Telelogic
System Architect Conversion Guide
Release 11.2

Before using this information, be sure to read the general information under Appendix, “Notices,” on page 11-1.

This edition applies to **Release 11.2**, **System Architect** and to all subsequent releases and modifications until otherwise indicated in new editions.

Table of Contents

1

Introduction to Conversion

Introduction

This chapter provides an overview of conversions necessary to bring work created in earlier System Architect versions to its latest version. This includes moving up to V10.4 and later and from previous versions (V10, V9, V8) moving to V8 from v6 and v7, and up to v6 or v7 from versions 4.0 and/or 3.1.

Overview of the Conversions

System Architect V11.2 automatically upgrades encyclopedias created in System Architect V10.7 to V11.2. System Architect V10.3 and later creates new stored procedures and tables (for history) and makes changes to existing stored procedures in all encyclopedias. Because of this, encyclopedias created in V10.1, V10.0, or V9 of the product must first be converted/upgraded for use in System Architect V10.3 format before they can be upgraded to V11.0. The instructions on how to do this follow in the next section.

If the version of System Architect that you are using is V9 or V10.0, and you have created custom macros that run off menu items, you will need to adjust them slightly for use in SA V10.1 and later. Details are provided in Chapter 3 of this manual.

Opening a V9 encyclopedia with V10 requires the System Architect administrator to grant access rights to a new GetFileSize stored procedure. Details are provided later in this chapter.

The conversion of V8 encyclopedias to V9, V10, or later is accomplished by following the procedures in Chapter 2. To convert an encyclopedia built with a previous version of System Architect to V9 (SQL Server underlying database), the encyclopedia must be converted to V8 first.

The conversions for previous versions of the product to V8 are performed by invoking the conversion routine, which is accessed in System Architect by selecting **Tools, Conversion** (with any encyclopedia open). If you are converting an encyclopedia created with a previous version of System Architect (versions 6 or 7), a GUID/UML conversion will run and create a new, converted encyclopedia in a new path (the old encyclopedia remains intact).

If you are converting an encyclopedia created with a System Architect 4.0 product, a data-modeling/long-names conversion will run first to convert the encyclopedia to System Architect status, and the GUID/UML conversion will run second to take the resulting, converted encyclopedia and convert it to System Architect V8 status.

Converting SA V11.0 or V11.1 Encyclopedias with the UML 2 Configuration Property Set to SA V11.2

Upgrading UML 2 Encyclopedias from SA V11.0 or V11.2

If you created encyclopedias with SA V11.0 or V11.1 and enabled the UML 2 property set, you need to run the 'SAEncyclopediaConversion.mac' macro to upgrade that encyclopedia for SA V 11.2, as follows:

1. If System Architect running, close and reopen it.
2. Click **File** and select **Open Encyclopedia**.
3. Select the encyclopedia you need to convert and click **OK**.
4. A dialog message informs you that the encyclopedia is configured for an earlier version of System Architect, and asks if you want to upgrade it. Click **No**.
5. Click on **Tools** and select **Macro Projects**.
6. In the **Macro Projects** dialog, click the **Add Macro** icon.
7. Navigate to the subfolder <INSTALLATION DIRECTORY>\SAUML2\1.1\etc\, select the **SAEncyclopediaConversion.mac** macro and click **Open**.
8. Click **OK** to close the **Macro Projects** dialog.
9. A Microsoft Visual Basic dialog informs you "macros may contain viruses." Click **Enable Macros**.
10. Click on **Tools** and select **Run Macro**.

11. Select **ExportForConversion11_0** or **ExportForConversion11_1** macro and Click **Run**.
12. Close and reopen System Architect.
13. Reopen the encyclopedia you are converting.
14. A dialog message informs you the encyclopedia is configured for an earlier version of System Architect, and asks if you want to upgrade it. Click **Yes**.
15. Click on **Tools** and select **Run Macro**.
16. Select **ImportForConversionFrom11_0** or **ExportForConversion11_1** and click **Run**.

Upgrading to SA V10.4 and later – Converting SA V10.3, V10.1, V10.0, and V9 Encyclopedias

Conversion Procedure to Upgrade Encyclopedias to SA V10.4 and later

Beginning with System Architect V10.3 new stored procedures and tables (for history) are created and changes are made to existing stored procedures in all encyclopedias (SQL Server 2000, MSDE, SQL Server 2005, SQL Express, Oracle 9i or Oracle 10g databases). To upgrade System Architect V10.4 and later encyclopedias to V11.0 you only need to open the V10.4 and later encyclopedias with System Architect V11.0.

However, encyclopedias created with V10.1, V10.0, or V9 of the product must be converted to V10.3 format before it can be updated automatically to V11.0. This is a two-stage process:

Note: SAEM cannot perform standard functions on an encyclopedia residing on a remote server.

Stage 1 – convert the data of the encyclopedia using SAEM, as follows:

1. Run System Architect Encyclopedia Manager (SAEM) and select **Start, Programs, Telelogic, Telelogic Lifecycle Solutions, System Architect, SAEM (SQL Server**) or **SAEM (Oracle**), depending on your server.
2. In SAEM, log in to the server containing the encyclopedia you wish to upgrade (select **Server, Connect**).
3. Choose the encyclopedia you wish to upgrade (select **Database, Select Database**).

4. In SAEM, select **Tools, Convert to 10.3**. This will convert the data to the format necessary to be used with System Architect 10.3.

Stage 2 – A person with a System Administrator's role, or the owner of the encyclopedia, must open the converted encyclopedia in System Architect V10.3 or later, as follows:

1. Find out whether the owner of an encyclopedia is using SAEM (for MSDE) or Microsoft's Enterprise Manager (for SQL Server). Have that person or a person with a System Administrator's role, open the converted encyclopedia in SA V10.3 or later.

2. System Architect will automatically try to add new stored procedures and tables to the existing encyclopedia when it is initially opened with SA V10.3 or later. As long as a person with a Systems Administrator role or the owner of the encyclopedia opens converted encyclopedia, the stored procedures will be properly created with an owner of 'dbo'.

Upgrading to SA V10.4 and later – Converting SA-DOORS Link Info to SA-DOORS V2

Conversion Procedure to Upgrade Encyclopedias to SA V10.4 and later

If you have been using the SA-DOORS interface to link model artifacts with DOORS objects, you will need to convert your encyclopedia for use with the SA-DOORS V2 interface, introduced with System Architect V10.4. To do so, perform these steps:

Note: SAEM cannot perform standard functions on an encyclopedia residing on a remote server.

1. We advise that you backup the encyclopedia used with the SA-DOORS V1 interface. You may backup the encyclopedia using SAEM (run SAEM, connect to your server, choose the encyclopedia, and select **Database, Backup**).
2. Run System Architect and open the encyclopedia that you wish to convert.
3. Run DOORS and open the database that the encyclopedia is associated with.
4. In DOORS, from the top menu select **SA, Upgrade/Recover System Architect**.

SA-DOORS Interface -- Updating DOORS-linked Encyclopedias for Use with SA V10.4 and later

If you have a SA V10.3 encyclopedia that you linked to DOORS, you will need to update that encyclopedia with SAEM to reestablish the link. To do so, perform these steps:

1. Backup the encyclopedia (see above).
2. Launch SAEM (SQL Server) or SAEM (Oracle), depending on the your server.

3. Click Tools menu, and select Update DOORS. The message pane at the bottom of the SAEM window confirms your results.
4. Open the encyclopedia with System Architect V10.4 and later. The links should now be updated to work with SA V10.4 and later.

Adjusting Macros that Run Off Menus in SA V10.1

Conversion Procedure for Macros Associated with Menu Items in SA V10.1

As of V10.1 System Architect has almost completely released the menus and toolbars from their previous fixed format. This means that users are now able to customize menus, and their customizations will remain. The only exceptions to this are the Draw menu and toolbar, where customization is only possible after the drawing tool items.

The goal of this change was to enable more tailored menu and toolbar configurations for specific user groups via SA Catalog Manager.

In general users should notice no significant changes to menus in this release of System Architect. All menu items and toolbars will remain where they were in previous versions.

If you have created custom macros that run off menu items, you will need to adjust them slightly for use in SA V10.1 and later. Details are provided in Chapter 3 of this manual.

