Adabas System Coordinator

Adabas System Coordinator Versioning Tool

Version 8.1.2

June 2008

Adabas System Coordinator

This document applies to Adabas System Coordinator Version 8.1.2 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 Adabas System Coordinator Versioning Tool

This document describes the Adabas System Coordinator Versioning Tool which enables multiple versions of Add-on products to co-exist in the Adabas servers and the clients that use them.

The following products are currently supported for co-existence:

- Adabas System Coordinator Version 7.4.2
- Adabas Fastpath Version 7.4.2
- Adabas Vista Version 7.4.2
- Adabas SAF Security Version 7.4.2
- Adabas System Coordinator Version 8.1.2
- Adabas Fastpath Version 8.1.2
- Adabas Vista Version 8.1.2
- Adabas SAF Security Version 8.1.2

2 Components

The main components of the versioning tool are:

- Adabas Database
  - Module VERPOP. This is a special version of the System Coordinator database interface module (ADAPOP). At installation time, VERPOP is copied to a new versioning library and renamed to ADAPOP.
  - The database versioning table – module VERDBT. This is created from macro statements (VERDB) that describe the versions you want to support, and a module suffix for each version. The suffix enables multiple copies of required modules, at different versions, to be loaded and used concurrently. During installation, modules of a required version are copied to the versioning library and renamed with the selected suffix.
- Client
  - Modules VERLNK01 (for batch), VERLNK09 (for batch with ADALNKR) and VERCIC01 (for CICS). These are special versioning link modules that are renamed during installation.
  - The client versioning table modules – VERC01 (batch), VERC09 (batch with ADALNKR) and VERC03 (CICS). These are created from macro statements (VERCL), and are used to identify the (suffixed) link module to be used for each job and CICS transaction.
- Daemon
  - Module VERSCO. This module is used by the System Coordinator daemon (SYSCO) to identify the suffix to use for modules loaded by the daemon. At installation time, VERSCO is copied to the versioning library and renamed to SYSCOVER.
  - The daemon versioning table – module VERDMT. This is created from macro statements (VERDM) that describe the module suffixes to be used for each Coordinator daemon.

Notes:

1. It is possible to implement versioning in an Adabas server without implementing client versioning. In this case the database can be accessed by multiple clients, each running a single version of the add-ons.
2. It is possible to implement client versioning without implementing server versioning, but care must be taken to ensure that commands cannot be routed to a database that is not running the appropriate version.
3. The use of VERSCO and the daemon versioning table is optional. Its purpose is to enable you to use a common versioning library, with common module suffixes, for the Adabas servers and Coordinator daemon tasks. Multiple version support in the daemon is achieved by running multiple daemon instances.

3 Installation

The Versioning Tool modules, macros and sample jobs are supplied with Adabas System Coordinator. All files have the name prefix "VER".

The following table provides an overview of the steps required to install the Versioning Tool:

Step	Description
1	Create versioning load libraries.
2	Assemble the database versioning table. **Note:** This step is required for database versioning only.
3	Assemble the Coordinator daemon versioning table. **Note:** This step is required for daemon versioning only.
4	Create client versioning tables(s) and link module(s) for batch clients. **Note:** This step is required for client versioning only.
5	Create client versioning tables(s) and link module(s) for CICS clients. **Note:** This step is required for client versioning only.
6	Apply co-requisite product maintenance.
7	Copy and rename required product modules into the versioning library.
8	Add the versioning libraries to required job control.

to install the Versioning Tool, perform the following steps:

1 **Create versioning load libraries.**

- **Library for Versioning**

Software AG recommends that versioning is implemented by creating a new library to hold copies of product modules that have been renamed in support of versioning. This will protect the original modules. It will also enable new maintenance to be applied to the original modules, which can then be refreshed in the versioning libraries by re-running the appropriate installation jobs.

In the sample job control supplied with the product this library is called SAG.CORVER.ALLVERS.LOAD.

The library is added to the Joblib or Steplib concatenation of all servers, CICS tasks, batch Adabas server start-up jobs and Coordinator daemon tasks that require versioning support.

- **Additional Library for Client Versioning**

  If you are implementing client versioning, you need to create a second library that will be used for the special link modules (VERLNK01, VERCIC01, ADALNKnn, ADABASnn), and the client versioning tables.

  In the sample job control supplied with the product this library is called SAG.CORVER.CLIENT.LOAD.

  

  Note: It is imperative that these link modules are only used in the client environment. Therefore, they must not be placed in the ALLVERS load library that is used by the Adabas servers and Coordinator daemon.

2 **Assemble the database versioning table.**

Note: This step is required for database versioning only.

Use sample job VERI055.

