPY, CLCOPY, or ALCOPY jobs, as well as a sample MVS command UX12SRPC to be customized for initiating ADARES PLCOPY, CLCOPY, or ALCOPY started tasks.

To activate the sample user exit 12:

1 Specify the `UEX12=UX12SAMP` ADARUN parameter for the Adabas nucleus and one or more of the following ADARUN parameters:

   - `NPLOG={2-8}`

     To use this parameter, you must also set `PLOGDEV` and `PLOGSIZE`.

For more information about NPLOG, see *Operations > ADARUN Parameter Directory > NPLOG Parameter: Number of Protection Log Data Sets*.

- NCLOG={2-8}

  To use this parameter, you must also set CLOGDEV and CLOGSIZE.

  For more information about NCLOG, see *Operations > ADARUN Parameter Directory > NCLOG Parameter: Number of Command Log Data Sets*.

- NALOG={2-8}

  To use this parameter, you must also set ALOGDEV and ALOGSIZE.

  For more information about NALOG, see *Operations > ADARUN Parameter Directory > NALOG Parameter: Number of Audit Log Data Sets*.

2 Under the DDNAME COPYJOB, supply the model job that you want the user exit to adjust and submit, or the model command that you want the user exit to adjust and issue.

3 If you want the sample user exit to submit jobs, provide the following DD statement:

```
//INTRDR2  DD  SYSOUT=(*,INTRDR)
```

Note: Both sample user exits 2 and 12 use this DD statement for submitting PLCOPY, CLCOPY, or ALCOPY jobs.

Member UX12ZAP in the Adabas jobs library contains a sample IMASPZAP job for customizing certain aspects of the sample user exit 12 load module (UX12SAMP), such as the wait time if all log data sets are full, or the message ID for the messages it writes to the operator console.