Opening V9 Encyclopedias with System Architect V10

Procedure Required for Opening V9 Encyclopedias with V10

System Architect V10 requires a GetFileSize stored procedure in all encyclopedias (SQL Server 2000 or MSDE databases) which System Architect V9 did not require. System Architect will automatically try to add this stored procedure when the V9 encyclopedia is initially opened with V10. We recommend that the first person to open the V9 encyclopedia with V10 has a System Administrators role, or is the owner of the encyclopedia (you may find out who the owner of an encyclopedia is using SAEM (for MSDE) or Microsoft's Enterprise Manager (for SQL Server). When either of these types of users opens a V9 encyclopedia with V10, the GetFileSize stored procedure will be properly created with an owner of 'dbo'.

After the stored procedure is created (with an owner of 'dbo'), the System Architect administrator must also grant Execute privileges on the GetFileSize stored procedure to all users -- whether that is accomplished by granting it to their group, role, or individual account, depending on how they have set up security on SQL Server.

If a user other than the System Administrator or the owner opens the encyclopedia initially, executive rights to the stored procedure will be granted to that user only, and no other users will be able to access the encyclopedia. The reason is that if the GetFileSize stored procedure is created with an owner other than 'dbo', then System Architect will not recognize the stored procedure. When other users open the encyclopedia, they will get an error message indicating that the GetFileSize stored procedure wasn't found.

Conversion from V8 to V9, V10 or later – From a dBase to a SQL Server Encyclopedia

Procedure Required for Opening V8 Encyclopedias with V9, V10, or Later

System Architect versions V9, V10, and later offers a new underlying repository structure from V8 and earlier of the product. Starting with V9, encyclopedias are built as databases on SQL Server. You may create encyclopedias on either SQL Server 2000 or Microsoft Server Desktop Engine (MSDE). The procedure to convert existing V8 encyclopedias to V9, V10, and later is provided in Chapter 2.

Convert By Using Encyclopedia Merge Facility

In general, you create a new encyclopedia on SQL Server 2000 (or MSDE), and merge the older V8 dBase encyclopedia contents into it.

Changes to IMF Calls Affect VBA Macros

If you have created custom VBA macros for System Architect, and use certain IMF calls, you will need to make some minor modifications to those macros before proceeding. These changes are also detailed in Chapter 2.

Conversion to SA V8 – GUIDs and UML

Procedure Required for Opening SA 2001 Encyclopedias with V8

The conversion routine to bring encyclopedias created with System Architect v6 or v7 to V8 status has two broad functions to perform:

- assign a GUID to every object,
- convert any existing UML model elements to the new.

GUIDs

Every diagram, symbol and definition in the encyclopedia being converted will be given a Global Unique Identifier or GUID. Note that this applies to **all** objects, whether they are part of UML or otherwise. The GUID is a 64 character long value provided by Microsoft code. The value is supposed to be unique throughout the known universe and for all time.

UML

System Architect V8 features a major upgrade to UML functionality in the product. The UML upgrade features representational consistency of relationships drawn between UML classifiers, a new package structure (with many UML classifiers now 'keyed' to the package they are contained in), and new handling of implementation language types. Because of these and other features, a conversion of UML diagrams and definitions is necessary. Detailed information on the UML Conversion is provided in Chapter 5.

Conversion to SA/2001 – Data Modeling and Long File Names

In general there are two areas of change in System Architect that requires conversion for work done in version 4.0 of the product (or SA/BPR 3.2):

- Long Names (up to 80 characters) are now allowed for Symbols, Definitions, and Diagrams (the previous limit was 31 characters), and
- Data modeling has changed significantly in SA/2001.

When conversion is run against an encyclopedia created with a previous version of System Architect, a separate conversion routine is run for the each of the two topics above.

Long File Names

System Architect supports names up to a maximum of 80 characters long, compared to the old limit of 31 characters. The conversion routine converts all underlying fields so that they can hold the additional characters. Existing names of symbols will not change. You will, after the conversion process is complete, be able to rename old names to new names that are up to 80 characters long.

Data Modeling

Data modeling functionality was greatly enhanced in System Architect 2001 and later. Data models built with previous versions of System Architect (4.0 and 3.1, etc) must be run through a conversion routine to make them compatible with data modeling in System Architect. Details about the changes to data modeling functionality and the specifics of what the conversion program does are provided in chapters 7 and 8 of this manual. An overview is provided here.

Special Note for SA/Catalyst Users

In addition, changes have been made to some of the names of business modeling diagrams and definitions for the Catalyst methodology. This only affects users of SA/Catalyst who have made changes to the properties file (USRPROPS.TXT). Users who have modified properties for SA/Catalyst should read the conversion instructions in chapter 2, and refer to the name changes in chapter 9 of this manual to see if the name changes affect them.

Special Note for SA/BPR Users

In addition SA/BPR users should be aware that IDEF1X data models are not directly converted to the new data modeling format. They can be converted if they are first mapped to physical diagrams in the SA/BPR tool, converted to physical diagrams in SA/2001, then mapped to ER diagrams in SA/2001. In addition, Swimlanes from IDEF and Organizational Units from the Catalyst methodology are both present in SA/2001. SA/BPR users who have modified the properties of a Swimlane definition through USRPROPS.TXT should pay special attention to the conversion instructions for SA/BPR users in Chapter 2.

2

Procedures for Converting Existing Encyclopedias

Introduction

This chapter provides instructions on how to convert encyclopedias created with previous versions of products in the System Architect suite to encyclopedias for use with System Architect V8.

Overview

System Architect V10.3 creates new stored procedures and tables (for history) and makes changes to existing stored procedures in all encyclopedias. Because of this, encyclopedias created in V10.1, V10.0, or V9 of the product must be converted/upgraded for use in System Architect V10.3. The instructions on how to do this follow in the next section.

Starting with System Architect V9, a new underlying repository structure was introduced into the tool – encyclopedias are built as databases on SQL Server. You may create encyclopedias on either SQL Server 2000 or Microsoft Server Desktop Engine (MSDE). Encyclopedias used with System Architect V8 were based on a proprietary format based on dBase. They must be converted to SQL Server for use with System Architect V9, V10 and later.

Opening a V9 Encyclopedia with V10

System Architect V10 requires a GetFileSize stored procedure in all encyclopedias (SQL Server 2000 or MSDE databases) which System Architect V9 did not require. System Architect will automatically try to add this stored procedure when the V9 encyclopedia is initially opened with V10. After the stored procedure is created (with an owner of 'dbo'), the System Architect administrator must also grant Execute privileges on the GetFileSize stored procedure to all users. More details are provided in Chapter 1 of this manual.

Converting V8 to V9/V10 and Later

To convert an encyclopedia for use with V9, V10, and later versions of System Architect, it must be in V8 format first. Instructions to do this are provided in the next section.

Converting Previous Versions to V8

System Architect V8 has an automatic conversion utility that converts encyclopedias created in previous versions of the product. However, because previous versions of the product enabled the user to model inconsistent information in some areas, the conversion utility may come across decision points during conversion that need human user input. The conversion will point out modeling discrepancies -- it is then up to the user to fix these discrepancies.

Maintaining an Old Version of SA 2001 Is Recommended

The recommended place to fix a discrepancy is in the original encyclopedia. Because of this, it is recommended that a version of System Architect 2001 be maintained by the user if possible, on a separate machine. Conversion can then be run against the encyclopedia using System Architect V8, and if discrepancies are found, the encyclopedia can be moved to the machine with System Architect 2001, the discrepancies can be fixed, and the encyclopedia can then undergo a final conversion with System Architect V8.

Amalgam of Two Conversions -- Long File Names & Data Modeling; UML & GUIDs

The conversion routine accessed is accessed in System Architect by selecting **Tools, Conversion** (with any encyclopedia open). The conversion routine is actually an amalgam of two conversion routines – a data-modeling/long-names conversion to convert encyclopedias created with versions of System Architect prior to SA/2001 (ie, SA 4.0) to System Architect 2001 status, and a GUID/UML conversion to convert encyclopedias created with System Architect 2001, v6 or v7, to System Architect V8.

