Migration to ADL 2.3 and Backward Migration

This section covers the following topics:

- Migrating to ADL 2.3
- Backward Migration
- JCL Requirements
- Other Changes

Migrating to ADL 2.3

The Adabas Bridge for DL/I uses several internal fields to reflect the DL/I hierarchy. These fields are the Z-field (`Z0, Z1, Z2` etc.) and the secondary index fields (their Adabas names start usually with `'X'`). The Z-fields are stored with each dependent record, the secondary index fields with each secondary index source record.

Some of the internal fields contain Adabas ISNs; with ADL 2.2 the `Z0` and `Z1` fields additionally contained a DBID and file number. The layout of ADL internal fields has been changed with ADL 2.3 to be able to handle 4-byte ISNs and 2-byte DBIDs and file numbers. The new layout (see the section *Performance Considerations* in the *ADL Interfaces* documentation for more information) is not compatible with the ADL 2.2 layout, i.e. ADL 2.3 does not support the ADL 2.2 layout and vice versa. Therefore all data which have been converted with ADL 2.2 or below must be migrated to the new ADL 2.3 layout before you can work with ADL 2.3.

Although ADL 2.3 is able to run with ADL 2.2 directory entries, physical DBD definitions must be reconverted to generate the new ADACMP and DAZUEX06 cards. It is not required to re-convert PSB definitions or logical DBD definitions.

For the migration, all ADL files (i.e. files converted from DL/I) must be unloaded and reloaded with Adabas utilities. At the reload the Adabas User Exit 6 DAZUEXMI maps the old layout to the new layout.

The following steps must be performed for the migration of the data to the new layout:

- Step 1: Preparation
- Step 2: Install ADL 2.3
- Step 3: Run the DBD definition through the ADL CBC utility
- Step 4: Create the Adabas User Exit 6 for migration
- Step 5: Modify the ADACMP cards
- Step 6: Unload and reload the data with Adabas utilities

After these steps have been performed for all ADL files, you can run your applications against ADL 2.3.

For emergency, ADL offers a backward migration from ADL 2.3 to ADL 2.2. See section *Backward Migration* for details.

The following sections describe the migration steps in more details.

Step 1: Preparation

Copy the ADL directory file with standard Adabas utilities. The new file will be used as ADL 2.3 directory.

Note:
Although this step is not required, it allows to convert the DBD definitions to the new directory (step 3) while ADL 2.2 is still running.

Save the ADL 2.2 User Exit 6 extension (if available) and the old ADACMP cards. The old User Exit 6 extension simplifies the data migration (step 6) because you do not need to specify the SEQ parameter. The old ADACMP cards will be checked later against the new ADACMP card to identify manual changes (step 5)

Step 2: Install ADL 2.3

The installation of ADL 2.3 is described in *Upgrading to New ADL Releases* in the section *Miscellaneous*.

Load the new error messages into the new ADL 2.3 directory file.

If you have used logical file numbers with ADL 2.2, you can delete the DAZLDT entries from the ADL 2.3 parameter module.

Step 3: Run the DBD definition through the ADL CBC utility

The CBC utility is described in details in the *ADL Conversion* documentation, section *ADL Conversion Utilities for DBDs and PSBs*. Consider the following:

- There is no need to re-assemble the DBD definition (step 1 of the CBC utility).
- The `DBID` and `FNR` parameters of the GENDBD/GENSEG functions reflect the physical file, i.e. the Adabas file where the data is stored (with ADL 2.2 it reflected the logical file).
- If the DBID of the data file is the same as the DBID of the ADL directory, omit the `DBID` parameter for the GENDBD function. This eases the creation of mirror databases.
- Specify as `LOGID` the previous value of `FNR` for every GENDBD/GENSEG function where the `FNR` parameter was specified.
- Specify all other parameters as with ADL 2.2. Especially if you have used the `BACKW` parameter, it must use the same value as before.
- The member with the ADACMP cards is named W*fffff*, where *fffff* is the file number. Use several libraries to save ADACMP cards of several DBIDs.

- The member with the User Exit 6 extension is named I*fffff*, where *fffff* is the file number.

Step 4: Create the Adabas User Exit 6 for migration

Assemble the Adabas User Exit 6 extension and link-edit this with `DAZUEXMI`. The Adabas User Exit 6 extension is the output of step 3 of the control block conversion.

If available, use the extension generated by ADL 2.2, otherwise the extension generated by ADL 2.3.

Step 5: Modify the ADACMP cards

If you had adjusted the Adabas FDT (additional user fields, super descriptors, etc.) with previous ADL versions, you must apply the changes to the ADACMP cards accordingly.

Step 6: Unload and reload the data with Adabas utilities

Unload each Adabas file used to store the converted data using the standard Adabas utility ADACMP DECOMPRESS. Specify the option ISN.

