Connecting UES-Enabled Databases

Overview

Prior to Adabas Version 7, Entire Net-Work converted all data for mainframe Adabas when necessary from ASCII to EBCDIC. Starting with Version 7, Adabas is delivered with its own data conversion capability called universal encoding support (UES). Entire Net-Work detects when it is connected to a target database that converts data and passes the data through to Adabas without converting it.

For Adabas Version 7.4, the link routines ADALNK and ADALCO are delivered with UES enabled. That is, LNKUES and the translation tables are linked in.

Note:
The use of UES-enabled link routines is transparent to applications, including applications that do not require UES translation support: it is not necessary to disable UES support.

Load Modules

The load modules for ADALNK and ADALCO have been linked with LNKUES and the default translation tables.

LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

The two standard translation tables are

- ASC2EBC: ASCII to EBCDIC translation; and

- EBC2ASC: EBCDIC to ASCII translation.

The Adabas translation table pair is provided in the section *Translation Tables*.

Default or Customized Translation Tables

You may use the load modules with the default translation tables linked in, or you may prepare your own customized translation tables, re-assemble the tables, and link them with the LNKUES module that is delivered.

Notes:

1. It should only be necessary to modify these translation tables in the rare case that some country-specific character other than "A-Z a-z 0-9" must be used in the Additions 1 (user ID) or Additions 3 field of the control block.
2. The load module LNKUESL delivered with earlier levels of Adabas version 7 is no longer supplied since the link jobs now specify the LNKUES module and the translation tables separately.
3. The LNKUES module is functionally reentrant; however, it is not linked that way in the Adabas load library.
4. When linking the LNKUES load module and the translation tables, the linkage editor may produce warning messages concerning the reentrant or reusability status of the linked module. These warning messages can be ignored.

Source Modules

The ADALNK and ADALCO source modules have been coded to enable UES support by default when assembled:

- The &UES Boolean assembly variable is set to 1 by default.; the statement to set it to 0 has been commented out.
- The setting of the other Boolean variables and equates such as the SVC number and the database ID remain unchanged from earlier deliveries of the source modules.

Job Steps

Job library members ALNKLCO, ALNKLNK, and ALNKLNKR are set up to assemble and link the ADALCO, ADALNK, and ADALNKR modules, respectively, with the UES components in three steps:

- Assemble the link module into the Adabas load library.
- Assemble the two translation tables into the Adabas load library.
- Link the link module with LNKUES and the translation tables and put the resulting load module into a user load library.

Calling LNKUES

LNKUES is called only on ADALNK request (X’1C’) and reply (X’20’) calls if the first byte of the communication ID contains X’01’ and the second byte does not have the EBCDIC (X’04’) bit set.

- For requests, LNKUES receives control before UEXITB.
- For replies, LNKUES receives control after UEXITA.

Required Environment

The Adabas database must be UES-enabled. See *DBA Tasks* and the ADACMP and ADADEF utility sections in the *Adabas Utilities* documentation for more information.

Connection Possibilities

UES-enabled databases can be connected to machines with different architectures through Com-plete or Smarts, or through Entire Net-Work.

Connection Through Com-plete or Smarts

Adabas SQL Gateway (ACE) clients may not be strictly EBCDIC in an environment where databases are connected through Software AG’s internal product Smarts (APS).

The relevant Adabas link routine ADALCO is UES-enabled by default. The sample jobstream to assemble the ADALCO module is ASMLCO.X.. Sample JCL to link the ADALCO.PHASE may be found in job LINKS.X.

The assembled and linked ADALCO (as delivered or with customized and reassembled translation tables) is placed in the Smarts sublib or a user sublib and made available in the LIBDEF chain for the job.

- Step 1: Assemble the ADALCO Module into the Adabas Sublibrary (SMA Job Number I070)
- Step 2: Assemble the Two Translation Tables into the Adabas Load Library (SMA Job Number I056)
- Step 3: Link the ADALCO.PHASE with the Translation Tables and LNKUES (SMA Job Number I088)
- Step 4: Make ADALCO Available to Smarts

Step 1: Assemble the ADALCO Module into the Adabas Sublibrary (SMA Job Number I070)