When you select **Tools, Conversion**, the conversion routine will ask you to specify the encyclopedia to be converted, examine the encyclopedia, and determine what type of conversion is necessary. If the encyclopedia was created with a version of System Architect prior to SA/2001, the data-modeling/long-names conversion will run first, to convert the encyclopedia to SA/2001 status. When that conversion finishes, the GUID/UML conversion will run to convert the encyclopedia to SA/2001 V8 status. If the encyclopedia was created with SA/2001 v6 or v7, only the GUID/UML conversion will run to bring it to V8 status.

The procedure for the data-modeling/long-names conversion routine is essentially the same no matter what encyclopedia you convert. However, there are nuances involved with each encyclopedia created by a different System Architect product.

This chapter is divided into four sections, the first detailing the procedure for converting SA/2001 v6 or v7 encyclopedias to V8 status, and three other sections, each one giving the specific procedure for performing a data-modeling/long-names conversion for a particular System Architect product. There is a section for converting encyclopedias created with System Architect 4.0 products (including those created with

SA/Object Architect 4.0, SA/Data Architect 4.0, and SA/Catalyst 4.0). There is a section for converting encyclopedias created with SA/BPR 3.2. There is also a section for converting encyclopedias created with SA/Data Architect 4.5.

Upgrading to SA V10.3 – Converting SA V10.1, V10.0, and V9 Encyclopedias

Conversion Procedure to Upgrade Encyclopedias to SA V10.3

System Architect V10.3 creates new stored procedures and tables (for history) and makes changes to existing stored procedures in all encyclopedias (SQL Server 2000 or MSDE databases). Because of this, encyclopedias created with V10.1, V10.0, or V9 of the product must be converted to V10.3 format. This is a two-stage process:

Stage 1 – convert the data of the encyclopedia using SAEM, as follows:

1. Run System Architect Encyclopedia Manager (SAEM) (select **Start, Programs, Telelogic, System Architect Suite, SAEM**).
2. In SAEM, log in to the server containing the encyclopedia you wish to upgrade (select **Server, Connect**).
3. Choose the encyclopedia you wish to upgrade (select **Database, Select Database**).
4. In SAEM, select **Tools, Convert to 10.3 Format**. This will convert the data to the format necessary to be used with System Architect 10.3.

Stage 2 – A person with a System Administrator’s role, or the owner of the encyclopedia, must open the converted encyclopedia in System Architect V10.3, as follows:

1. Find out who the owner of an encyclopedia is using SAEM (for MSDE) or Microsoft's Enterprise Manager (for SQL Server). Have that person or a person with a System Administrator’s role, open the converted encyclopedia in SA V10.3.

2. System Architect will automatically try to add new stored procedures and tables to the existing encyclopedia when it is initially opened with SA V10.3. As long as a person with a Systems Administrator role or the owner of the encyclopedia opens converted encyclopedia, the stored procedures will be properly created with an owner of 'dbo'.

Upgrading to SA V10.3 – Converting SA-DOORS Link Info to SA-DOORS V2

Conversion Procedure to Upgrade Encyclopedias to SA V10.3

If you have been using the SA-DOORS interface to link model artifacts with DOORS objects, you will need to convert your encyclopedia for use with the SA-DOORS V2 interface, introduced with System Architect V10.3. To do so, perform these steps:

1. We advise that you backup the encyclopedia used with the SA-DOORS V1 interface. You may backup the encyclopedia using SAEM (run SAEM, connect to your server, choose the encyclopedia, and select **Database, Backup**).

2. Run System Architect and open the encyclopedia that you wish to convert.

3. Run DOORS and open the database that the encyclopedia is associated with.

4. In DOORS, from the top menu select **SA, Upgrade/Recover System Architect**.

Opening V9 Encyclopedias with System Architect V10

Procedure Required for Opening V9 Encyclopedias with V10

System Architect V10 requires a GetFileSize stored procedure in all encyclopedias (SQL Server 2000 or MSDE databases) which System Architect V9 did not require. System Architect will automatically try to add this stored procedure when the V9 encyclopedia is initially opened with V10. We recommend that the first person to open the V9 encyclopedia with V10 has a System Administrators role, or is the owner of the encyclopedia (you may find out who the owner of an encyclopedia is using SAEM (for MSDE) or Microsoft's Enterprise Manager (for SQL Server). When either of these types of users opens a V9 encyclopedia with V10, the GetFileSize stored procedure will be properly created with an owner of 'dbo'.

After the stored procedure is created (with an owner of 'dbo'), the System Architect administrator must also grant Execute privileges on the GetFileSize stored procedure to all users -- whether that is accomplished by granting it to their group, role, or individual account, depending on how they have set up security on SQL Server.

If a user other than the System Administrator or the owner opens the encyclopedia initially, executive rights to the stored procedure will be granted to that user only, and no other users will be able to access the encyclopedia. The reason is that if the GetFileSize stored procedure is created with an owner other than 'dbo', then System Architect will not recognize the stored procedure. When other users open the encyclopedia, they will get an error message indicating that the GetFileSize stored procedure wasn't found.

Converting V8 to V9/V10

Encyclopedias created with versions of System Architect prior to V9 were based on a proprietary format based on dBase. That format was held in a folder with various files inside of it. Beginning with V9, encyclopedias are now stored in databases that can reside on SQL Server, MSDE or Oracle.

Preparatory Step if You Have Custom Graphics

The conversion process is as follows:

1. If you have custom .bmp and .wmf files associated with an encyclopedia, you must perform two preparatory steps:

    a. Use Windows Explorer or equivalent, to copy the custom .bmp and .wmf files associated with the V8 encyclopedia to be converted to the images folder of the main System Architect software directory (<C>:\Program Files\Telelogic\System Architect Suite\System Architect\Images).

    b. Copy the usrprops.txt file of the V8 encyclopedia that contains custom images, and paste it into the main System Architect directory (<C>:\Program Files\Telelogic\System Architect Suite\System Architect), replacing the usrprops.txt file that is there.

Convert Encyclopedias By Using Merge Facility

1. In System Architect, create a new encyclopedia in SQL Server or MSDE. (This new encyclopedia will now already contain all necessary custom pictures and usrprops.txt.). Procedures for creating a new encyclopedia in SQL Server or MSDE are provided in the on-line help or in the Installation Guide.
2. Use System Architect's Encyclopedia Merge utility to merge all information from the V8 (dBase) encyclopedia into the new V9/10 encyclopedia, as follows:

    a. Open the encyclopedia you created in the above step.

b. Click **Tools**, and select **Merge**.

c. In the **Merge/Extract** dialog, click the **Browse** button. The **Open Encyclopedia** dialog will open.

d. Click on the **Existing** icon.

e. Click the ‘**…**’ ellipses button to create a new connection. The **Connection Manager** dialog is displayed.

f. Click on the **New** icon. A new entry for your connection is created in the **Available connections** field.

g. Enter a name in the **Connection Name** column.

h. Click on the **Server Type** column and select **dBase**.

i. Click once on the **Server Name** column then click the ‘**…**’ ellipses button that appears in that column.

j. Browse to the folder with your dBase encyclopedia, select it, and click **OK**.

k. Click **OK** to close the **Connection Manager**.

l. Select your dBase encyclopedia from the **Open Encyclopedia** dialog and click **OK**.

m. Select your merge options in the **Merge/Extract** dialog, and click **OK**. A reports window displays the results of the merge/extract operation.

Modification of VBA Macros and SA/Word Macros

Users who have built their own VBA macros for use with System Architect, or their own SA/Word VBA macros (which use the sawd2001.dll file), need to be aware of certain changes that have been made to System Architect’s IMF calls, and may need to modify their code appropriately. Please refer to Chapter 3 for details.

Procedures to Convert SA 2001 Encyclopedias to SA V8

What Is Being Converted

Encyclopedias created with the System Architect 2001, versions 6 or 7, must undergo both a GUID conversion and a UML conversion. This conversion must take place for all encyclopedias, even if no UML diagrams or definitions have been created.

GUID Conversion

The conversion routine will perform the GUID conversion first, creating a new encyclopedia in a directory specified by the user. The original encyclopedia is left in tact.

UML Conversion

The UML conversion is run on the intermediate, new encyclopedia created by the GUID conversion. The UML conversion routine will create a new encyclopedia in a directory within the directory containing the GUID conversion.