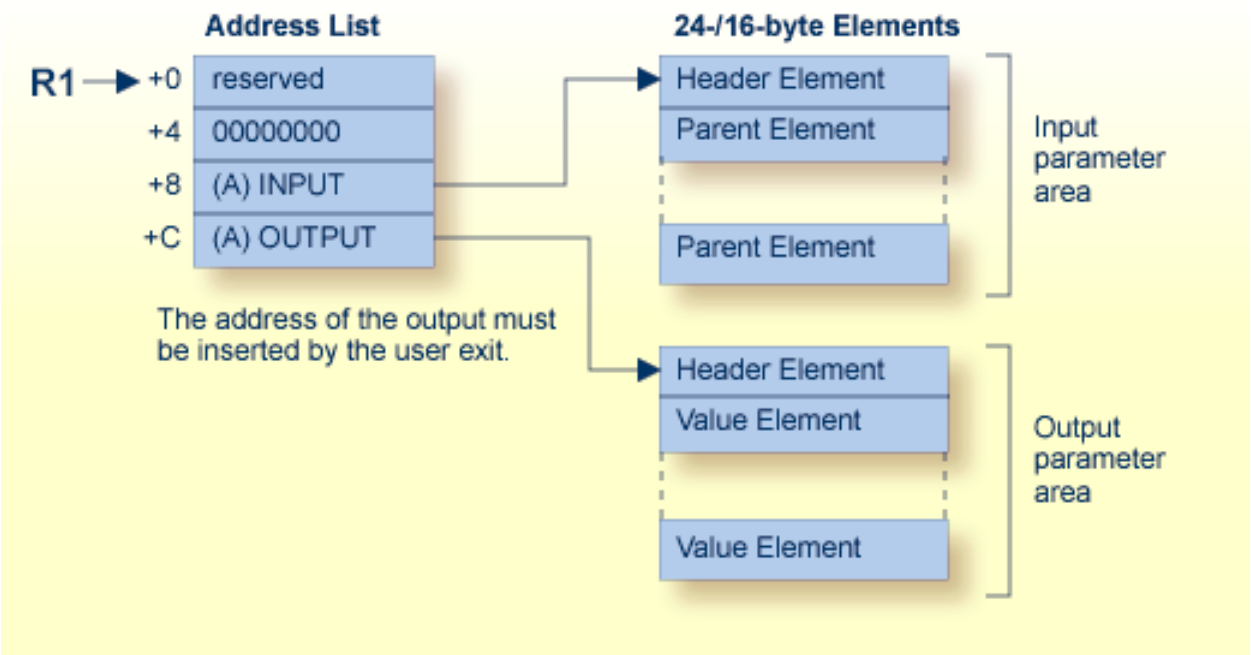
Messages by Sample User Exit 12

The sample user exit 12 routine writes messages to the operator console, identified by message IDs. Refer to the *UEX12* - Messages Written by the Sample User Exit 12* section in *Messages and Codes* for more information.

13 Hyperdescriptor Exits 01 - 31

- Main Parameter Area 61
- Input Parameter Area (Pointed to by Third Parameter Address) 62
- Output Parameter Area 64
- Null Value Option 65
- Hyperdescriptor Exit Initialization Call 66
- Hyperdescriptor Exit Sample 66

The hyperdescriptor exits 1 through 31 (HEX01...HEX31) are required to define the algorithm for user-supplied descriptor values (see the *Adabas Utilities Manual* documentation). A hyperdescriptor exit is called by the ADACMP utility, ADACHK utility or the Adabas nucleus whenever a hyperdescriptor value is to be generated. ADACMP and ADACHK always use the hyperdescriptor exit specified in its own `HEXnn` ADARUN statement. When the ADAINV utility specifies a hyperdescriptor exit, the exit used is the one specified in the Adabas nucleus' ADARUN statement.



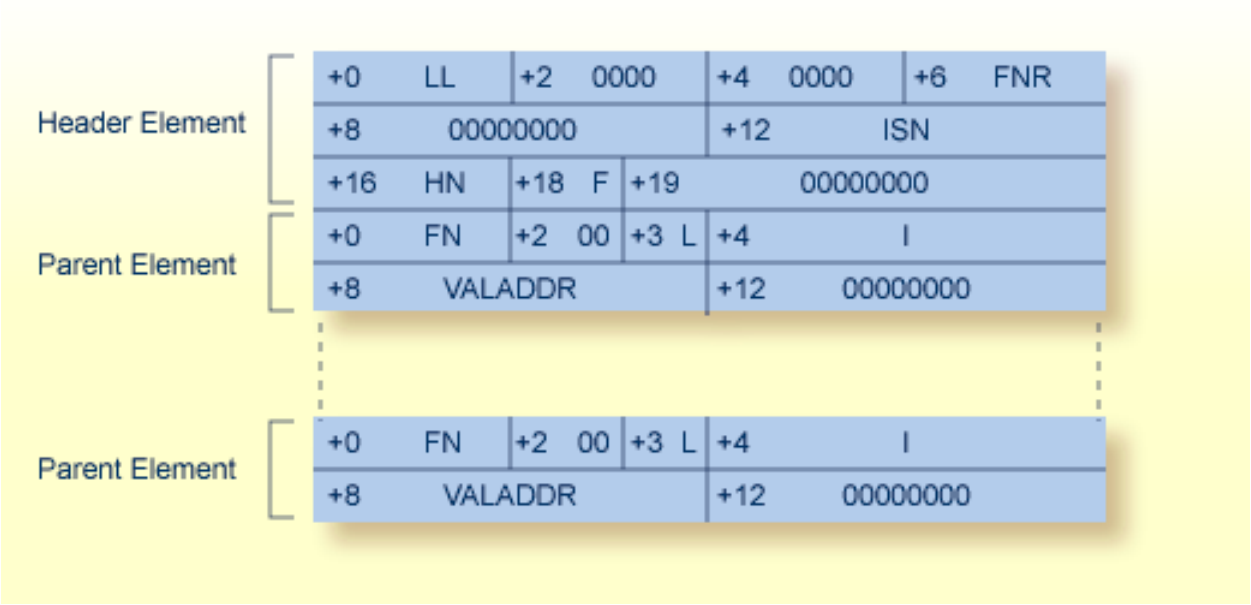
Notes:

1. Hyperdescriptor exits must return the same AMODE value to the calling program that was active when the hyperdescriptor exit was called.
2. An initialization call is made to each loaded hyperdescriptor exit during Adabas nucleus or ADACMP startup. For more details, read ***Hyperdescriptor Exit Initialization Call***, elsewhere in this chapter.