The data of each file is loaded individually using the standard Adabas utilities ADACMP COMPRESS and ADALOD.

The sequential file produced by the ADACMP DECOMPRESS is taken as input for the ADACMP COMPRESS step, as are the ADACMP statements generated by the CBC utility.

Each Adabas file used must be loaded with the option USERISN. This applies to both the ADACMP and the ADALOD steps (for ADACMP it is already generated by the CBC utility).

Adabas User Exit 6

The ADACMP COMPRESS step uses Adabas User Exit 6. This exit consists of two parts which were linked together in step 4:

1. **Fixed Part**
   The fixed part consists of the `DAZUEXMI` module, which maps the ADL 2.2 layout to the ADL 2.3 layout.

2. **User Exit 6 Extension**
   The User Exit 6 extension is generated by the `CBC` utility and contains information on the structure of the DBD being migrated, and the default record layouts of the Adabas file(s) used to store the data. You can use the User Exit 6 Extension generated by ADL 2.2 as well as the extension generated by ADL 2.3.

Adabas User Exit 6 needs a control statement to indicate which Adabas file should be migrated. The syntax of this control statement is as follows:

```
FNR=nnnnn,MODE=MIGR,SEQ=sss,DBID=nnn
```

Parameter	Description	Possible values	Default
FNR	Specifies the file number to be processed. The value must be the same as the `FNR` specified at the CBC utility run, i.e. the logical file number if the extension was generated with ADL 2.2 or the physical file number if it was generated with ADL 2.3.	1 - 65534	
MODE	Specifies whether the data should be migrated from ADL 2.2 to ADL 2.3 or vice versa.	MIGR -The data is migrated from ADL 2.2 to ADL 2.3 BACK - The data is migrated back from ADL 2.3 to ADL 2.2	MIGR
SEQ	Specifies the ADL 2.2 processing sequence of the file. This parameter must only be specified if the extension was generated with ADL 2.3.	SEG - Sequence is "segment". This was the default for the ADL 2.2 conversion. PAR - Sequence is "parent". This is the setting for all DBDs converted with ADL 2.1 or before.	SEG
DBID	Specifies the logical DBID to be inserted to the ADL internal field. The value must be the same as the DBID specified at the ADL 2.2 `CBC` utility run. This parameter is mandatory for the backward migration (`MODE=BACK`), otherwise it is not required.	1 - 255	

Backward Migration

For emergency, ADL offers a backward migration from ADL 2.3 to ADL 2.2. The following steps must be performed for a backward migration:

- Install ADL 2.2
- Restore the ADL directory
- Unload and reload the data with Adabas utilities

Install ADL 2.2

The ADL 2.2 installation is described in the ADL 2.2 documentation. If the old ADL load library is still available, you can omit this step.

Restore the ADL directory

Use the original ADL 2.2 directory (see step 1 of the migration).

Alternatively you can convert the DBD definition with ADL 2.2. The `DBID` and `FNR` parameters of the ADL 2.2 GENDBD/GENSEG functions reflect the logical file which is described in details in the ADL 2.2 documentation.

If the logical file numbers differ from the physical file numbers, DAZLDT entries must be added to the ADL parameter module.

Unload and reload the data with Adabas utilities

Unload each Adabas file used to store the converted data using the standard Adabas utility `ADACMP DECOMPRESS.` Specify the option `ISN`.

The data of each file is loaded individually using the standard Adabas utilities `ADACMP COMPRESS` and `ADALOD`. The sequential file produced by the `ADACMP DECOMPRESS` is taken as input for the `ADACMP COMPRESS` step. Use the `ADACMP` statements generated by the ADL 2.2 `CBC` utility run. Each Adabas file used must be loaded with the option USERISN. This applies to both the `ADACMP` and the `ADALOD` steps.

The `ADACMP COMPRESS` step uses Adabas User Exit 6. For the backward migration use the same Adabas User Exit 6 which was used for the (forward-) migration of the file to ADL 2.3.

The parameters of the Adabas User Exit 6 are described in details in the section above. For the backward migration you must specify `MODE=BACK`. Additionally to the parameters used for the migration, you must specify the `DBID` parameter, reflecting the logical DBID used at the ADL 2.2 CBC utility run.

JCL Requirements

The JCL requirements for the migration and backward migration are the same as for the initial load. Refer to *z/OS JCL Requirements* or *z/VSE JCS Requirements* in the section *ADL Data Conversion Utilities* of the *ADL Conversion* documentation.

Other Changes

z/OS CICS application programs which have been linked with the ADL language interface DAZLICI2, must be re-linked with the new language interface DAZLICI3. This is especially true for all precompiled EXEC DLI programs. See the *ADL Interfaces documentation*, section *CICS Installation and Operation* for further information.