Modify the member ASMLCO.X to assemble ADALCO as described in the example below:

```
X $$ JOB JNM=ASMLCO,CLASS=0,DISP=D 000100
X $$ LST CLASS=A,DISP=D 000200
// JOB ASMLCO ASSEMBLE THE COM-PLETE/SMARTS LINK
* *******************************************************
* ASSEMBLE ADALCO AS LCOVSE.OBJ
* ********************************************************
*
* Customize this job as follows:
*
* Change the ”X $$” to ”* $$” on the power cards.
* Change ”$*” to”/*”.
* Change ”$&” to ”/&”.
* Change ”X EOJ” to ”* EOJ”.
* Customize the CLASS, DEST and other information as needed.
* Change vvvvvv to the volume for the punch work file.
* Change ssss to the extent location for the punch file.
* Change ttt to the size of the work file extent.
* Provide your PROC information for the Adabas PROC.
* Provide proper LIBDEF information for the search chain.
* Provide the proper sublib information on the LIBR parm.
*
* *********************************************************
// EXEC PROC=ADAV7LIB 000400
// OPTION DECK,NOLINK
// LIBDEF SOURCE,SEARCH=(ADALIB.SUBLIB,...),TEMP
// DLBL IJSYSPH,'PUNCH.WORK1',0
// EXTENT SYSPCH,vvvvvv,1,0,ssss,ttt
ASSGN SYSPCH,DISK,VOL=vvvvvv,SHR
// EXEC ASMA90,SIZE=(ASMA90,50K), X
PARM='EXIT(LIBEXIT(EDECKXIT(ORDER=AE)))'
COPY ADALCO
END
$*
CLOSE SYSPCH,PUNCH
* CATALOG LCOVSE
// DLBL IJSYSIN,'PUNCH.WORK1'
// EXTENT SYSIPT,vvvvvv
ASSGN SYSIPT,DISK,VOL=vvvvvv,SHR
// EXEC LIBR,PARM='AC S=ADALIB.SUBLIB;CATALOG LCOVSE.OBJ R=Y'
$*
CLOSE SYSIPT,READER
$&
// JOB RESET
* *********************************************************
* RESET SYSIPT AND SYSPCH
* *********************************************************
ASSGN SYSIPT,READER
ASSGN SYSPCH,PUNCH
$*
$&
X $$ EOJ
```

Step 2: Assemble the Two Translation Tables into the Adabas Load Library (SMA Job Number I056)

Assemble the ASCII to EBCDIC and EBCDIC to ASCII translation tables, either default or customized. The source modules are in the distribution source sublibrary as ASC2EBC.A and EBC2ASC.A.

Use the ASMLNK.X jobstream as a guide to prepare a jobstream for assembling and cataloging the ASC2EBC and EBC2ASC object modules.

Be sure to:

- Use either COPY ASC2EBC or COPY EBC2ASC as the input to the assembly step after the EXEC ASMA90 JCL statement.
- Specify the desired object module name on the LIBR parm, ASC2EBC for the ASCII to EBCDIC translation table or EBC2ASC for the EBCDIC to ASCII translation table.

Step 3: Link the ADALCO.PHASE with the Translation Tables and LNKUES (SMA Job Number I088)

Link the ADALCO, ASC2EBC, EBC2ASC and LNKUES object modules into a final ADALCO phase that is UES-enabled. Place this phase into a sublibrary which will be made available through a LIBDEF search chain at execution time. The LINKS.X job contains a step to do this. A segment of this JCL is:

```
PHASE ADALCO,*
MODE AMODE(31),RMODE(24)
INCLUDE LCOVSE
INCLUDE LNKUES
INCLUDE ASC2EBC
INCLUDE EBC2ASC
ENTRY ADABAS
// EXEC LNKEDT,PARM='MSHP'
```

Step 4: Make ADALCO Available to Smarts

The (re)linked ADALCO must be made available to Smarts. If you are calling Adabas version 7 and you do not have the correct LNKUES/ADALCO module, Adabas produces unexpected results: response code 022, 253, etc.

Additional Steps for Installing UES Support for Adabas

The following LIBR sublibraries are distributed with ADABAS for UES support:

```
SAGLIB.BTE421CS
SAGLIB.BTE421DS
SAGLIB.APS27102
SAGLIB.APS271
```

Note:
In Version 7.4.2, the link routines come linked with UES support by default.

to install these libraries:

1. Create a VSE library for the BTE and APS libraries.

   ```
   * $$ JOB JNM=CRUESL,CLASS=0,DISP=D,LDEST=(,XXXXXX)
   * $$ LST CLASS=A,DISP=D
   // JOB LIBRDEF
   // ASSGN SYS005,DISK,VOL=vvvvvv,SHR
   // DLBL DDECSOJ,'SAG.ECS.ADA74.LIB',2099/365,SD
   // EXTENT SYS005,vvvvvv,1,0,ssss,tttt
   // EXEC LIBR
   DEFINE L=DDECSOJ R=Y
   /*
   /&
   * $$ EOJ
   ```