Main Parameter Area

Parameter	Content
0 (R1)	Reserved (must not be changed)
4 (R1)	Fullword of zeros (must not be changed)
8 (R1)	Address of the beginning of the input parameter area.
12 (R1)	Address of the beginning of the output parameter area. This address must be inserted by the user-written program. An output parameter area must always be returned by the user hyperdescriptor exit. If no values are to be returned, the address will point to a Header Element with a total length that indicates no Value Elements exist.

Input Parameter Area (Pointed to by Third Parameter Address)



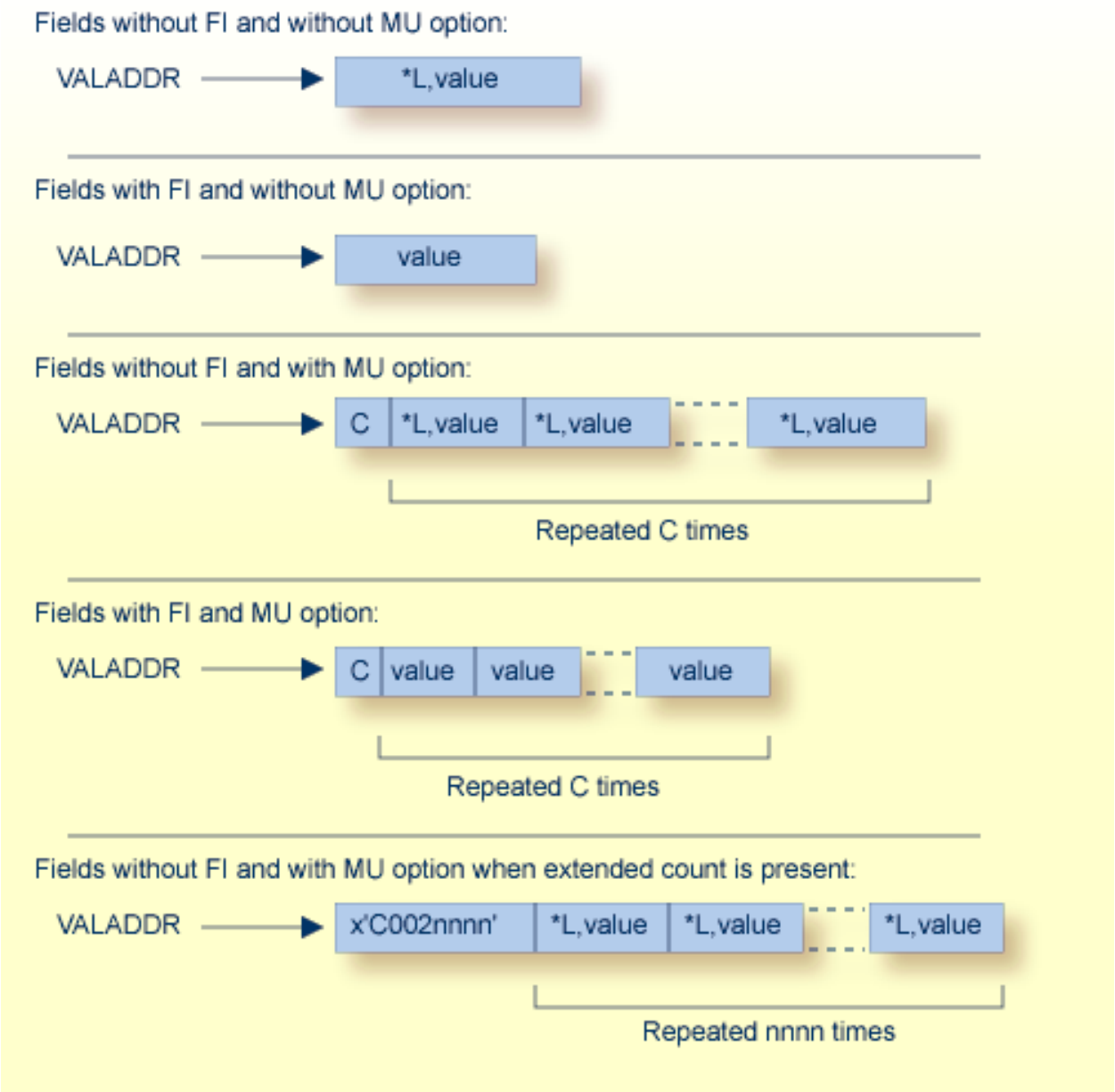
Header Element Fields

LL	Total length of the input parameter area, including this length field
FNR	File number
ISN	ISN assigned to the record
HN	Name of the hyperdescriptor
F	Flag byte: ■ X'02' indicates file with extended MU or PE fields ■ X'80' indicates initialization call

Parent Element Fields

FN	Name of the parent field
L	Length of the value pointed to by VALADDR if the parent field is defined with the FI option.
I	Four-byte periodic group index of the parent field. If the parent field is not part of a PE group, these bytes contain zeros.
VALADDR	Address of the value of the parent field. The format of the value depends on the options of the fields. If the parent field is defined with the NU (null value suppression) option and the value for this field is suppressed, no input parameter element is created.

The following examples show formats for the value pointed to by VALADDR for parent fields with combinations of the FI (fixed storage) and MU (multiple-value) options:



where:

C A one-byte value representing the MU count. If the MU value is for a file defined with extended MU or PE fields, an extended count may be present. For more details on the extended count, read *Identifying MU and PE Occurrences Greater Than 191 in Compressed Records* , found in the ADACMP documentation in *Adabas Utilities Manual*.

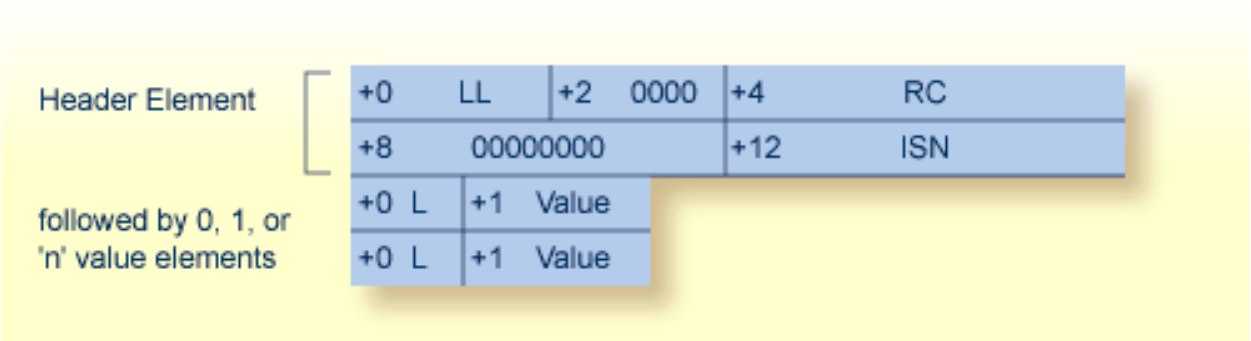
*L A hexadecimal value length, including this one- or two-byte length value. For lengths from 1 through 127, only a single byte is required. For lengths ranging 128 to 255, two bytes are needed: the first byte is set to X`80', and the second byte is set to the actual length value (see the following example table):

Length	Byte 1	Byte 2
L=127:	x`7F'	(<none>)
L=128:	x`80'	x`80'
L=255:	x`80'	x`FF'

Output Parameter Area

This area must be allocated and filled within the hyperdescriptor user exit. The address of this area must be placed into the fourth position of the **main parameter area** (at offset 12).

This area consists of a 16-byte header followed by the generated hyperdescriptor values in compressed format.



Header Element

LL	Total length of the output parameter area, including this length field. If no values are returned, the total length is set to the length of the Header Element.
00	Reserved space. This must be set to zeros.
RC	Return code. The hyperdescriptor exit may set a non-zero value here to indicate the call is rejected; a value of "16" is recommended. If this field is non-zero, the call will fail with response code 79 (ADARSP079).

