Adabas Native SQL

Installation Manual for z/VSE

Version 2.4.1

September 2009

Adabas Native SQL

This document applies to Adabas Native SQL Version 2.4.1 and to all subsequent releases.

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

Table of Contents

1 Installation Manual for z/VSE

This document describes how to install Adabas Native SQL on z/VSE.

This information is structured into the following sections:

General Information	This section provides general information about the installation procedure.
Tape Layout	This section describes the contents of the distribution medium.
Installation	This section describes how to perform the installation.
Installation Checklist	This section describes how to verify that the installation has been successful.
Hints for Using Adabas Native SQL	This section contains a list of useful hints about using Adabas Native SQL.

2 GENERAL INFORMATION

This chapter covers the following topics:

Installation Jobs

The installation of Software AG products is performed by installation jobs. These jobs are either adapted “manually” or generated by System Maintenance Aid (SMA).

For each step of the installation procedure described below, the job number of a job performing the respective task is indicated. This job number refers to an installation job generated by SMA.

Using System Maintenance Aid

If you are using Software AG's System Maintenance Aid (SMA) for the installation process, refer to the System Maintenance Aid manual.

COBOL Compiler

Adabas Native SQL Version 2.4.1 has been built using a COBOL 2 compiler (IGYCRCTL), whereas previous versions used COBOL 1 (FCOBOL). This may be important when linking changed source modules.

Because the Adabas Native SQL precompilers are written in COBOL 2, it is necessary to add the COBOL 2 runtime library to the

```
// LIBDEF *,SEARCH...
```

card for the precompiler step.

3 TAPE LAYOUT

The tape contains the following files. The notation *nnn* represents the version, release and SM level.

File	Format	Contents	Space Requirements
SQL*nnn*.LIBR	RECFM =U BLKSIZE =16632	z/VSE BACKUP for Adabas Native SQL	approx. 30 blocks
SQL*nnn*.ERRN	RECFM =VB BLKSIZE =4628 LRECL =4624	The Adabas Native SQL preprocessor error messages. It was created using the utility program ERRLODUS.	approx. 280 blocks

The library contains the following phases:

Name	Description
ADASQLA.PHASE	The executable Adabas Native SQL preprocessor (load module) that generates Ada code.
ADASQLC.PHASE	The executable Adabas Native SQL preprocessor (load module) that generates COBOL code.
ADASQLF.PHASE	The executable Adabas Native SQL preprocessor (load module) that generates FORTRAN code.
ADASQLP.PHASE	The executable Adabas Native SQL preprocessor (load module) that generates PL/I code.

The library contains the following obj's:

Name	Description
APPEX1DA.OBJ	An Adabas Native SQL module (Ada).
APPEX1DC.OBJ	An Adabas Native SQL module (COBOL).
APPEX1DF.OBJ	An Adabas Native SQL module (FORTRAN).
APPEX1DP.OBJ	An Adabas Native SQL module (PL/I).
APPEX2S.OBJ	An Adabas Native SQL module.
APPEX3S.OBJ	An Adabas Native SQL module.
APPMSG.OBJ	An Adabas Native SQL module.
APPTIME.OBJ	An Adabas Native SQL module.
ASQL1A.OBJ	An Adabas Native SQL module (Ada).
ASQL1C.OBJ	An Adabas Native SQL module (COBOL).
ASQL1F.OBJ	An Adabas Native SQL module (FORTRAN).
ASQL1P.OBJ	An Adabas Native SQL module (PL/I).
CMDIX.OBJ	Module that generates the names of the record buffer.
FGFID.OBJ	Module that generates a global format ID.
FJOBNAME.OBJ	Module that gets the job name.
PRPABEND.OBJ	This module is used to abend an application program if an error occurred.
PRTRACE.OBJ	Module that traces the record buffers and format buffers during the execution of the application program if MODE TRACE is set (object module, COBOL).
RESPINT.OBJ	The response code interpretation routine (object module, COBOL).
SQFRDATE.OBJ	Routine which converts format D number to numeric date.
SQFRTIME.OBJ	Routine which converts format T number to numeric date and numeric time.
SQTODATE.OBJ	Routine which converts numeric date to format D number.
SQTOTIME.OBJ	Routine which converts numeric date and numeric time to format T number.