Overall Procedure

Maintaining a Version of SA 2001

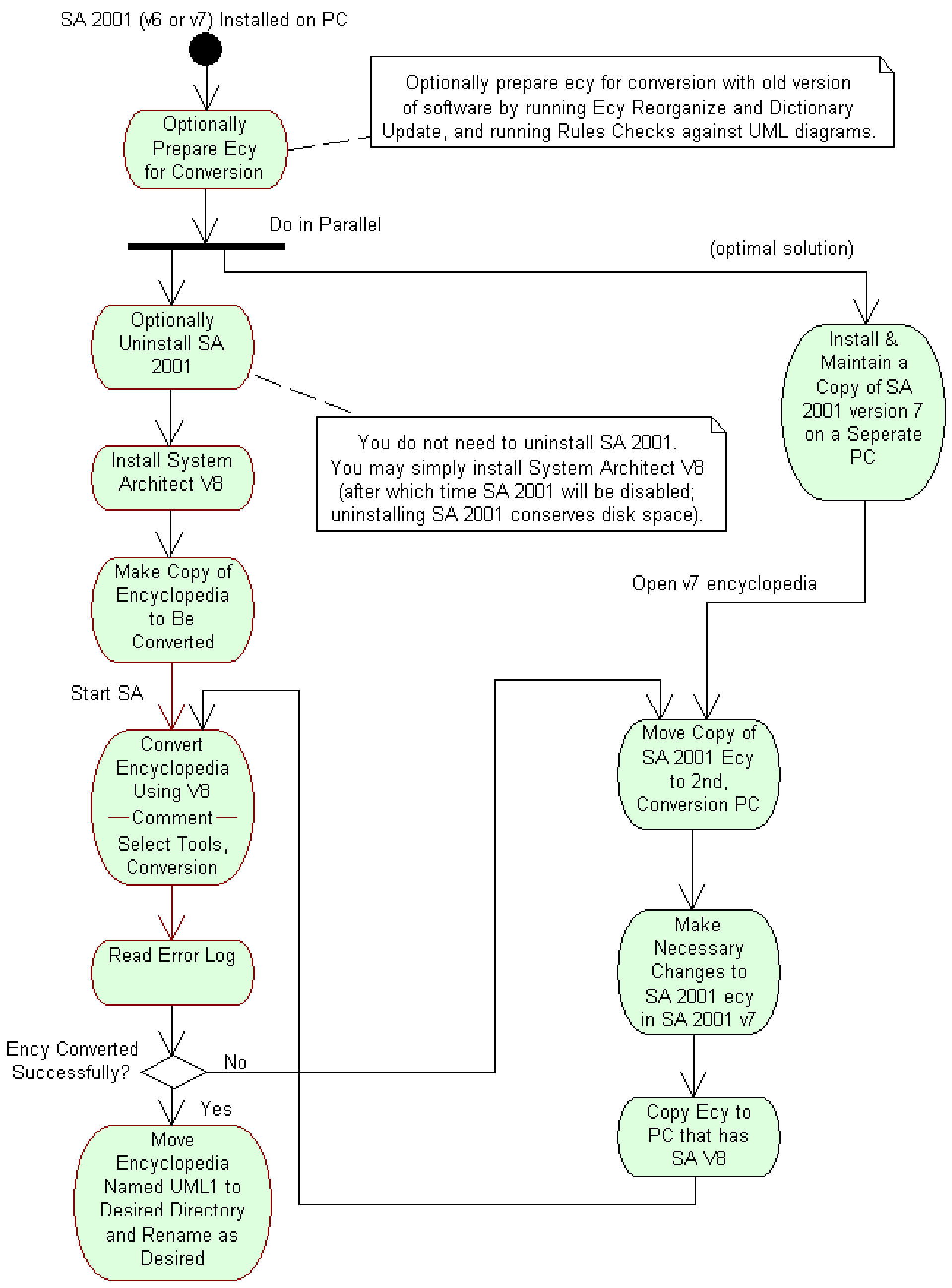
As mentioned in the Overview section earlier, because the conversion can run into modeling discrepancies that only the user can decide how to rectify, it is recommended that a version of System Architect 2001 be maintained by the user if possible, on a separate machine. Conversion can then be run against the encyclopedia using System Architect V8, and if discrepancies are found, the encyclopedia can be moved to the machine with System Architect 2001, the discrepancies can be fixed, and the encyclopedia can then undergo a final conversion with System Architect V8.

The overall procedure for conversion is as follows (it is also graphically depicted below).

1. With SA 2001 still installed, prepare the encyclopedia for conversion by running Encyclopedia Reorganize and Dictionary Update, and running Rules Checks agains UML Diagrams (see next section for details).
2. If you can, install SA 2001 on a separate machine that you can use to fix discrepancies found in the encyclopedia during conversion.

3. Install System Architect V8 on your normal machine (you may optionally uninstall SA 2001. If you don’t uninstall it, it will simply be disabled when you install System Architect V8.
4. Make a copy of the encyclopedia to be converted. Convert the copy. (Note: making a copy is a redundantly conservative step -- System Architect makes a copy of the encyclopedia during conversion (and names it UML1). SA does not make any changes to the existing encyclopedia.
5. Run the conversion routine (see section later in this chapter named Running the V8 Conversion Routine).
6. Read the Error Log. If there are errors in the log, and the encyclopedia was not successfully converted, reopen the encyclopedia in SA 2001, refer to Chapter 3 of this manual to correct errors.
7. Rerun the conversion routine on the corrected encyclopedia.
8. If you have created any custom-built SA/Word Reports, VBA macros, or Matrices in previous versions of the product, you must convert them as well, following instructions at the end of Chapter 3.

Procedures to Convert SA 2001 Encyclopedias to SA V8


SA 2001 (v6 or v7) Installed on PC
Optionally Prepare Ecy for Conversion
Optionally prepare ecy for conversion with old version of software by running Ecy Reorganize and Dictionary Update, and running Rules Checks against UML diagrams.
Do in Parallel
(optimal solution)
Optionally Uninstall SA 2001
Install & Maintain a Copy of SA 2001 version 7 on a Seperate PC
You do not need to uninstall SA 2001. You may simply install System Architect V8 (after which time SA 2001 will be disabled; uninstalling SA 2001 conserves disk space).
Install System Architect V8
Make Copy of Encyclopedia to Be Converted
Open v7 encyclopedia
Start SA
Convert Encyclopedia Using V8
Comment
Select Tools, Conversion
Move Copy of SA 2001 Ecy to 2nd, Conversion PC
Read Error Log
Make Necessary Changes to SA 2001 ecy in SA 2001 v7
Ency Converted Successfully?
No
Yes
Copy Ecy to PC that has SA V8
Move Encyclopedia Named UML1 to Desired Directory and Rename as Desired

Preparing the SA/2001 v6 or v7 Encyclopedia for V8 Conversion

Before commencing the conversion, we strongly recommend that you put the encyclopedia through a few preparatory steps using the old version of the software.

1. Select **Tools, Reorganize** to restructure the encyclopedia database.

2. Select **Tools, Dictionary Update** to update and correct the dictionary.

3. Open each UML model diagram in your encyclopedia and select **Reports, Rules Check** to check diagrams and symbols and their related dictionary definitions for defects such as omissions, inconsistencies, and violations of Methodology Rules.

Most of the utilities recommended above will produce reports that will warn you of errors encountered and whether or not System Architect took actions to fix them. Please read all of these reports and try to fix as many errors as you can.

Running the V8 Conversion Routine

Selecting the Encyclopedia

Select the encyclopedia to be converted as follows:

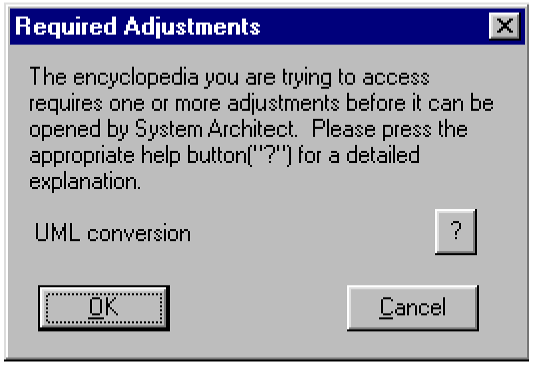
1. Launch System Architect V8. You may open any current encyclopedia.
2. From the **Tools** menu, select **Conversion**.
3. In the **Existing** tab, browse to the directory of the encyclopedia that you wish to convert. Click **OK** to begin the conversion process.