The versioning table source consists of a number of VERDB macro statements. Here is an example:

```
VERDB SUFFIX=74,VRL=742
```

```
VERDB SUFFIX=81,VRL=811
VERDB TYPE=END
```

The database versioning table will have an entry for each version to be run. There are 2 parameters:

Parameter	Description
SUFFIX=	Identifies the suffix for all the required modules of all the products at that version.
VRL=	Denotes the version, release and maintenance level of the suffixed collection of modules. VRL needs to be set accurately because it is used internally by the software to ensure correct operation.

The following rules apply:

- The values for SUFFIX= are unique across all VERDB entries in the table.
- The values for VRL= are unique across all VERDB entries in the table.
- The table must end with VERDB TYPE=END.

3 **Assemble the Coordinator daemon versioning table (VERDMT).**

Note: This step is required for deamon versioning only.

Use sample job VERI060.

The daemon versioning table enables you to use the re-named Coordinator and product modules in the System Coordinator daemon. This ensures that the same modules at the same maintenance levels are available in both the Adabas and the System Coordinator daemon.

Note: Multiple version support in the daemon is achieved by running a separate daemon for each required version. The Coordinator daemon does not support multiple versions in a single daemon instance.

Note: This is an optional activity. You may still run the Coordinator daemon from the original product libraries, if desired.

The versioning table source consists of a number of VERDM macro statements. Here is an example:

```
VERDM SUFFIX=74,VRL=742,JOBNAME=SYSCO742
VERDM SUFFIX=81,VRL=811,JOBNAME=SYSCO811
VERDM TYPE=END
```

The daemon versioning table will have an entry for each daemon to be run. There are 3 parameters:

Parameter	Description
JOBNAME=	Identifies the jobname of a System Coordinator daemon.
SUFFIX=	Identifies the suffix for all the required modules of all the products at that version.
VRL=	Denotes the version, release and maintenance level of the suffixed collection of modules. VRL needs to be set accurately because it is used internally by the software to ensure correct operation.

4 **Create client versioning table(s) and link module(s) for batch clients.**

Note: This step is required for client versioning only.

Use sample job VERI065.

The batch versioning table is named VERC01 and consists of a number of VERCL macro statements.

- **Create the batch versioning table**

  Here is an example:

```
JOBTYPE=BATCH,WTO=YES,SVCPROP=YES   (see Note a)
VERCL  VRL=742,SUFFIX=74 (see Note b)
VERCL  VRL=811,SUFFIX=81 (see Note c)
VERCL  JOBNAME=TSOUSRA (see Note d)
VERCL  JOBNAME=TSOUSRB (see Note d)
VERCL  JOBNAME=BATCHJB1 (see Note d)
VERCL  JOBNAME=BATCHJB2 (see Note d)
VERCL  TYPE=END (see Note e)
```

Notes:

1. Defines the table type (BATCH includes TSO) and whether or not the contents of the table are to be displayed on the system console at startup. SVCPROP specifies whether or

not the SVC number supplied by ADARUN is to be propagated to the link modules (the default is NO). If `SVCPROP=YES` is specified, all link modules will be modified to use the SVC number supplied by ADARUN. `SVCPROP=YES` may only be specified for table type BATCH. If you specify `WTO=YES,SVCPROP=YES`, an additional message is displayed after the contents of the table: MC0102I ADARUN SVC number will be propagated

2. Defines the default link module. You must define one and one only default link module – that is, one with no following `JOBNAME` entries. The `VRL` is used by VERLNK01 to route internal commands from the online services to the correct link module.
3. Defines an alternate link module, specifying its `VRL` and the suffix to be used which must be the same as that used when copying product modules (see copy required link modules below).
4. Defines those TSO users and batch jobs for which Adabas calls are to be routed through the preceding link module.
5. Generates the table.

The resulting load module must be named VERC01 and the versioning table load modules must be linked REUS, – not RENT.

- **Copy VERLNK01 to the client versioning library**

Copy VERLNK01 to your client versioning library as ADALNK, using sample job VERI085.

Important: This library must not be made available to any target in the Software AG network which acts as a DBID – such as Adabas, Broker, Com-Plete, Entire System Server, Net-Work, System Coordinator daemon etc.

- **Copy required link modules to the client versioning library**

Copy your existing ADALNK modules to the client versioning library, renaming them appropriately (VERI086 is a sample job). Using the above example, you would need:

ADALNK74 742 ADALNK, containing Adabas Fastpath and Adabas Vista versions 742.

ADALNK81 811 ADALNK, containing Adabas Fastpath, Adabas Vista and System Coordinator 811.

Important: The renamed copies of your ADALNK modules must contain the correct SVC number, default dbid and buffer lengths. Default dbid and buffer lengths should already be set, however SVC number may not be. Sample job VERI087 shows how to change the SVC number.