2. Restore the BTE and APS libraries to this file. Refer to the *Report of Tape Creation* for the file positions on the distribution tape.

```
* $$ JOB JNM=RESTECS,DISP=D,CLASS=0,LDEST=(,xxxxxx)
* $$ LST DISP=D,CLASS=A
// JOB RESTECS
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,'SAG.ECS.ADA74.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// ASSGN SYS006,cuu
// MTC REW,SYS006
// EXEC LIBR,PARM='MSHP'
RESTORE SUBLIB=SAGLIB.BTE421CS:DDECSOJ.BTE421CS -
TAPE=SYS006 -
LIST=YES -
REPLACE=YES
/*
// MTC REW,SYS006
// ASSGN SYS006,UA
/&
* $$ EOJ
```

3. Repeat for BTE421DS and APS271 APS27102.

4. Modify the Adabas startup JCL, adding the UES env section after ADARUN parms:

```
ADARUN .....
ADARUN ....
/*
ENVIRONMENT_VARIABLES=/DDECSOJ/APS271/ENVVARS.P
/*
/&
* $$ EOJ
```

5. Reference the library where the libraries were restored in your Adabas proc:

```
// ASSGN SYS005,DISK,VOL=vvvvvv,SHR
// DLBL DDECSOJ,'SAG.ECS.ADA74.LIB',2099/365,SD
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
and add the libraries to the libdef chain:
(be sure APS27102 is referenced before APS271)
// EXTENT SYS005,vvvvvv,1,0,ssss,tttt
// LIBDEF PHASE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADA742... X
SAGLIB.AOS742,SAGLIB.ADE742,SAGLIB.ACF742,...X
DDECSOJ.BTE421CS,DDECSOJ.BTE421DS, X
DDECSOJ.APS27102,DDECSOJ.APS271)
// LIBDEF OBJ,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADA742... X
SAGLIB.AOS742,SAGLIB.ADE742,SAGLIB.ACF742, X
DDECSOJ.BTE421CS,DDECSOJ.BTE421DS, X
DDECSOJ.APS27102,DDECSOJ.APS271)
// LIBDEF SOURCE,SEARCH=(SAGLIB.USRLIB,SAGLIB.ADA742,... X
SAGLIB.AOS742,SAGLIB.ADE742,SAGLIB.ACF742, X
DDECSOJ.BTE421CS,DDECSOJ.BTE421DS, X
DDECSOJ.APS27102,DDECSOJ.APS271)
// LIBDEF PHASE,CATALOG=SAGLIB.USRLIB
```

6. Modify the ENVVARS.P file, adding the following line in the APS271 library:

```
* This member contains Environment Variables used by SMARTS and
* SMARTS based applications.
*
ECSOBJDIR=FILE://DDECSOJ/BTE421CS
```

7. Run the ADADEF utility set UES=YES:

   ```
   * $$ JOB JNM=ADADEF,CLASS=0,DISP=D,LDEST=(,xxxxxx)
   * $$ LST CLASS=A,DISP=D
   // JOB ADADEF EXECUTE THE ADABAS VERSION 7 UTILITY ***DEF***
   // OPTION LOG,PARTDUMP
   *
   // EXEC PROC=ADALIB
   // EXEC PROC=ADAFIL
   *
   // EXEC ADARUN,SIZE=ADARUN
   ADARUN PROG=ADADEF,MODE=SINGLE,SVC=svc,DEVICE=dddd,DBID=nnnn
   /*
   ADADEF MODIFY UES=YES
   /*
   /&
   * $$ EOJ
   ```

8. Start the database.

   You should see the following message:

   ```
   ENTIRE CONVERSION SERVICES INITIALIZED
   ```

Connection Through Entire Net-Work

UES-enabled databases are connected through Software AG’s Entire Net-Work (WCP) using the Adabas nonreentrant batch or TSO link routine ADALNK. The sample jobstream to assemble and link the nonreentrant ADALNK module is ASMLNK.X.

The assembled and linked nonreentrant, batch ADALNK (as delivered or with customized and reassembled translation tables) should be placed in the LIBDEF search chain with Entire Network.