If Necessary, Data Modeling and Long Names Conversion Will Run First

System Architect examines the encyclopedia. If it is one created with a version of the software prior to System Architect, the data-modeling/long-names conversion will kick in first, and you will be presented with a **Required Adjustments** dialog that tells you that these two conversions must be run. Refer to the following sections of this chapter for details on the data-modeling/long-names conversion.

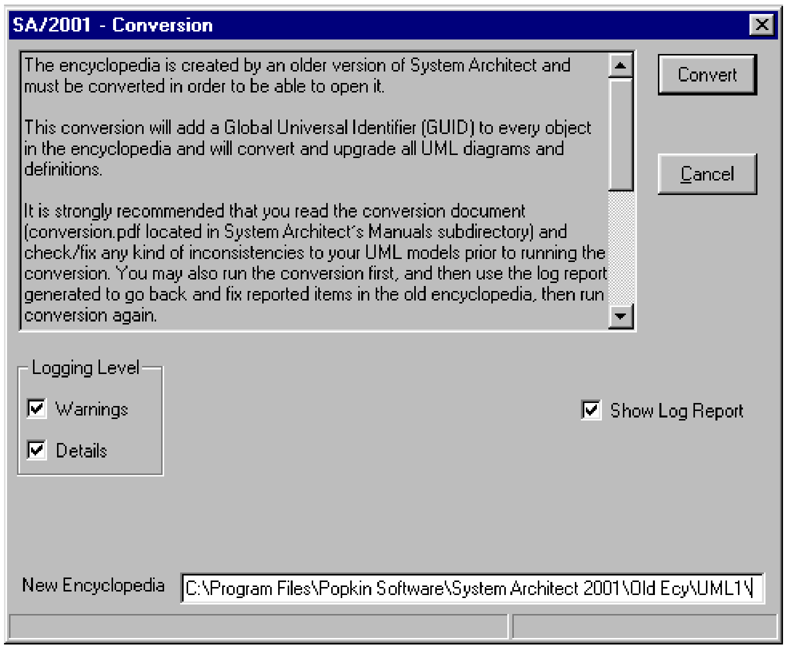
UML and GUID Conversion

If the encyclopedia is one created with System Architect v6 or v7, the only conversion that will run is the GUID/UML conversion. The **Required Adjustments** dialog specifies that this is the conversion that must be run.



The encyclopedia you are trying to access requires one or more adjustments before it can be opened by System Architect. Please press the appropriate help button("?") for a detailed explanation.

4. In the **Required Adjustments** dialog, select **OK**. The main conversion dialog will open.



The encyclopedia is created by an older version of System Architect and must be converted in order to be able to open it.

This conversion will add a Global Universal Identifier (GUID) to every object in the encyclopedia and will convert and upgrade all UML diagrams and definitions.

It is strongly recommended that you read the conversion document (conversion.pdf located in System Architect's Manuals subdirectory) and check/fix any kind of inconsistencies to your UML models prior to running the conversion. You may also run the conversion first, and then use the log report generated to go back and fix reported items in the old encyclopedia, then run conversion again.

Logging Level
✓ Warnings
✓ Details

✓ Show Log Report

New Encyclopedia C:\Program Files\Popkin Software\System Architect 2001\Old Ecy\UML1\

5. If you would like to see all details of the conversion, toggle on both **Warnings** and **Details** for the **Logging Level** property.

6. In the **New Encyclopedia** path, specify the directory of the new encyclopedia. The default is a directory within the current encyclopedia path called **UML1**.

7. Click on the **Convert** button. The **GUID conversion** will run first, and right after it, the **UML conversion** will run. No user interaction is required. A new, converted encyclopedia will be created in a directory specified. The conversion will stop if any significant errors are encountered (E3021 through E3027). See chapter 3 for details of error messages and warnings.

Procedures to Convert Encyclopedias Created with SA 4.0 to SA/2001

What is Being Converted

Encyclopedias created with the System Architect 4.0 product line (including SA/Object Architect 4.0, SA/Data Architect 4.0, and SA/Catalyst 4.0) must undergo both a long-name conversion and a data modeling conversion.

Preparing the Encyclopedia

Users should prepare their encyclopedia for conversion first, by running some specified utilities native to the 4.0 product, to make sure the encyclopedia is indexed and organized well, and that the data models are close to being 'methodologically correct.'

Long-Name Conversion

The conversion routine will perform the long-name conversion first, creating a new encyclopedia in a directory specified by the user. The original encyclopedia is left in tact.

Data Modeling Conversion

The data modeling conversion is run on the intermediate, new encyclopedia created during the long-name conversion. The conversion routine will create a new encyclopedia in a directory within the directory containing the long-name conversion encyclopedia, or wherever you specify. The long-name conversion encyclopedia is left in tact. So, at the end of the conversion process, you have three encyclopedias. The one that has undergone both the long-name conversion and the data modeling conversion is the final, converted encyclopedia.

SA/Catalyst Name Changes Affecting USRPROPS.TXT

In addition, SA/Catalyst users need to check Appendix A for a listing of name changes that have been made to certain business modeling diagrams, symbols, and definitions. These name changes will only affect an existing user if they have made changes to these items through USRPROPS.TXT. The user will need to adjust their USRPROPS.TXT manually to reflect the name changes.

Preparing the SA 4.0 Encyclopedia for Conversion

Before commencing the conversion, we strongly recommend that you put the encyclopedia through a few preparatory steps using the old version of the software.

1. Select **Tools, Reorganize** to restructure the encyclopedia database.

2. Select **Tools, Dictionary Update** to update and correct the dictionary.

3. Open each data model (Entity Relation and Physical) diagram in your encyclopedia and select **Reports, Rules Check** to check diagrams and symbols and their related dictionary definitions for defects such as omissions, inconsistencies, and violations of Methodology Rules. You may also wish to run the Rules Checks for all other diagrams in your encyclopedia, but it is especially important to get a feel for how well your data models are methodologically correct.

4. Select **Reports, Expression Check** to invoke a complete data expression syntax check and validation, and update all associated cross referencing. Online error indications are provided.

5. Open each data model diagram and select **Dictionary, Propagate FKs** to propagate foreign keys. The routine will place the initials "FK" next to foreign keys, and notate the foreign key relationship within the definition.

Most of the utilities recommended above will produce reports that will warn you of errors encountered and whether or not System Architect took actions to fix them. Please read all of these reports and try to fix as many errors as you can.

For example, if the **Rules Check** report tells you that you have attributes that do not have underlying data elements, you may want to manually add data elements to these attributes if you do not want the conversion program to do it automatically for you (matching the name of the attribute with a data element).

Running the Conversion Routine

Selecting the Encyclopedia

Select the encyclopedia to be converted as follows:

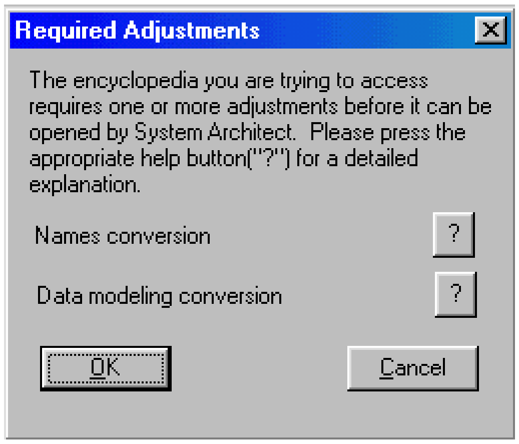
1. Launch System Architect. You may open any current encyclopedia.
2. From the **Tools** menu, select **Conversion**.
3. In the **Existing** tab, browse to the directory of the encyclopedia that you wish to convert. Click **OK** to begin the conversion process.

Required Conversion Varies by Encyclopedia

System Architect examines the encyclopedia and tells you the types of conversions that are necessary in the **Required Adjustments** dialog. Depending on the type of encyclopedia you are trying to convert, up to three different conversions may be necessary:

4. Network conversion is needed when opening a merge encyclopedia with a network version of System Architect. Please note that you can not open a network encyclopedia with a merge version of SA/2001.
5. Long name conversion is needed for all encyclopedias created in versions prior to SA/2001.
6. Data modeling conversion is needed for encyclopedias of the System Architect 4.0 family (including SA/Object Architect 4.0, SA/Data Architect 4.0, and SA/Catalyst 4.0), and SA/BPR 3.2. It is ***not*** necessary for SA/Data Architect 4.5.

