Miscellaneous

This chapter covers the following topics:

- User Exit DAZUEX01
- User-supplied Index Maintenance Exit Routines
- Application/Integration of Software Corrections
- Upgrading to New ADL Releases

User Exit DAZUEX01

Before ADL issues an Adabas call, it passes control to a user-written user exit. The purpose of this user exit is to allow any modifications to Adabas parameters, for example, assignment of Adabas command IDs, to conform to user site standards. Also, this user exit might be useful for monitoring programs as well as access control programs.

Note, however, that the user is responsible for the integrity of the Adabas call parameters. This is in particular true for the Adabas command ID. ADL uses specific Adabas command IDs which are determined according to an internal algorithm. Should the user intend to modify these command IDs, care should be taken that identical Adabas command IDs used by ADL are reflected by identical command IDs in the user exit.

The entry point name of the user exit must be `DAZUEX01`. On entry to the user exit the following register conventions apply in batch and online:

R1	points to the Adabas parameter list
R13	points to a 18-fullword save area
R14	return address
R15	entry point address of DAZUEX01

In addition, under CICS, R8 points to the `CSA` and `R12` points to the user `TCA`.

The contents of those registers not mentioned above are undefined. On return, all registers must be restored to their original values with the exception of `R15`. The save area pointed to by `R13` may be used to save the registers of ADL.

The user exit must be re-enterable. When running under z/OS/XA, the user exit must eventually be able to run in `AMODE 31`, depending on how your environment has been configured.

In order to become active, the user exit must be link-edited with the ADL batch region controller, `DAZIFP`, with the executable ADL Consistency Interface module for batch, `DAZNUCA`, and with the executable ADL Interfaces module for CICS, `DAZNUCC`. See the section *z/OS Installation* or *z/VSE Installation* for more details on how to link-edit these modules.

User-supplied Index Maintenance Exit Routines

With the EXTRTN keyword of the XDFLD statement in the DBD definition, the name of a user-supplied index maintenance exit routine can be specified. This exit routine allows the suppression of indexing for certain data base records.

On entry to the exit routine, the following register conventions apply:

Register	Usage
R2	points to the proposed index pointer segment
R3	points to the exit routine table entry described below
R4	points to the index source segment
R13	points to an 18-fullword save area
R14	return address
R15	entry point address of the exit routine

The contents of registers not mentioned above are undefined. On return, all registers must be restored to their original values with the exception of R15, which must contain a return code of either 0 or 4. A return code of 0 means that the index is to be allocated, whereas a return code of 4 means that the indexing of the data base record is suppressed.

Control is passed to the exit routine for insert and replace calls which affect the corresponding secondary index. Unlike DL/I, the exit routine is not called by ADL for delete requests, since the descriptor value corresponding to this index is deleted by Adabas anyway. Therefore, logical error situations cannot occur.

An exit routine table entry has the following layout (see the source library member MEXRENDS):

```
EXRENDS  DSECT
EXRENTGN DS    CL8                NAME OF TARGET SEGMENT
EXRENXDN DS    CL8                NAME OF XDFLD
EXRENNAM DS    CL8                NAME OF USER EXIT ROUTINE
EXRENADR DS    A                  ENTRY POINT OF USER EXIT ROUTINE
EXRENDBD DS    A                  ADDRESS OF DBD ICB
EXRENSEX DS    H                  O.T. SEC. INDEX ICB (WRT. DBDIC)
EXREN$FL DS    X                  FLAG:
EXREN#AD EQU   B'00000001'          'USER EXIT ALREADY DEFINED'
EXREN#NF EQU   B'00000010'          'USER EXIT NOT FOUND'
         DS    0F                 ALIGN
EXRENLLL EQU   *-EXRENDS
```

In order to become active, the exit routine must be an executable module. The module name must be the same as specified with the EXTRTN keyword. It must be present on the corresponding load library. Under CICS, an entry in the DFHCSDUP table must be defined for each exit routine.

A sample exit routine is supplied in the source library member ADLIMEX.

Application/Integration of Software Corrections

Unless otherwise stated, all corrections distributed for ADL will be valid for both z/OS and z/VSE operating systems.

Source changes are to be applied directly to the affected members in the ADL source library. Under z/VSE, source changes to macros must be followed by an EDECK assembly of the affected macros.

Fixes to ADL load modules (ZAPs) are distributed in the IMASPZAP format for the z/OS operating system, and in MSHP format for the z/VSE operating system.

When applying corrections to load modules, it is recommended to follow the guidelines given in the *Adabas Installation* documentation.

It is strongly recommended to apply all ZAPs to the affected object modules directly and to repeat the link-edit of the affected ADL executable load modules afterwards. This is to ensure that the corrections remain active, even if an executable module should be re-linked.

Upgrading to New ADL Releases

In general, an upgrade to a new ADL release will require the following steps to be performed:

1. Load the ADL libraries to disk. (Step 1 of the original installation procedure.)
2. Update the error messages on the ADL directory file. (Step 2 of the original installation procedure.)
3. Reload the ADL Online Services. (Step 3 of the original installation procedure.)
4. Regenerate the ADL parameter module. (Step 4 of the original installation procedure.)
5. Create the new ADL executable load modules. (Steps 5 to 9 of the original installation procedure.)
6. Create the new Consistency front-ends. (Steps 10 to 11 of the original installation procedure.)
7. Regenerate the CICS Runtime Control Tables.

Steps 1 through 6 are described in detail in the sections *z/OS Installation* and *z/VSE Installation*. Step 7 is described in the *ADL Interfaces* documentation.

Steps 1 through 5 can be performed with Software AG's System Maintenance Aid (SMA).

When upgrading to a new ADL release, you will usually not have to reconvert your DBD definition or to repeat the database conversion procedure. The upgrade to ADL 2.3 requires a data migration as described in the section *Migration to ADL 2.3 and Backward Migration*.