The library contains the following sources:

Name	Description
AEX1.A....AEX3.A	Ada examples using various Adabas Native SQL statements.
TYPESADA.A	Data definitions for use in Ada programs. This file must be compiled before using the Adabas Native SQL preprocessor with Ada source programs.
CEX1.C...CEX3.C	COBOL examples using various Adabas Native SQL statements.
CEXC.C	A COBOL example using Adabas Native SQL under CICS.
PRTFLOW.C	The source code of the routine that prints a flow-trace of all executed Adabas Native SQL statements during the execution of the program if MODE FLOW is set (COBOL).

Name	Description
PRTRACE.C	The source code of the routine that traces the record buffers and format buffers during the execution of the application program if MODE TRACE is set (COBOL).
PRTRCICS.C	The source code of the routine that traces the record buffers and format buffers during the execution of the application program if MODE TRACE is set (COBOL; for use in CICS programs).
RESPCICS.C	The source code of the response code interpretation routine (COBOL; for use in CICS programs).
RESPINT.C	The source code of the response code interpretation routine (COBOL).
SQFRDATE.C	Routine which converts format D number to numeric date.
SQFRTIME.C	Routine which converts format T number to numeric date and numeric time.
SQTODATE.C	Routine which converts numeric date to format D number.
SQTOTIME.C	Routine which converts numeric date and numeric time to format T number.
FEX1.F...FEX3.F.F	FORTRAN examples using various Adabas Native SQL statements.
PRTFLO.F	The source code of the routine that prints a flow-trace of all executed Adabas Native SQL statements during the execution of the program if MODE FLOW is set (FORTRAN).
PRTRAC.F	The source code of the routine that traces the record buffers and format buffers during the execution of the application program if MODE TRACE is set (FORTRAN).
RESPF.F	The source code of the response code interpretation routine (FORTRAN).
PEX1.P...PEX3.P	PL/I examples using various Adabas Native SQL statements.
PRTFLOP.P	The source code of the routine that prints a flow-trace of all executed Adabas Native SQL statements during the execution of the program if MODE FLOW is set (PL/I).
PRTRACP.P	The source code of the routine that traces the record buffers and format buffers during the execution of the application program if MODE TRACE is set (PL/I).
RESPPL1.P	The source code of the response code interpretation routine (PL/I).

If necessary, the source modules may be edited to accommodate installation-specific requirements. The original source modules should be retained for future reference.

The members ADASQL.J and SQLJCL.J contain the following procedures with parameterized data. The parameters are documented.

Name	Procedures
ADASQL.J	A sample JCL procedure to link Adabas Native SQL. A sample procedure to generate the procedure SQLLIBS with the utility MAINT.
SQLJCL.J	A sample JCL procedure to preprocess, compile, link and execute an Ada program. A sample JCL procedure to preprocess, compile, link and execute a COBOL program. A sample JCL procedure to preprocess, compile, link and execute a FORTRAN program.

Name	Procedures
	A sample JCL procedure to preprocess, compile, link and execute a PL/I program. A sample JCL procedure to preprocess an Ada, COBOL, FORTRAN or PL/I program. The sub-procedure ASQL must be called with one of the following parameters as appropriate : "SQL='ADASQLA.PHASE'" (Ada) "SQL='ADASQLC.PHASE'" (COBOL) "SQL='ADASQLF.PHASE'" (FORTRAN) "SQL='ADASQLP.PHASE'" (PL/I) Also, the global parameter `LANG` must be set to the correct value.

4 INSTALLATION

Predict Version 3.4 or above is a prerequisite for the installation of Adabas Native SQL. Please check which version of Predict is installed.