Long Names Conversion


Required Adjustments

The encyclopedia you are trying to access requires one or more adjustments before it can be opened by System Architect. Please press the appropriate help button("?") for a detailed explanation.

Names conversion ?

Data modeling conversion ?

OK Cancel

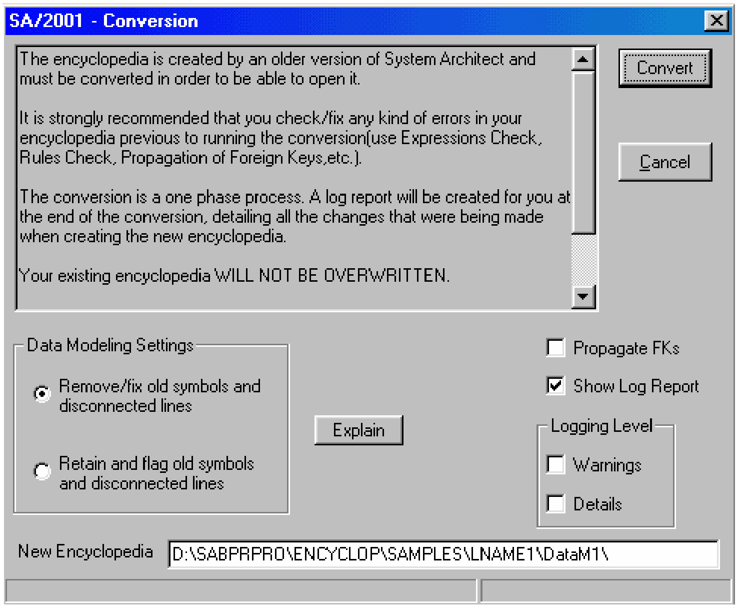
4. In the **Required Adjustments** dialog, select **OK**. The long names conversion will run first, creating a new encyclopedia in a directory within the original encyclopedia. The new encyclopedia is named LName1 by default (if LName1 already exists, then LName2 is created, and so on). You cannot specify a different name for the long-name conversion encyclopedia.

After the new, long-name conversion encyclopedia is created, the data modeling **Conversion** dialog will appear, presenting you with a number of choices for the data modeling conversion program.

What to Do After the Conversion

When the conversion is complete, System Architect should have the newly converted encyclopedia open. The encyclopedia is ready for further work.

Data Modeling Conversion


SA/2001 - Conversion

The encyclopedia is created by an older version of System Architect and must be converted in order to be able to open it.

It is strongly recommended that you check/fix any kind of errors in your encyclopedia previous to running the conversion(use Expressions Check, Rules Check, Propagation of Foreign Keys,etc.).

The conversion is a one phase process. A log report will be created for you at the end of the conversion, detailing all the changes that were being made when creating the new encyclopedia.

Your existing encyclopedia WILL NOT BE OVERWRITTEN.

Data Modeling Settings: [✓ Remove/fix old symbols and disconnected lines] / Retain and flag old symbols and disconnected lines

Propagate FKs: ☐

Show Log Report: ☑

Logging Level: Warnings ☐, Details ☐

New Encyclopedia: D:\SABPRPRO\ENCYCLOP\SAMPLES\LNAME1\DataM1\

5. Make your selections in the **Conversion** dialog, based on information presented below.

Data Modeling Settings:

a) Remove/fix old symbols and disconnected lines – Some symbols, such as indexes and access paths, are no longer used in SA/2001. In addition, SA/2001 does not allow disconnected relation or constraint lines. Please refer to Chapters 6 and 7 of this manual for details.

This option will cause the conversion program to automatically delete the obsolete symbols and unconnected relation lines.

b) Retain and flag old symbols and disconnected lines – This option will retain and flag symbols that are no longer used in SA/2001, and retain and flag unconnected relation or constraint lines.

New Encyclopedia:

Specify the name and path of the new encyclopedia to be created by the data modeling conversion. The data modeling conversion is run on the intermediate, new encyclopedia created during the long-name conversion. The default encyclopedia name is DataM1

Show Log Report:

With this selection toggled on, the conversion program creates a detailed log of all the actions performed, assumptions made, and errors discovered during the conversion process.

Logging Level can be chosen so that just Warnings are added to the log, or all Details are added to the log, or both.

Propagate FK's:

With this selection toggled on, SA/2001 automatically updates all foreign keys during the conversion process.

What to Do After the Conversion

When the conversion is complete, System Architect should have the newly converted encyclopedia open. The encyclopedia is ready for further work. If you are an SA/Catalyst user who has modified the properties of any of the Catalyst diagram, symbol, or definition types in USRPROPS.TXT, you should take a look at the next section.

Adjusting USRPROPS.TXT – SA/Catalyst Users Only

SA/Catalyst users need to check Appendix A to for a listing of name changes that have been made to certain business modeling diagrams, symbols, and definitions. These name changes do not affect existing work – the applicable diagrams, symbols, and definitions simply appear in SA/2001 under the new names. The conversion program does not actively change anything.

For example, the **Process Flow** diagram has been renamed to the **Process Chart** diagram, and **Catalyst Elem. Business Process** has been renamed to **Elementary Business Process**. If you had an encyclopedia with Process Flow diagrams that contained Catalyst Elem. Business Processes, when you open the encyclopedia in SA/2001, you will see a Process Chart diagram that contains Elementary Business Processes. All underlying definitions, and diagram layout, remain exactly as they were.

These changes will affect you if you have made changes to the properties of the changed items through USRPROPS.TXT. You will need to adjust your USRPROPS.TXT manually to reflect the name changes. For example, if you added a new property to a Catalyst Elem. Business Process, you must open your USRPROPS.TXT and rename Catalyst Elem. Business Process to Elementary Business Process for the USRPROPS.TXT change to be recognized.

Procedures to Convert Encyclopedias Created with SA/BPR 3.2 to SA/2001

What is Being Converted

Encyclopedias created with SA/BPR 3.2 must undergo a long-name conversion, and a data modeling conversion for physical data models only.

The current version of SA/2001 does not actively convert IDEF1X data model diagrams to the new format for data modeling. If you want to convert your IDEF1X data models to the new format, you should make sure that each IDEF1X diagram is represented by a physical diagram in the existing SA/BPR encyclopedia. The physical diagram in SA/BPR will be converted to a physical diagram in SA/2001, which you can then map to a logical data model using System Architect's physical-to-logical conversion.

Also during conversion, SA/BPR's property file, BPRPROPS.CFG, is converted to an applicable SAPROPS.CFG file for SA/2001. The USRPROPS.TXT file is incorporated into the new, converted encyclopedia unchanged. However, users may need to make a manual change to it based on a name change to the IDEF3 Process Flow diagram (see below).

Preparing the Encyclopedia

Users should prepare their encyclopedia for conversion first, by running some specified utilities native to the SA/BPR 3.2 product, to make sure the encyclopedia is well indexed and organized.

Long-Name Conversion

The conversion routine will perform the long-name conversion first, creating a new encyclopedia in a subdirectory of the old one (with a default name of LName1). The original encyclopedia is left in tact.

Data Modeling Conversion

The data modeling conversion is run against physical data models of the intermediate, long-file-name encyclopedia created during the long-name conversion. The data modeling conversion routine will create a new encyclopedia in a directory that you specify. The long-name conversion encyclopedia is left in tact. So, at the end of the conversion process, you have three encyclopedias – the original one, the one that has undergone the long-name conversion only, and the final, converted one that has undergone both the long-name conversion and the data modeling conversion.

SA/BPR Swimlane Change Affecting USRPROPS.TXT

In addition, SA/BPR users may need to modify their USRPROPS.TXT file if they have modified properties of the Swimlane definition. The reason for this is that in SA/BPR the horizontal or vertical lines used to model business units in IDEF3 Process Flow diagrams, IDEF0 diagrams, and Flow Charts were called Swimlanes. In SA/Catalyst, the same item is defined as an Organizational Unit.

