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, client application jobs (for example, batch jobs, TSO CLISTs, CICS jobs), 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.

Notes:

1. This step is required for database versioning only.
2. Database versioning is used when a database is accessed by client jobs that use different versions of System Coordinator and its co-products. For example, if you have a CICS job using System Coordinator version 8.2 and another using version 8.1 accessing the same databases, those databases need to use database versioning.

Use sample job VERI055.

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

```
VERDB SUFFIX=81,VRL=812
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 daemon 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=81,VRL=812,JOBNAME=SYSCO812
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.

Batch versioning configuration is managed using macro settings to produce a configuration module, loaded dynamically at runtime. Here is an overview of what must or can optionally be defined:

- The (mandatory) first element introduces the defaults for Batch. This element defines the defaults, and these settings are cascaded to the specific versioned link modules defined below, and optionally overridden by individual elements.

- One or more elements identifying the versioned link module(s) to be used. There must be at least one. One of these acts as the default versioned link module. The others are each followed by one or more job-name elements that are a list of the job names that will use the versioned link module that precedes them. Where a job name is not matched (at runtime) it will use the default element's versioned link module defined here.

- The end of the settings definitions is specified.

There are various options for using a global module (generically referred to as LNKGBLS) when using versioning. These options accommodate sites wishing to:

- Enforce use of the dynamically loaded global module (generically referred to as LNKGBLS).

- Disallow use of the dynamically loaded global module.

- Enforce use of the linked-in global module.

- Disallow use of the linked-in global module.

- Allow/disallow versioned dynamic global module, paired with the versioned link module.

- Allow propagation of ADARUN DBID= and/or SVC= settings into the dynamically loaded, versioned global module.

- Mixtures of these options across different link modules.

Here is an example versioning configuration:

```
VERCL JOBTYPE=BATCH,       Batch defaults
      OVERIDES=YES,
      WTO=NO,
      LNKGBLXX=YES,
      BINDGBLS=IGNORED,
      DBID=DDCARD,
      SVC=DDCARD
VERCL VRL=823,             8.2.3 module, the default
      DEFAULT=YES,
      SUFFIX=82
VERCL VRL=814,             8.1.4 module
      SUFFIX=81,
      LNKGBLXX=YES,
      BINDGBLS=IGNORED,
      DBID=15,
      SVC=ASIS
VERCL JOBNAME=jobname,     Use 8.1.4
      BINDGBLS=IGNORED,
      DBID=15,
      SVC=ASIS
VERCL TYPE=END             The end.
```

Controls for the Default BATCH Settings

This section describes the controls which can be used for the default BATCH settings:

Control	Description
JOBTYPE=BATCH	BATCH for batch configuration entries defaults.
OVERIDES=YES|NO	YES: Allow override settings for individual versioned elements. NO: The settings in this element cannot be overridden.
WTO=YES|NO	YES: Details of the configuration will appear on the console. NO: The configuration settings are silent.
LNKGBLXX=YES|NO	YES: A versioned global module LNKGBL*xx* (where *xx* is the suffix) is to be used, paired with the versioned link module of the same suffix. For re-entrant link modules, the versioned global module is named LNRGBL*xx*. NO: No versioned global module LNKGBL*xx* (where *xx* is the suffix) is to be used even when using a suffixed link module.
BINDGBLS=IGNORED|REQUIRED	IGNORED: Dynamically loaded global modules must be used, if one cannot be found an error is raised. REQUIRED: Dynamically loaded global modules are ignored in favor of using the module that is linked with the link module.
DBID=DDCARD|ASIS|number	DDCARD: The value from the `ADARUN DBID=` setting is to be propagated to the suffixed dynamic global module (if used). **Note:** In order for this `ADARUN DBID= DDCARD` propagation to occur the ADARUN LNKGNAME= setting must also be used. ASIS: The value in the global module (whichever one is used) is taken "as is"; any other values (such as `ADARUN DBID=`) is ignored. Number: A specific default database number.
SVC=DDCARD|ASIS|number	DDCARD: The value from the `ADARUN SVC=` setting is to be propagated to the dynamic global module that is suffixed (if used). **Note:** In order for this `ADARUN SVC= DDCARD` propagation to occur the `ADARUN LNKGNAME=` setting must also be used. ASIS: The value in the global module (whichever one is used) is taken "as is"; any other values (such as `ADARUN SVC=`) is ignored. Number: A default SVC number.

Controls for the Versioned Link Module Entry Settings

This section describes the controls which can be used for the versioned link module entry settings:

Control	Description
VRL=814|823	814: This identifies an 8.1.4 link module. 823: This identifies an 8.2.3 link module.
DEFAULT=NO|YES	YES: This identifies the default link module. Only one default link module is allowed. Do not define any jobs elements for the default link module; it is used by any job not specified to use another versioned link module. NO: This link module is only used by the jobs listed immediately following it.
SUFFIX=*xx*	*xx*: This identifies the two character suffix to be used when loading the link module and the global module (where appropriate). Example: `SUFFIX=81` would cause ADALNK81 to be loaded (or ADALNR81 for reentrant operation). Also, where appropriate LNKGBL81 is loaded too in the situation where `SUFFIX=81` is used.
The following controls are only allowed if `OVERIDES=YES` is used. They default to the overall values specified on the JOBTYPE definition.	
LNKGBLXX=YES|NO	YES: A versioned global module LNKGBL*xx* (where *xx* is the suffix) is to be used, paired with the versioned link module of the same suffix. NO: No versioned global module LNKGBL*xx* (where *xx* is the suffix) is to be used even when using a suffixed link module.
BINDGBLS=IGNORED|REQUIRED	IGNORED: Dynamically loaded global modules must be used, if one cannot be found an error is raised. REQUIRED: Dynamically loaded global modules are ignored in favor of using the module that is linked with the link module.
DBID=DDCARD|ASIS|number	DDCARD: The value from the `ADARUN DBID=` setting is to be propagated to the suffixed dynamic global module (if used). **Note:** In order for this `ADARUN DBID= DDCARD` propagation to occur the ADARUN LNKGNAME= setting must also be used. ASIS: The value in the global module (whichever one is used) is taken "as is"; any other values (such as `ADARUN DBID=`) is ignored. Number: A specific default database number.

Control	Description
SVC=DDCARD|ASIS|number	DDCARD: The value from the ADARUN SVC= setting is to be propagated to the dynamic global module that is suffixed (if used). **Note:** In order for this ADARUN SVC= DDCARD propagation to occur the ADARUN LNKGNAME= setting must also be used. ASIS: The value in the global module (whichever one is used) is taken "as is"; any other values (such as ADARUN SVC=) is ignored. Number: A default SVC number.

Controls for Specific Job Name Entry Settings

This section describes the controls which can be used for specific job name entry settings:

Control	Description
JOBNAME=jobname	This identifies the jobs that are to use the link module version identified prior to this job-name list.
The following controls are only allowed if `OVERIDES=YES` is used. They default to the values specified on or inherited by the preceding link module definition.	
LNKGBLXX=YES|NO	YES: A versioned global module LNKGBL*xx* (where *xx* is the suffix) is to be used, paired with the versioned link module of the same suffix. NO: No versioned global module LNKGBL*xx* (where *xx* is the suffix) is to be used even when using a suffixed link module.
BINDGBLS=IGNORED|REQUIRED	IGNORED: Dynamically loaded global modules must be used, if one cannot be found an error is raised. REQUIRED: Dynamically loaded global modules are ignored in favor of using the module that is linked with the link module.
DBID=DDCARD|ASIS|number	DDCARD: The value from the `ADARUN DBID=` setting is to be propagated to the suffixed dynamic global module (if used). **Note:** In order for this `ADARUN DBID= DDCARD` propagation to occur the ADARUN LNKGNAME= setting must also be used. ASIS: The value in the global module (whichever one is used) is taken "as is"; any other values (such as `ADARUN DBID=`) is ignored. Number: A specific default database number.
SVC=DDCARD|ASIS|number	DDCARD: The value from the `ADARUN SVC=` setting is to be propagated to the dynamic global module that is suffixed (if used). **Note:** In order for this `ADARUN SVC= DDCARD` propagation to occur the `ADARUN LNKGNAME=` setting must also be used. ASIS: The value in the global module (whichever one is used) is taken "as is"; any other values (such as `ADARUN SVC=`) is ignored. Number: A default SVC number.

The resulting load module must be named VERC01 for standard batch or VERC09 for re-entrant batch and the versioning table load modules must be linked REUS, – not RENT. Sample job VERI065 shows how to assemble and link the versioning table.

Copy the client versioning modules to the client versioning library

Copy VERLNK01 to your client versioning library as ADALNK and VERLNR01 as ADALNKR, using sample job VERI085 (making sure the real ADALNK and ADALNKR are not lost). Certain fixed offsets in the Adabas link module contain information which is required by some callers of ADALNK. For example, length of the USER and Review extensions. If you use other than default values, you must zap the correct values into the ADALNK and ADALNKR copies of VERLNK01 and VERLNR01. Sample job VERI088 may be used to do this.

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 and LNKGBLS modules to the client versioning library, renaming them to have the appropriate suffix:

```
ADALNKxx and LNKGBLxx for standard batch
ADALNRxx and LNRGBLxx for re-entrant batch
```

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 and above, 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); 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 other name that you choose for the Adabas client versioning link module) 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,        CICS defaults
      COFF=96,
      WTO=NO
VERCL VRL=814,             8.1.4 module, the default
      DEFAULT=YES,
      SUFFIX=81
VERCL VRL=823,             8.2.3 module
      SUFFIX=82
VERCL TRAN=AA82            Use 8.2.3
VERCL TYPE=END             The end.
```

- **Controls for the Default CICS Settings**

  This section describes the controls which can be used for the default CICS settings:

Control	Description
JOBTYPE=CICS	CICS for CICS configuration entries defaults.
COFF= TWA	Offset to be used for saving context information.
WTO=YES|NO	YES: Details of the configuration will appear on the console. NO: The configuration settings are silent.