ISN	The ISN to be assigned to the descriptor values. If the original ISN is to be changed, the new ISN must be inserted here. If these four bytes contain zero on return to the Adabas nucleus, the original ISN is used. This is a four-byte binary value.

Note: If the hyperdescriptor exit returns an ISN in the ISN field of the header element, the file must be defined with USERISN=YES to prevent ISN reassignment when the file is later reloaded.

Value Elements

L	Length of the following value, including this length byte. The maximum length depends on the format in use for the hyperdescriptor.
Value	The descriptor value to be inserted into the index. The value must follow the rules in effect for the format assigned to this hyperdescriptor. If the hyperdescriptor is defined with the PE option, one byte containing the one-byte PE index must immediately follow the value and be included in length L. If the hyperdescriptor defined with the PE option is for a file defined with extended MU or PE fields, two bytes containing the two-byte PE index must immediately follow the value and be included in length L. The nucleus checks values of packed or numeric format for validity. Valid signs for packed fields are A,C,E,F (positive) and B,D (negative). The nucleus changes all signs to F or D.

Examples:

L	Value	Notes
04	R E D	
06	B L U E02	where X'02' is a PE index
03	123F	packed 123
04	123F01	packed 123 in PE group with index 1
07	B L U E0002	where X'0002' is a PE index for a file defined with extended MU or PE fields
05	123F010A	packed 123 in extended PE group with index 266

Null Value Option

The NU (null value) option is possible for the hyperdescriptor or parent fields. The possible combinations are as follows:

- The hyperdescriptor is not NU:
  - The parent field is not NU and the value is null, the hyperdescriptor exit is called and the null value is passed.
  - The parent field is NU and the value is null, the hyperdescriptor exit is called and no input parameter element is created for this parent field.

- All parent fields are NU and all values are null, the hyperdescriptor exit is called and no input parameter element is created for any parent field.
- The hyperdescriptor is NU:
  - The parent field is not NU and the value is null, the hyperdescriptor exit is called and the null value is passed.
  - The parent field is NU and its value is null, the hyperdescriptor exit is called and no input parameter element is created for this parent field.
  - All parent fields are NU and all values are null, the hyperdescriptor exit is not called.

Hyperdescriptor Exit Initialization Call

During Adabas nucleus or ADACMP startup, each loaded hyperdescriptor exit is called with an initialization call. The main parameter area must be used as documented. The third parameter address will point to an input parameter area with a header length indicating that no values follow. The flag byte will be set to x'80' to indicate the initialization call. Upon return, the hyperdescriptor exit must set the fourth parameter address to an output parameter area with a header length indicating that no values are returned.

Hyperdescriptor Exit Sample

Sample hyperdescriptor exit source is supplied in the Adabas source library in member USERHX03. A sample job to assemble and link the hyperdescriptor exit is supplied in member ASMUHX03 of the sample job library in z/OS environments.

Caution: Sample user exits and programs and are not supported under any maintenance contract agreement.

14 Collation Descriptor Exits 01 - 08

- Collation Descriptor Exit Interface 68

The collation descriptor exits 1 through 8 (CDX01 through CDX08) are used for encoding and decoding values for the corresponding collation descriptors.

A collation descriptor may be defined for a field with alphanumeric or wide format. Its values are stored in the index, not in the record itself. The number of the collation descriptor exit used to derive the values is associated with the collation descriptor.

A sample collation descriptor exit CDXE2A is provided in the Adabas source data set. It converts EBCDIC to ASCII for the encoding function and the reverse (ASCII to EBCDIC) for the decoding function.