In SA/2001, both items may be used. If, in the encyclopedia's configuration dialog (**Tools, Customize Method Support**), you have **IDEF3** and **IDEF0** selected, and ***do not*** have **Enterprise** modeling selected, a swimlane is defined as a Swimlane. However, if you have **Enterprise** modeling selected (with or without IDEF turned on) Swimlanes are globally renamed to Organizational Units in the encyclopedia.

The end result is that if you plan to model in System Architect using the business models of **Enterprise** modeling (with or without IDEF), and you have a USRPROPS.TXT that specifically modifies the Swimlane definition, then you must alter that USRPROPS.TXT so that **Swimlane** is replaced by **Organizational Unit**.

Preparing the SA/BPR 3.2 Encyclopedia for Conversion

Before commencing the conversion, we strongly recommend that you put the encyclopedia through a few preparatory steps using the old version of the software.

1. If you want to convert existing IDEF1X data models to the new SA/2001 data modeling format, make sure that every logical IDEF1X diagram has a corresponding physical diagram in the encyclopedia to be converted.

2. Select **Tools, Reorganize** to restructure the encyclopedia database.

3. Select **Tools, Dictionary Update** to update and correct the dictionary.

4. Open each data model (Entity Relation and Physical) diagram in your encyclopedia and select **Reports, Rules Check** to check diagrams and symbols and their related dictionary definitions for defects such as omissions, inconsistencies, and violations of Methodology Rules. You may also wish to run the **Rules Checks** for all other diagrams in your encyclopedia, but it is especially important to get a feel for how well your data models are methodologically correct.

5. Select **Reports, Expression Check** to invoke a complete data expression syntax check and validation, and update all associated cross referencing. Online error indications are provided.

6. Open each data model diagram and select **Dictionary, Propagate FKs** to propagate foreign keys. The routine will place the initials "FK" next to

foreign keys, and notate the foreign key relationship within the definition.

Most of the utilities recommended above will produce reports that will warn you of errors encountered and whether or not System Architect took actions to fix them. Please read all of these reports and try to fix as many errors as you can.

Running the Conversion Routine

Selecting the Encyclopedia

Select the encyclopedia to be converted as follows:

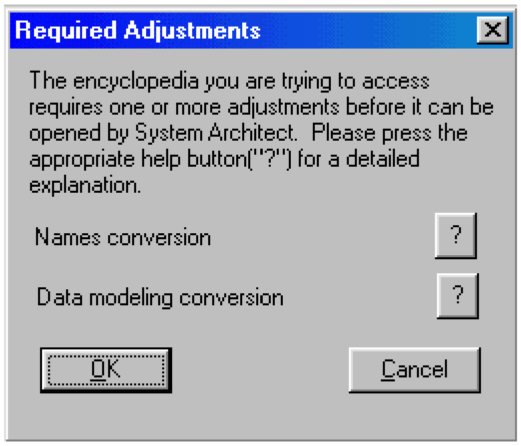
1. Launch System Architect. You may open any current encyclopedia.
2. From the **Tools** menu, select **Conversion**.
3. In the **Existing** tab, browse to the directory of the encyclopedia that you wish to convert. Click **OK** to begin the conversion process.

Required Conversion Varies by Encyclopedia

System Architect examines the encyclopedia and tells you the types of conversions that are necessary in the **Required Adjustments** dialog. Depending on the type of encyclopedia you are trying to convert, up to three different conversions may be necessary:

1. Network conversion is needed when opening a merge encyclopedia with a network version of System Architect. Please note that you can not open a network encyclopedia with a merge version of SA/2001.
2. Long name conversion is needed for all SA/BPR encyclopedias.
3. Data modeling conversion is needed for all SA/BPR encyclopedias.

Long Names Conversion

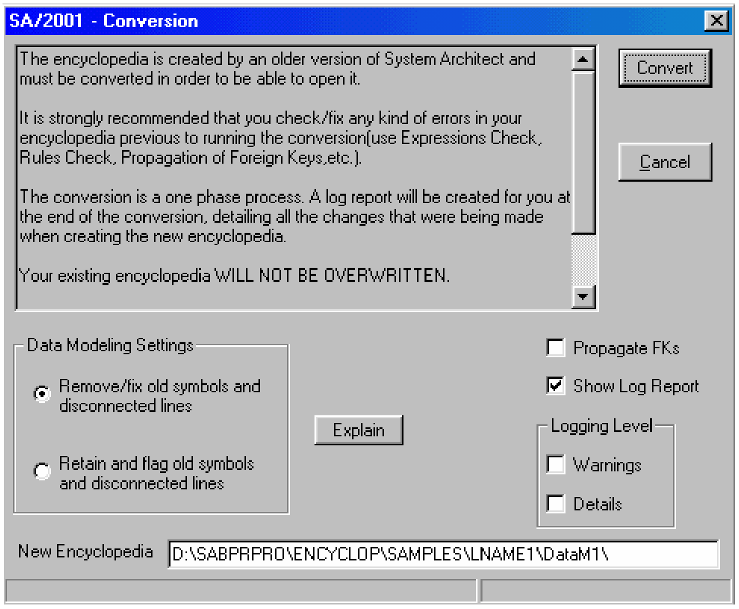


The encyclopedia you are trying to access requires one or more adjustments before it can be opened by System Architect. Please press the appropriate help button("?") for a detailed explanation.

4. In the **Required Adjustments** dialog, select **OK**. The long names conversion will run first, creating a new encyclopedia in a directory within the original encyclopedia. The new encyclopedia is named LName1 by default (if LName1 already exists, then LName2 is created, and so on). You cannot specify a different name for the long-name conversion encyclopedia.

After the new, long-name conversion encyclopedia is created, the data modeling **Conversion** dialog will appear, presenting you with a number of choices for the data modeling conversion program.

Data Modeling Conversion


SA/2001 - Conversion

The encyclopedia is created by an older version of System Architect and must be converted in order to be able to open it.

It is strongly recommended that you check/fix any kind of errors in your encyclopedia previous to running the conversion(use Expressions Check, Rules Check, Propagation of Foreign Keys,etc.).

The conversion is a one phase process. A log report will be created for you at the end of the conversion, detailing all the changes that were being made when creating the new encyclopedia.

Your existing encyclopedia WILL NOT BE OVERWRITTEN.

Data Modeling Settings
- [✓ Remove/fix old symbols and disconnected lines]
- Retain and flag old symbols and disconnected lines

- Propagate FKs
- [✓ Show Log Report]

Logging Level
- Warnings
- Details

New Encyclopedia: D:\SABPRPRO\ENCYCLOP\SAMPLES\LNAME1\DataM1\

5. Make your selections in the **Conversion** dialog, based on information presented below.

Data Modeling Settings:

a) Remove/fix old symbols and disconnected lines – Some symbols, such as indexes and access paths, are no longer used in SA/2001. In addition, SA/2001 does not allow disconnected relation or constraint lines. Please refer to Chapters 6 and 7 of this manual for details.

This option will cause the conversion program to automatically delete the obsolete symbols and unconnected relation lines.

b) Retain and flag old symbols and disconnected lines – This option will retain and flag symbols that are no longer used in SA/2001, and retain and flag unconnected relation or constraint lines.

New Encyclopedia:

Specify the name and path of the new encyclopedia to be created by the data modeling conversion. The data modeling conversion is run on the intermediate, new encyclopedia created during the long-name conversion. The default encyclopedia name is DataM1

Show Log Report:

With this selection toggled on, the conversion program creates a detailed log of all the actions performed, assumptions made, and errors discovered during the conversion process.

Logging Level can be chosen so that just Warnings are added to the log, or all Details are added to the log, or both.

Propagate FK's:

With this selection toggled on, SA/2001 automatically updates all foreign keys during the conversion process.

What to Do After the Conversion

When the conversion is complete, System Architect should have the newly converted encyclopedia open. The encyclopedia is ready for further work. If you are an SA/BPR user who has modified the properties of any of the Swimlane definition in USRPROPS.TXT, you should take a look at the next section.

If you want to create logical data models from your physical data models (which correspond to your IDEF1X data models), you may do so. Please refer to the on-line help for instructions.

Adjusting USRPROPS.TXT

As mentioned previously, in SA/BPR the horizontal or vertical lines used to model business units in IDEF3 Process Flow diagrams, IDEF0 diagrams, and Flow Charts were called Swimlanes. In SA/Catalyst, the same item is defined as an Organizational Unit.