- Step 1: Assemble the ADALNK Module into the Adabas Load Library (SMA Job Number I055)
- Step 2: Assemble the Two Translation Tables into the Adabas Load Library (SMA Job Number I056)
- Step 3: Link the Translation Tables and LNKUES into ADALNK (SMA Job Number I088)
- Step 4: Make ADALNK Available to Entire Net-Work

Step 1: Assemble the ADALNK Module into the Adabas Load Library (SMA Job Number I055)

Modify the member ASMLNK.X to assemble and link ADALNK, as follows:

```
* Change the ”* $$” to ”* $$” on the power cards.
* Change ”$*” to”/*”.
* Change ”$&” to ”/&”.
* Change ”X EOJ” to ”* EOJ”.
* Customize the CLASS, DEST and other information as needed.
* Change vvvvvv to the volume for the punch work file.
* Change ssss to the extent location for the punch file.
* Change ttt to the size of the work file extent.
* Provide your PROC information for the Adabas PROC.
* Provide proper LIBDEF information for the search chain.
* Provide the proper sublib information on the LIBR parm.
```

```
* $$ JOB JNM=ASMLNK,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ASMLNK ASSEMBLE ADALNK CATALOG IT AS LNKVSE.OBJ
* *******************************************************
* ASSEMBLE ADALNK CATALOG IT AS LNKVSE.OBJ
* ********************************************************
* ********************************************************
// EXEC PROC=ADALIB
// OPTION DECK,NOLINK
// LIBDEF SOURCE,SEARCH=(ADALIB.SUBLIB,...),TEMP
// DLBL IJSYSPH,'PUNCH.WORK1',0
// EXTENT SYSPCH,vvvvvv,1,0,ssss,ttt
ASSGN SYSPCH,DISK,VOL=vvvvvv,SHR
// EXEC ASMA90,SIZE=(ASMA90,50K), X
PARM='EXIT(LIBEXIT(EDECKXIT(ORDER=AE)))'
COPY ADALNK
/*
CLOSE SYSPCH,PUNCH
* CATALOG LNKVSE.OBJ
// DLBL IJSYSIN,'PUNCH.WORK1'
// EXTENT SYSIPT,vvvvvv
ASSGN SYSIPT,DISK,VOL=vvvvvv,SHR
// EXEC LIBR,PARM='AC S=ADALIB.SUBLIB;CATALOG LNKVSE.OBJ R=Y'
/*
CLOSE SYSIPT,READER
/&
// JOB RESET
* ********************************************************
* RESET SYSIPT AND SYSPCH
* ********************************************************
ASSGN SYSIPT,READER
ASSGN SYSPCH,PUNCH
/*
/&
$$ EOJ
```

Step 2: Assemble the Two Translation Tables into the Adabas Load Library (SMA Job Number I056)

Sample VSE jobs to assemble and catalog the ASC2EBC and EBC2ASC translate table modules may be found in members ASMA2E.X and ASME2A.X in the distribution sublibrary. The jobs should be customized according to the same instructions as the ASMLNK.X job cited above.

If you prefer to use the same translation tables that are used in Entire Net-work, change the COPY statements in ASC2EBC and EBC2ASC from UES2ASC and UES2EBC to NW2ASC and NW2EBC, respectively. After modifying the translation tables, be sure to (re)assemble them and link them with the delivered LNKUES module.

The Entire Net-Work translation table pair is also provided in the section *Translation Tables*.

Step 3: Link the Translation Tables and LNKUES into ADALNK (SMA Job Number I088)

Link the ADALNK, ASC2EBC, EBC2ASC, LNKUES, and other user exit modules into a final ADALNK module that is UES-enabled. Place this phase into a sublibrary which will be made available through a LIBDEF search chain at execution.

```
* $$ JOB JNM=ADALNKS,CLASS=A,DISP=D
* $$ LST CLASS=A,DISP=D
// JOB ADALNKS
// EXEC PROC=ADALIB <===== ADABAS PROCEDURE NAME
// OPTION CATAL
PHASE ADALNK,*,NOAUTO
MODE AMODE(31),RMODE(24)
INCLUDE LNKVSE
INCLUDE LNKUES
INCLUDE ASC2EBC
INCLUDE EBC2ASC
ENTRY ADABAS
// EXEC LNKEDT,PARM='MSHP'
/*
/&
* $$ EOJ
```

Step 4: Make ADALNK Available to Entire Net-Work

Make sure that the ADALNK phase is placed in a sublibrary available in the Entire Net-work job’s LIBDEF search chain.

The (re)linked ADALNK must be made available to Entire Net-Work. If you are calling Adabas version 7 and you do not have the correct LNKUES/ADALNK module, Adabas produces unexpected results: response code 022, 253, etc.