The procedure for using versioning with a reentrant ADALNK (ADALNKR) is identical to standard batch, with these exceptions:

- VERLNR01 must be renamed to ADALNK (or whatever you usually call ADALNKR).

- The batch versioning table must be named VERC09.
- The renamed ADALNKR modules must be called ADALNR*xx* where *xx* is a suffix defined in VERC09. You must have an ADALNR*xx* module for each defined suffix.
- The caller must provide a MODIFIED area as the seventh parameter, with the top bit of the address set.
- Neither VERLNR01 nor VERC09 should be linked reentrant. Also, if you set `SVCPROP=YES`, the ADALNR*xx* modules must not be linked reentrant.

5 **Create client versioning table(s) and link module(s) for CICS clients.**

Note: This step is required for client versioning only.

- **Prepare CICS clients**

  There are several prerequisites which must be fulfilled before you can use the CICS client versioning facility:

  - All transactions which will use it must be defined with a minimum TWA size equal to the value specified as the VERCL `COFF` parameter plus 32 bytes. So, if you specify `COFF=96`, TWA size must be at least 128.
  - Adabas 7.4 link modules must be assembled with the ADAGSET parameter `PARMTYP=ALL`.
  - For Adabas 8.1, ensure that `PARMTYP=ALL` is set in LGBLSET (this is the default).
  - You must not mix TRUE-enabled link modules with non-TRUE-enabled link modules. Either all link modules must use the TRUE or none.
  - PPT entries must be defined and installed for the Adabas client versioning link module (ADABAS, or other name you may choose, or if using Adabas Version 8, ADACICS); for VERC03, and for all renamed Adabas CICS link modules, Adabas TRUEs and Adabas PLT programs (LNKENAB and ADACIC0). You may require additional PPT entries for add-on product modules (specifically – Version 742 and later modules can be loaded from DFHRPL if PPT entries are defined for the product modules).

    Additionally, the PPT entries for ADABAS (or ADACICS) and VERC03 *must* specify Resident.

- **Create the CICS versioning table**

  Use sample job VERI065.

The CICS versioning table is named VERC03 and consists of a number of VERCL macro statements.

```
VERCL  JOBTYPE=CICS,COFF=96,TSPR=VERC (see Note a)
VERCL  VRL=742,SUFFIX=74 (see Note b)
VERCL  VRL=811,SUFFIX=81 (see Note c)
VERCL  TRAN=AA73 (see Note d)
VERCL  TRAN=BB73 (see Note d)
VERCL  TYPE=END (see Note e)
```

Notes:

1. Defines the table type, the offset in TWA to be used for saving context information (COFF) and the Temporary Storage queue prefix to be used for saving context information across transactions. The TS queue name consists of the specified prefix (default VERC) and the terminal ID.
2. The first `VRL` entry defines the default link module. You must define one and one only default link module – that is, one with no following `TRAN` entries. `VRL` identifies the version of the products contained in that link module and must be unique.
3. Subsequent `VRL` entries define alternate link modules, specifying their `VRL` and the suffix to be used which must be the same as that used when copying product modules (see copy required link modules below).
4. Defines those transaction codes for which Adabas calls are to be routed through the preceding link module.
5. Generates the table.

The resulting load module must be named VERC03 and the versioning table load modules must be linked REUS, not RENT.

- **Link-edit VERCIC01 into the client versioning library**

  You must also link VERCIC01 with your CICS stubs, naming the output load module appropriately (by default: ADABAS). Installation job VERI085 includes sample JCL.

- **Copy required link modules to the client versioning library**

  If you do not use the enhanced Adabas installation (TRUE), you can simply copy your existing CICS link modules to the client versioning library, renaming them appropriately (VERI086 contains sample JCL). Starting with Adabas Version 8, use of the TRUE is required, so if you wish to include an Adabas Version 8 link module in client versioning, you must use the enhanced installation for earlier Adabas versions.

- **Enhanced installation using TRUE**

If you use the Adabas TRUE, you must create an ADAENAB and an ADATRUE for each link module, as well as the link module itself. For each suffix defined in VERC03, you will need modules called:

Module	Description
ADAENA*xx*	PLT TRUE enabler
ADATRU*xx*	Adabas Version 7.4 TRUE
ADACICT	Adabas Version 8.1 TRUE
ADABAS*xx*	Link module

where *xx* is the suffix and the first 6 characters of the module name must be as shown. ADAENA*xx* must refer to the versioning tool, rather than its associated link module.

For Adabas 7.4 link modules, the easiest way to do this is to take copies of your existing installation jobs (CICCASM, CICEASM and CICTASM). Here is an example of how to create a set of modules for a link module with suffix 74. The example assumes that VERCIC01 has been linked as ADACICS.