The Adabas files and fields that will be used by Adabas Native SQL application programs must be defined in the data dictionary.

This chapter covers the following topics:

Copying to a z/VSE Disk

If you are using System Maintenance Aid (SMA), refer to the SMA documentation (included on the current edition of the Natural documentation CD).

If you are not using SMA, follow the instructions below.

This section explains how to:

- Copy data set COPYTAPE.JOB from tape to library.
- Modify this member to conform with your local naming conventions.

The JCL in this member is then used to copy all data sets from tape to disk.

If the datasets for more than one product are delivered on the tape, the member COPYTAPE.JOB contains the JCL to unload the datasets for all delivered products from the tape to your disk, except the datasets that you can directly install from tape, for example, Natural INPL objects.

After that, you will have to perform the individual install procedure for each component.

Step 1: Copy data set COPYTAPE.JOB from tape to disk

The data set COPYTAPE.JOB (file 5) contains the JCL to unload all other existing data sets from tape to disk. To unload COPYTAPE.JOB, use the following sample JCL:

```
* $$ JOB JNM=LIBRCAT,CLASS=0,                                          +
* $$ DISP=D,LDEST=(*,UID),SYSID=1
* $$ LST CLASS=A,DISP=D
// JOB LIBRCAT
* ***************************************
*     CATALOG COPYTAPE.JOB TO LIBRARY
* ***************************************
// ASSGN SYS004,nnn                              <------  tape address
// MTC REW,SYS004
// MTC FSF,SYS004,4
ASSGN SYSIPT,SYS004
// TLBL IJSYSIN,'COPYTAPE.JOB'
// EXEC LIBR,PARM='MSHP; ACC S=lib.sublib'       <------- for catalog
```

```
/*
// MTC REW,SYS004

ASSGN SYSIPT,FEC
/*
/&
* $$ EOJ
```

Where:

NNN	is the tape address.
lib.sublib	is the library and sublibrary of the catalog.

Step 2: Modify COPYTAPE.JOB

Modify COPYTAPE.JOB to conform with your local naming conventions and set the disk space parameters before submitting this job.

Step 3: Submit COPYTAPE.JOB

Submit COPYTAPE.JOB to unload all other data sets from the tape to your disk.

Modifying the Sample JCS Procedures

The following specifications in the procedures should be adapted to suit the system environment. Use the JCS procedure SQLJCL.J to adapt all the JCS procedures.

Modifying the Global Parameters

The global parameters for Adabas Native SQL must also be modified.

The error message texts for decoding preprocessor-time errors are typically stored in the Natural system file, specified by the global SYSFILE FNAT parameter.

The error messages for decoding run-time errors are typically stored in the data dictionary file. If this is not the case, use the global parameter `ABORT FILE` to specify the location (number) of this file.

See also chapter *Global Parameters* in the *Adabas Native SQL Reference Manual* for further information.

Typical global parameter definitions:

```
SYSFILE FDIC=(1,11) FNAT=(1,8).
LANG COBOL.
XREF ON.
ABORT FILE=8.
```

Relinking Adabas Native SQL

If a different Adabas interface module is to be used instead of the standard module ADAUSER, Adabas Native SQL must be relinked. The procedure SQLLINK.J can be used to link the module ADASQL.

The module ADAUSER reads ADARUN parameters from SYSIPT. If ADALNK is used, no ADARUN card is necessary.

When relinking Adabas Native SQL, be sure to use the correct language-dependent modules:

Language	Modules
Ada	ASQL1A.OBJ and APPEX1DA.OBJ
COBOL	ASQL1C.OBJ and APPEX1DC.OBJ
FORTRAN	ASQL1F.OBJ and APPEX1DF.OBJ
PL/I	ASQL1P.OBJ and APPEX1DP.OBJ

Testing Adabas Native SQL

Before testing the examples, check that the files EMPLOYEES and VEHICLES are loaded, and that they are documented in the Predict data dictionary. If the file definitions are not yet present in the data dictionary, they can be loaded from the file PRD*nnn*.DEMO, which can be found on the Predict installation tape. The notation *nnn* represents the Version, Release and SM level of Predict at your site. Ensure that the relationships for soft coupling are defined in the data dictionary. If you wish to use Adabas Native SQL to preprocess FORTRAN programs, ensure that the necessary language synonyms are defined in the data dictionary. See for example Appendix B in the *Adabas Native SQL Reference Manual*.