The Collation Exit implements three function entry points which are called on the following events:

INITIALIZE function

- nucleus session start
- utility initialization when collation exits have been defined (ADARUN parameters)

ENCODE function

- update/insert/delete of the parent's value (Nucleus)
- Search specifying the collation descriptor with the search value (Nucleus)
- compression of a record (ADACMP)

DECODE function

Read Index (L9) by Collation DE, only if the exit supports the DECODE function (Nucleus)

Collation Descriptor Exit Interface

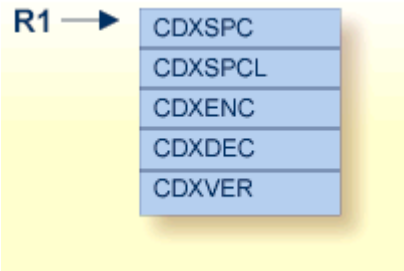
The collation descriptor exit interface is defined in the CDXPARM DSECT in the Adabas source data set. The interface has three functions:

- initialization
- encoding
- decoding (optional)

Initialization Parameters

R1 points to a list of addresses that point to five storage areas of the caller. The collation descriptor exit must set the five areas as follows:

CDXSPC	default space character; a maximum of 4 bytes
CDXSPCL	fullword containing the size of the space character
CDXENC	address of encoding function
CDXDEC	address of decoding function If the returned address is zero, decoding is not supported. The collation descriptor cannot then be used for L9 processing.
CDXVER	address of zero-byte delimited version string



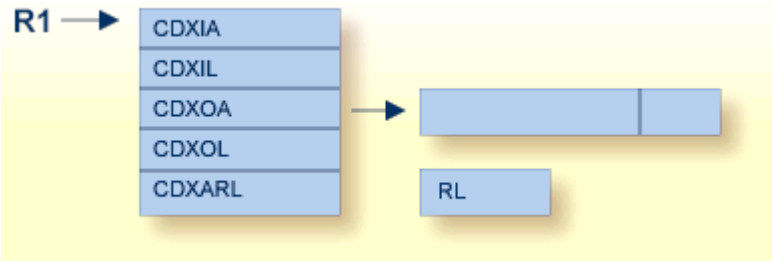
Encoding/Decoding Parameters

R1 points to a list of five fullword-sized parameters:

CDXIA	address of the input string
CDXIL	length of the input string
CDXOA	address of the output string
CDXOL	size of the output area
CDXARL	address of the length of the returned output string

The collation descriptor exit stores

- the output string in the area at the address specified by CDXOA; and
- the input string length in the fullword at the address specified by CDXARL.



15 SMF User Exit

The SMF user exit can be used to add a user-define detail section to the Adabas SMF record. The user exit is a separate load module whose name is provided by the ADARUN UEXSMF parameter.

The SMF user exit may not incur any TCB waits, for example, from I/O and WTORs. A wait will delay all nucleus activity.

The exit is invoked in AMODE 31 and Primary ASC mode using `BASR R14,R15` and must return in the same addressing and ASC mode with registers R2-R13 unaltered. The user exit should have these registers set on entry:

- R13 - Standard Format-1 Savearea
- R14 - Return address
- R15 - Entry address
- R1 - The address of a parameter list mapped by macro SMFEXPL.

Adabas expects the following registers to be set on the return from the user exit:

- R15 - The address of a detail secton of one or more instances to be included in the SMF record if R0 is not equal to zero.
- R0 - The number of instances of the detail section whose address is in R15. A value of zerio will inhibit adding a detail section.
- R1 - The length of each instance of the detail section whose address is in R15, if R0 is not equal to zero.

The parameter list is mapped by macro SMFEXPL. Here is the DSECT of the SMF user exit parameter list:

```
SMFEXPL   DSect ,
SMFEXPLA  DS    A           Address of 1-byte Action code
SMFEXPAI  Equ    c'I'         Initialize environment
SMFEXPAT  Equ    c'T'         Terminate environment
SMFEXPAG  Equ    c'G'         Generate detail section
SMFEXPLD  DS    A           Address of 4-byte detail section
                              mnemonic (c'USER')
SMFEXPLB  DS    A           Address of Build Area buffer
SMFEXPLS  DS    A           Address of 4-byte buffer length
SMFEXPLC  DS    A           Address of a read-only copy of
                              SMF record base header, SDS and
                              ID sections
SMFEXPLU  DS    A           Address of doubleword for use by the exit
                              Zero on first call, never
                              altered by Adabas
SMFEXPLL  Equ   *- SMFEXPL  Length of SMFEXPL
```

The action code in SMFEXPLA specifies why the exit was entered. Code I (Initialize Environment) occurs once during nucleus initialization and before any SMF records have been written. The exit may wish to allocate a workarea for its own use. Code T (Terminate Environment) occurs once during nucleus termination, after all SMF records have been written. The exit may wish to free any storage it acquired. Code G (Generate SMF Record) occurs once each time an SMF record is generated.