- **Controls for the Versioned Link Module Entry Settings**

  This section describes the controls which can be used for the versioned link module entry settings:

Control	Description
VRL=814|823	814: This identifies an 8.1.4 link module. 823: This identifies an 8.2.3 link module.
DEFAULT=NO|YES	YES: This identifies the default link module. Only one default link module is allowed. Do not define any transaction elements for the default link module; it is used by any transaction not specified to use another versioned link module. NO: This link module is only used by the transactions listed immediately following it.
SUFFIX=*xx*	*xx*: This identifies the two character suffix to be used when calling the link module.

- **Controls for Specific Transaction Entry Settings**

  This section describes the controls which can be used for specific transaction entry settings:

Control	Description
TRAN=transaction code	This identifies the transactions that are to use the link module version identified prior to this transaction code list.

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

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

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

  For each suffix defined in VERC03, you will need modules called:

Module	Description
ADAENA*xx*	Adabas PLT TRUE enabler.
ADATRU*xx*	Adabas TRUE.
ADABAS*xx*	Adabas Link module.

  where *xx* is the suffix and the first 6 characters of the module name must be as shown.

  The Adabas 8.1 CICS link module uses fixed names for the TRUE and link module. Consequently, it is only possible to use one Adabas 8.1 link module. VERCIC01 must be linked as ADACICS and, assuming VERC03 defines suffix 81:

  - Copy the supplied ADACIC0 to ADAENA81.
  - Copy the supplied ADACICS to ADABAS81.
  - The TRUE must be called ADACICT.
  - Define PPT entries for ADAENA81, ADABAS81, ADACICS and VERC03.
  - Add ADAENA81 to the CICS PLT.

  Starting with Adabas 8.2, it is possible to use any name for the link module and it is also possible to use multiple Adabas 8.2 (or above) link modules.

  To use multiple link modules, you must have a set of modules for each suffix:

  - ACIOPT*xx*
  - ACINAM*xx*
  - CICGBL*xx*

- ADAENA*xx*
- ADABAS*xx*
- ADATRU*xx*
- And the suffixed modules for COR and related products.

Perform the following steps to create a set of modules with suffix 82:

1. Assemble and link ACIOPT82, using the ASMCOPT job supplied with Adabas. In the assembly, specify:

```
MACIOPT ENTRY=GLOBAL,MNTRUE=8
MACIOPT ENTRY=GROUP,GTNAME=CICGBL82
MACIOPT ENTRY=FINAL
END
```

2. Assemble and link ACINAM82, using the ASMCINS job supplied with Adabas. In the assembly, specify:

```
MACINS TRUENAME=ADATRU82,GTNAME=CICGBL82
END
```

3. Assemble and link the CICS globals table CICGBL82, using the ASMGBLS and LNKGCICS jobs supplied with Adabas. Specify the required parameters in the assembly, ensuring that the following parameters are set to the indicated values:

```
                                                GBLNAME=CICGBL82,   Default globals
module name            X
                                                ENTPT=ADABAS,           Adabas CICS
interface module       X
                                                TRUENM=ADATRU82,    Adabas CICS
TRUE name             X
```

   ENTPT must specify the name of the copied VERCIC01.

4. Link ADACIC0 with ACIOPT82 to create ADAENA82:

```
MODE AMODE(31),RMODE(ANY)
INCLUDE  ACILIB(ADACIC0)
INCLUDE  USERLIB(ACIOPT82)
ENTRY ADACIC0
NAME ADAENA82(R)
```

5. Link ADACICS with ACINAM82 to create ADABAS82:

```
MODE AMODE(31),RMODE(ANY)
REPLACE ACINAMES
INCLUDE  ACILIB(ADACICS)
INCLUDE  USERLIB(ACINAM82)
ENTRY ADACICS
NAME ADABAS82(R
```

6. Link or copy ADACICT to create ADATRU82:

```
MODE AMODE(31),RMODE(ANY)
   INCLUDE  ACILIB(ADACICT)
   ENTRY ADACICT
   NAME ADATRU82(R)
```

7. Link VERCIC01 to create ADABAS (the name specified as ENTPT in step 3), using job VERI085.

Now, during CICS initialisation you will see a set of ADAK messages for each TRUE, each of which will say "in use by ADABAS link routine ADABAS". This is because the versioning module is invoked by each PLT routine. The versioning module ensures that the correct TRUE and link module are initialized.

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.

Messages Issued by Client Versioning

The messages issued by client versioning (if you specify WTO=YES), have changed format slightly. The batch client versioning routines now only report the link module and options in use by this job, rather than all modules in the table, for example:

```
+VER101I Jobname  Link Module Globals  SVC    DBID
+VER101I -------- ----------- -------- ------ ------
+VER101I ADA81TST ADALNK81    BINDGBLS ASIS   ASIS
```

The CICS client versioning routine still lists the full table contents:

```
+VER101I  Criteria Link Module
+VER101I  -------- -----------
+VER101I  Default  ADABASDF
+VER101I  DEMO     ADABASDE
+VER101I  N426     ADABAS81
```

If client versioning detects an error (for example, the required link module is not available), it issues a message and rejects all Adabas commands with response 101, subcode 87.

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

Version 822: VERI080F

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.