Adapt the job SQLJCL.J and the appropriate section in this(J.COBOLGO, J.FORTGO, J.PLIGO) to perform the following steps:

1. Preprocess the program with Adabas Native SQL;
2. Compile the preprocessed program (COBOL, FORTRAN or PL/I).

3. Link-edit the program. The following modules may be called by the user program at runtime:

COBOL	COBOL/CICS	PL/I	Description
RESPINT (C)	RESPCICS (C)	RESPINT (C,P)	Error handling routine.
PRTRACE (C)	PRTRCICS (C)	PRTRACE (C,P)	Runtime trace routine. Activated by the error handling routine if `MODE TRACE` is set.
PRTFLOW (C)	--	PRTFLOW (C,P)	Runtime Adabas Native SQL statement printing. Activated if `MODE FLOW` is set.
PRPABEND (A)	CICSABEND	PRPABEND (A)	Terminates program execution. Called by the error handling routine.
SQFRDATE (C)	SQFRDATE (C)	SQFRDATE (C)	Routine which converts format D number to numeric date.
SQFRTIME (C)	SQFRTIME (C)	SQFRTIME (C)	Routine which converts format T number to numeric date and numeric time.
SQTODATE (C)	SQTODATE (C)	SQTODATE (C)	Routine which converts numeric date to format D number.
SQTOTIME (C)	SQTOTIME (C)	SQTOTIME (C)	Routine which converts numeric date and numeric time to format T number.

Ada	*FORTRAN*		*Description*
RESPF (F)	RESPF(F)		Error handling routine.
PRTRAC (F)	PRTRAC (F)		Runtime trace routine. Activated by the error handling routine if `MODE TRACE` is set.
PRTFLO (F)	PRTFLO (F)		Runtime Adabas Native SQL statement printing. Activated if `MODE FLOW` is set.
PRPABEND (A)	--		Terminates program execution. Called by the error handling routine.
SQFRDATE (C)	SQFRDATE (C)		Routine which converts format D number to numeric date.
SQFRTIME (C)	SQFRTIME (C)		Routine which converts format T number to numeric date and numeric time.
SQTODATE (C)	SQTODATE (C)		Routine which converts numeric date to format D number.
SQTOTIME (C)	SQTOTIME (C)		Routine which converts numeric date and numeric time to format T number.

Notes:

a. The FORTRAN error handling routine RESPF always returns condition code 8 when the job step terminates. The other error handling routines return the Adabas response code.

b. The characters in parentheses indicate the language in which the routines are written (Assembler, COBOL, FORTRAN or PL/I). Only the Assembler and COBOL modules are supplied in object form; if you wish to use FORTRAN or PL/I modules, you must compile the source modules.

c. The COBOL object modules of the routines RESPINT, PRTRACE and PRTFLOW supplied by Software AG can be linked with PL/I object modules to produce an executable load module. If you wish to use the PL/I versions of these routines, you must rename them: RESPPL1 - RESPINT, PRTRACP - PRTRACE, PRTFLOP - PRTFLOW, and then compile them, replacing the Software AG-supplied COBOL object modules by the PL/I object modules. The global parameter `ABORT PLI` should be coded. The procedure as supplied links the standard Adabas interface module ADAUSER to the application program. If a different module is used, the procedure must be modified accordingly.

4. Execute the program.

   If the program is to be passed through other preprocessors in addition to Adabas Native SQL, the corresponding job step should be included in the procedure following the Adabas Native SQL step and preceding compilation.