The user exit is given a doubleword in which it can pass information from one invocation to another. It is initialized to zeros and never altered by ADASMF. This is a good place to convey information between invocations such as the address of a workarea.

The user exit is given a read-only copy of the basic SMF record with header, self-defining and product ID sections. When the action code is G (Generate), the exit can determine from the subtype in the header section whether it is invoked for an initialization, interval, or termination record.

The user exit is given an area that it may use to build its detail section. The initial contents on each invocation are undefined. The length of the buffer is specified by SMFEXPLS. It is 128KB in the current release but is subject to change in future releases. Check the buffer length parameter before using it. If detail section instances are built in this area, return its address in R15.

The maximum size of a single user detail section instance is that which fits into a single SMF record. The maximum length of an SMF record is fixed by z/OS as 32,756. The space available for a single instance is 32,756 less the sizes of the header, self-defining, and product ID sections as specified in the ASMFREC macro DSECTs. In this release, the SMF record allows 32,434 bytes maximum for any detail section instance, but this is subject to change in future releases.

You may wish to pass information between the SMF user exit and other exits running in the nucleus or other programs running in the same system. The z/OS name/token services are a good way to exchange information such as the address of a common data area. Refer to *z/OS MVS Programming: Assembler Services Guide, IBM document number SA22-7605* for more information.

Index

A

Adabas Review
    hub event handler user exit 5, 27
ADAULD utility
    user exit 9 processing, 41

C

cipher code
    program to insert in ACBX, 46
collation descriptor exits
    description, 67
    encoding/decoding parameters, 69
    initialization parameters, 68
    interface, 68
command log
    dual data set user exit, 12
command log (CLOG)
    format, 20
    multiple data set user exit, 49
    user exit 4 processing, 19
command processing user exit, 9, 45

D

data compression
    user exit 6 processing before, 31
dual log processing, 11

E

exit 1, 9
exit 11
    input and output parameters, 47
    processing, 45
exit 12
    activating the sample, 55
    calling sequence, 52
    exit interface, 52
    output parameter, 55
    processing, 49
    sample user exit, 55
exit 2
    calling sequence, 13
    input parameters, 14
    output parameter, 16
    processing, 11
exit 3
    input parameters, 18
    processing, 17
exit 4, 19
exit 5
    input parameters, 28
    other registry values at entry, 29
    output parameters, 29
    processing, 27
exit 6
    ADACMP Header processing, 34
    input parameters, 35
    output parameters, 35
    processing, 31
    samples, 32
exit 8
    input parameters, 39
    processing, 37
exit 9
    processing, 41
    sample, 44

H

hyperdescriptor
    user exit, 59
hyperdescriptor exit, 59
    initialization call, 66
    input parameter area (pointed to by a third parameter address), 62
    main parameter area, 61
    null value option, 65
    output parameter area, 64
    sample, 66

L

logs
    dual data set user exit 2 processing, 11
    multiple data set user exit, 49

O

operator
    program to provide instructions to, 38
operator interface user exit, 37

P

password
    program to insert in ACBX, 46
phonetic processing
    processing with user exit 3, 17
protection log
    dual data set user exit, 12
protection log (PLOG)
    multiple data set user exit, 49

S

SMF user exit, 71

U

user
    log data processing, 20
user exit
    exit 12, 52
user exits
    collation descriptor exits, 67
    exit 1, 9
    exit 11, 45
    exit 12, 49
    exit 2, 11
    exit 3, 17
    exit 4, 19
    exit 5, 27
    exit 6, 31
    exit 8, 37
    exit 9, 41
    SMF user exit, 71