- Change ADAGSET to specify

```
ENTPT=ADACICS,ENABNM=ADAENA74,TRUENM=ADATRU74,
PARMTYP=ALL,SVCNO=nnn and any other required parameters
```

- Assemble LNKENAB and link it as ADAENA74
- Assemble LNKTRUE and link it as ADATRU74
- Assemble LNKOLSC and LNKOLM and link them (together with any other modules, such as CORS03) as ADABAS74
- Define PPT entries for ADAENA74, ADATRU74, ADABAS74 (not forgetting ADACICS and VERC03)
- Add ADAENA74 to the CICS PLT

For an Adabas 8.1 link module, the procedure is simpler. Here is an example of how to create a set of modules for a link module with suffix 81. VERCIC01 must be linked as ADACICS:

- Copy the supplied ADACIC0 to ADAENA81
- Copy the supplied ADACICS to ADABAS81

> **Note:** The Adabas 8.1 TRUE has a fixed name (ADACICT), so you may only have one Adabas 8.1 link module in the versioning table.

- Define PPT entries for ADAENA81, ADABAS81, ADACICS and VERC03

- Add ADAENA81 to the CICS PLT

Now, during CICS initialisation you will see 2 sets of ADAK messages. Note that the ADAK045 messages for both ADATRU74 and ADACICT will say "in use by ADABAS link routine ADACICS". This is because the versioning module is invoked by both ADAENA74 and ADAENA81. The versioning module ensures that the correct TRUE and link module are initialised.

Finally, you must ensure that your renamed Adabas link modules and add-on product modules are available in the CICS DFHRPL and Steplib concatenations.

The CICS versioning tool supports callers using both the Direct Call Interface and the EXEC CICS LINK interface. It always uses the DCI to call the various Adabas link modules. Only command-level link modules from Version 7.4 and above are supported.

When invoked via EXEC CICS LINK, it first looks for the Adabas parameter list in COMMAREA and then in TWA.

Do not use CICS NEWCOPY to reload the versioning tool, versioning table or any Adabas link modules defined in the versioning table.

If you use the COR Node Error Program, you must be sure to change the sample source so that it starts the CORNEP transaction for each link module.

Important: Certain fixed offsets in the Adabas link module(s) contain information which is required by the caller of ADALNK or ADABAS. For example, length of userinfo. If you use other than default values, you must zap the correct values into VERLNK and VERCIC. Sample job VERI088 may be used. Also, the renamed copies of your ADALNK modules must contain the correct SVC number, default dbid and buffer lengths. Default dbid and buffer lengths should already be set, however SVC number may not be. Sample job VERI087 shows how to change the SVC number.

6 **Apply co-requisite maintenance.**

The following maintenance fixes must be applied before using versioning with these products:

Product	Maintenance Fix
Adabas Version 742	AI742030
Adabas System Coordinator Version 742	Install the COR L005 load library update
Online Services INPL updates	IS06
Adabas Fastpath Version 742	AW742081
Adabas Vista Version 742	AV742017
Adabas SAF Security Version 742	AX742002

7 **Copy and rename required product modules into the versioning library.**

Customize and use the following sample jobs:

Version 742: VERI080D

Version 811: VERI080E

The sample jobs indicate all of the modules that require copying and renaming. No other product modules should be renamed.

Note: The System Coordinator module ADAPOB must not be renamed. If you have multiple versions of this module, ensure that the one in use is at the highest level.

8 **Add the versioning libraries to required job control.**

The versioning llibrary (the ALLVERS Library) must be added to the Joblib or Steplib concatenation of all Adabas server start-up jobs that require versioning support. The library should be added at the top of the concatenation, so that the special version of ADAPOP will be invoked at nucleus start-up.

If you want to run the System Coordinator daemon from the same versioning load library, you should also add the ALLVERS library first in the daemon Joblib or Steplib concatenation.

The ALLVERS load library and the client versioning load library must be added to the top of the Steplib concatenation for any batch jobs that require versioning support.

For CICS, the client versioning load library and the ALLVERS load library must be added to the DFHRPL and Steplib concatentations.

4 Operation

The database versioning tool will issue one or more of the following messages during nucleus initialization. These can be used to verify that the software is installed correctly, and the correct product/version sets modules have been located.

Message	Description
VER001 VER001I	VERPOP initialized
VER002 VER002I	ADAPOP*nn* loaded for Version *vrl*
VER003 VER003S	No versioning table found, cannot continue
VER004 VER004W	No versioning products found, calls suppressed
VER005 VER005W	ADAPOP*nn* not found for Version *vrl*
VER006 VER006I	ADA*ppp* loaded for Version 713
VER007 VER005W	ADA*ppp* not found for Version 713

Execution of more than one version will have an impact on the memory allocation for the database. Memory region sizes may need to be reviewed.