Symbolic Device Names

Adabas Native SQL refers to files using the following symbolic device names:

Device Name	Description
SYS000	The Adabas parameters that are used by Adabas Native SQL at preprocessor time. (Second input.)
SYS007	The global parameter definitions. (First input.) LRECL=80,RECFM=FB
SYS008	The source program to be preprocessed. (Third input.) LRECL=80,RECFM=FB
SYS010	A temporary work file. LRECL=80,RECFM=FB
SYS016	Adabas Native SQL writes messages at preprocessing time to this dataset.
SYS018	The output of the preprocessor. Embedded Adabas Native SQL statements in the source program appear in this file as comments and are followed by the generated Ada, COBOL, FORTRAN or PL/I code. LRECL=80,RECFM=FB
SYS019	A temporary work file. LRECL=80,RECFM=FB

At run time, the application program generated by Adabas Native SQL refers to files using the following symbolic device names (in addition to any other user-defined files):

Device Name	Description
SYS000	The Adabas parameters that are used by the ADARUN module at run time.
SYS013	If the TRACE facility is used, the trace output is written to this file. LRECL=80,RECFM=FB

5 INSTALLATION CHECKLIST

- Verify that Predict Version 3.4 or later is installed.
- Verify that the files and fields that are to be used by Adabas Native SQL applications are documented in the data dictionary. Adabas Native SQL recognizes only the PREDICT file types *Adabas file* and *Adabas userview*. Other file types are ignored by the Adabas Native SQL preprocessor.
- Use RESTORE to load the libraries from tape to disk.
- Verify that the error messages have been loaded (ERRLODUS).
- Edit the member SQLJCL.J as required.
- If an ADABAS interface module other that ADAUSER is to be used, relink ADABAS Native SQL using ADASQL.J.
- Ensure that the standard test files EMPLOYEES and VEHICLES are available and that they are correctly documented in the data dictionary. In particular, ensure that the file numbers and the database ID are correct. Ensure that the relationships for soft coupling are defined in the data dictionary. If you wish to use FORTRAN, ensure that appropriate language synonyms are defined in the data dictionary (see Appendix B in the *Adabas Native SQL Reference Manual*).
- Adapt the procedure SQLJCL.J to execute the test programs. Verify the output.

6 HINTS FOR USING ADABAS NATIVE SQL

This chapter covers the following topics:

Periodic Groups and Multiple Fields

If the maximum number of occurrences of periodic groups or multiple fields is known, specify this number in Predict. Otherwise Adabas Native SQL will allocate buffers using the default maximum values, resulting in wasted storage.

Group Structure of Periodic Groups

Correct use of the `GROUP STRUCT` attribute in Predict can save significant amounts of space in the format buffers that are used when accessing records containing periodic groups. See *Defining More Attributes of Fields* in section *Field* of chapter *Predefined Object Types* in the *Predict Reference Manual*.

Dynamic Command-IDs

If the database is accessed from many modules within one linked program, the global parameter `OPTIONS DYNAMCID.` should be specified. This ensures that a unique command-ID will be generated for each Adabas command if the same cursor-name is used in more than one module. Be aware that this can degrade run-time performance. See ***Improving Adabas Native SQL Efficiency*** and the section on using command-IDs in the *Adabas Command Reference Manual*.

Locating Errors

The TRACE and FLOW facilities, which are switched on by means of global parameters, can be used to find runtime errors in the application program.

Preprocessor Copy and Generate Facilities

The Adabas Native SQL preprocessor supports the COPY and GENERATE statements. These are compatible with the implementation of the COPY and GENERATE statements in the Predict preprocessor, so it should not normally be necessary to use both Adabas Native SQL and Predict preprocessors.

Restriction for MU Fields Within PE

The data dictionary definition of a multiple-value field within a periodic group should not specify a counter field.

Hyphens and Break Characters in PL/I Programs

Field names as defined in the data dictionary may include hyphens ("-") and/or break characters ("_"). With Adabas Native SQL, each reference to a field name must match the definition in the data dictionary exactly. However, when generating PL/I output, Adabas Native SQL will change all hyphens found in data dictionary definitions to break characters, since hyphens are not valid in PL/I identifiers.

With previous versions of Adabas Native SQL, it was not necessary for references to field names to match the definitions in the data dictionary exactly: all break characters found in the source program were converted to hyphens before being compared with the data dictionary definitions; then, when generating PL/I output, Adabas Native SQL changed all hyphens to break characters.

The global parameter `UNDERSCORE NO.` can be used to ensure that Adabas Native SQL processes hyphens and break characters in a manner compatible with earlier versions.

READ ISN Statement

After issuing the READ ISN statement with `OPTIONS SEQUENCE`, the program should check for end-of-file (ADACODE = 3). In some applications, it may be necessary to compare the ISN of the record that was read with the ISN that was specified in the `WHERE` clause of the statement.

PL/I - Margin Settings

When using the PL/I compiler, the margins should be set to (2,72) (these are the default values).

PL/I - Structure Variables in Superdescriptors

In PL/I, it is not possible to assign a structure variable to a superdescriptor in the WHERE clause. A string variable should be used in place of the structure variable.

Last Statement Restriction in COBOL/II

The last statement in a COBOL/II program should not be an Adabas Native SQL statement. If necessary, the statement `EXIT.` can be coded at the end of the program.

Adabas Native SQL Preprocessor Condition Codes

The Adabas Native SQL preprocessor can set the following condition codes:

Code	Meaning
0	No error was detected
4	One or more warning messages issued during processing.
8	One or more errors found during processing.
12	Adabas Native SQL abended.

Programs generated by the Adabas Native SQL preprocessor can cause various errors at runtime. Adabas errors are detected by the generated code. In COBOL and PL/I programs, the Adabas response code as described in the *Adabas Messages and Codes Manual* is returned to the calling procedure. In FORTRAN programs, response code 8 is returned to the calling procedure.

SQL0217 Error Message When Running the Precompiler

This message is:

```
SQL0217 FDIC DBID OR FDIC FNR DOES NOT MATCH THE CONTROL RECORD
```

This message occurs if a DDM from another system is loaded into the Natural FDIC file and thereafter the DBID and FNR of the DDM are changed online to the current FDIC's DBID and FNR.

To rectify this, start the Natural containing the Predict:

```
LOGON SYSDIC
MENU
D       for defaults
A       for Adabas Native SQL
<Enter>
.       to exit
```

This will reset the Adabas Native SQL control record.

Hints for Improving Adabas Native SQL Efficiency

Reducing Database Accesses

Using the HISTOGRAM and READ statements as much as possible in preference to FIND can help to reduce the number of accesses made to the database.

Command IDs

An explicit Adabas command ID is assigned whenever a cursor is declared in an Adabas Native SQL statement. Adabas Native SQL statements that are performed repeatedly should have a cursor, since the command ID derived from the cursor name is used by Adabas to indicate that the format buffer need not be translated repeatedly.

Variable Indices

Periodic groups and multiple fields should only be referenced using variable indices when this is unavoidable, since variable indices cause additional format translations and also additional RC commands to be executed.

Record Buffers

The number of record buffers generated by Adabas Native SQL can be reduced by coding UPDATE and INSERT statements with the `WHERE CURRENT OF cursor-name` clause but without the `SET` clause. This only applies if the UPDATE or INSERT statement and the statement referenced by the cursor-name contain the same field structure.

SELECT Clause

Program efficiency is improved if only those fields that are needed are coded in the SELECT clause. You may be tempted to write "SELECT *" instead of coding the name of each field, but this can result in excessively large record buffers and slower-running programs.

Global Format ID

Application programs, in particular online application programs, should use the the global options parameter OPTIONS GFORMAT. This can improve efficiency by reducing the number of format buffer translations that are performed at runtime.