In SA/2001, both items may be used. If, in the encyclopedia's configuration dialog (**Tools, Customize Method Support**), you have **IDEF3** and **IDEF0** selected, and ***do not*** have **Enterprise** modeling selected, a swimlane is defined as a Swimlane. However, if you have **Enterprise** modeling selected (with or without IDEF turned on) Swimlanes are globally renamed to Organizational Units in the encyclopedia.

If you have made modifications to the Swimlane definition in USRPROPS.TXT, and you will be using System Architect's Business Enterprise modeling in addition to IDEF, then you will need to make modifications to your USRPROPS.TXT file. You should rename **Swimlane** to **Organizational Unit**.

Procedures to Convert Encyclopedias Created with SA/Data Architect 4.5 to SA/2001

What is Being Converted

SA/Data Architect 4.5 already incorporates the new data modeling behavior and functionality present in SA/2001. Because of this, encyclopedias created with SA/Data Architect 4.5 do not need to go through the data modeling conversion routine. However, they must go through the long-name conversion to enable names of up to 80 characters to be created for diagrams, symbols, and definitions.

Preparing the Encyclopedia

Users should prepare their encyclopedia for conversion first, by running some specified utilities native to the SA/Data Architect 4.5 product, to make sure the encyclopedia is well indexed and organized.

Long-Name Conversion

The conversion routine will perform the long-name conversion, creating a new encyclopedia in a subdirectory of the old one (with a default name of LName1). The original encyclopedia is left in tact.

Preparing the SA/Data Architect 4.5 Encyclopedia for Conversion

To increase the chances of a smooth and successful conversion, we strongly recommend that the encyclopedia goes through a few preparatory steps using the old version of the software.

1. Select **Tools, Reorganize** to restructure the encyclopedia database.

2. Select **Tools, Dictionary Update** to update and correct the dictionary.

3. Select **Reports, Expression Check** to invoke a complete data expression syntax check and validation, and update all associated cross referencing. Online error indications are provided.

The utilities described above will produce reports which will warn you of any error that was encountered and whether or not System Architect took any action to fix it. Please read these reports and try fixing as many errors as you can.

Running the Conversion Routine

Select the Encyclopedia

Select the encyclopedia to be converted as follows:

1. Launch System Architect. You may open any current encyclopedia.

2. From the **Tools** menu, select **Conversion**.

3. In the **Existing** tab, browse to the directory of the encyclopedia that you wish to convert. Click **OK** to begin the conversion process.

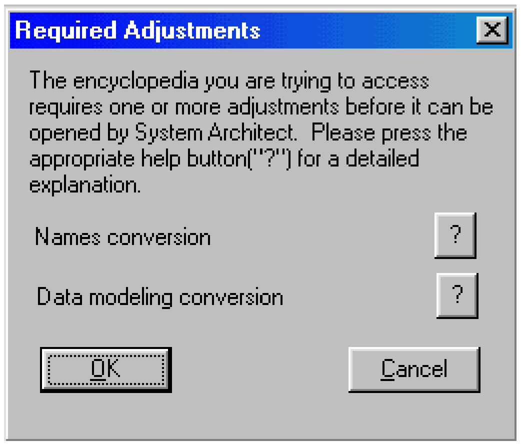
Required Conversion Varies by Encyclopedia

System Architect examines the encyclopedia and tells you the types of conversions that are necessary in the **Required Adjustments** dialog. Depending on the type of encyclopedia you are trying to convert, up to three different conversions may be necessary:

a) Network conversion is needed when opening a merge encyclopedia with a network version of System Architect. Please note that you can not open a network encyclopedia with a merge version of SA/2001.

b) Long name conversion is needed for all SA/Data Architect 4.5 encyclopedias.

Long Names Conversion


Required Adjustments

The encyclopedia you are trying to access requires one or more adjustments before it can be opened by System Architect. Please press the appropriate help button("?") for a detailed explanation.

4. In the **Required Adjustments** dialog, select **OK**. The long names conversion will run first, creating a new encyclopedia in a directory within the original encyclopedia. The new encyclopedia is named LName1 by default (if LName1 already exists, then LName2 is created, and so on). You cannot specify a different name for the long-name conversion encyclopedia.

What to Do After the Conversion

When the conversion is complete, System Architect should have the newly converted encyclopedia open. The encyclopedia is ready for further work.

3

Modifying Macros for Use with V10.1

Introduction

This chapter provides details on converting V8 encyclopedias (dBASE) to V9 (SQL Server) and later. It also describes the workflow changes that users who have been using System Architect V8 and before will encounter when using the V9 or later version of the product.

Modifying Macros that Run Off Menu System for V10.1

As of V10.1 System Architect has almost completely released the menus and toolbars from their previous fixed format. This means that users are now able to customize menus, and their customizations will remain. The only exceptions to this are the Draw menu and toolbar, where customization is only possible after the default list of drawing tool commands or buttons.

The goal of this change was to enable tailored menu and toolbar configurations for specific user groups via SA Catalog Manager.

How Will this Affect Current Users?

In general users should notice no significant changes to menus in this release of System Architect. All menu items and toolbars will remain where they were in previous versions, it is expected that the current layout will eventually be known as Classic System Architect.

Users will however be able to customize their menus toolbars (except the Draw toolbar) and have menu items remain where they are place, without the need to add macro code.

How Will this Affect Macros Which Change Menus?

IBM's aim to ensure backward compatibility is always a top priority, however some improvements are not without side effects. The updates to the menu system have been quite extensive and have involved a change in philosophy, where items are no longer removed from the menu system. Instead, where a menu item would previously have disappeared, it is simply hidden. This change of emphasis does impact on code written to support macro menus items.

Previously macro menu code tended to be written taking into account the fact that System Architect would destroy certain menus, and force them to be redrawn. This destruction meant that users who required menu items to disappear during a session, could save themselves the need to actually remove an item from a menu programmatically. A typical example might be when a diagram was closed – the tools menu would be destroyed and re-created. Thus the user would only be required to re-insert the menu item if it was required at that time. Now that System Architect does not destroy menus, these menu items will remain visible.

Essentially all macro menu code which requires menu items to be removed from the menu system, will need code added to the menu event handler logic, to remove (hide) a menu item that should no longer be visible.

What Will Happen if No Change is Made to Macro Code Which Affects Menu Items ?

If you do not make any changes to macros that you have written to appear on menus, the following will be the consequences:

- If all the macro menu items are expected to remain available for the full session – the code will run as before, no change is required.
- If some macro items are expected to disappear when the menu they are on is ‘redrawn’ – these macro menu item will be shown the first time they are activated and remain visible for the duration of the session.

How Can I Change the Macro to Follow the New Regime?

The changes are quite simple to implement.

1. Identify the piece of code which inserts the item in the menu
2. Add an Else clause, which is operated when the macro should not be shown.
3. In the Else clause add code to remove the item from the menu. This will hide the item.

Example

Here is an example:

```
Option Explicit
Public WithEvents App As Application  ' The application which will raise events
Private Sub App_ReportsMenuUpdate()
   ' This event happens whenever a diagram is shown or hidden.
    UpdateSamplePopupMenu
End Sub
Public Sub UpdateSamplePopupMenu()
    Dim lngErr As Long
    If Not Application.Encyclopedia Is Nothing Then
        If Not Application.Encyclopedia.GetCurrentDiagram() Is Nothing Then
            ' Insert our popup menu 'MENUSAMPLEPOPUP' in the 'Reports' menu, before
the 'Print all Diagrams' item.
            lngErr = Application.InsertPopupMenuItemInMenu(MENUSAMPLEPOPUP,
"&Reports", "Print all &Diagrams")
        Else
            ' Remove our popup menu 'MENUSAMPLEPOPUP' from the menu system
            lngErr = Application.RemoveItemFromMenu(MENUSAMPLEPOPUP, "")
        End If
    End If
End Sub
```

The code in **bold** above has been added to ensure the macro item is removed (hidden) from the menu system, where previously it would have been removed by System Architect’s menu redrawing technique. The use of the menu name to remove from is not required as it is no longer used.

How do I Reset My Toolbar to the Default State ?

There are two ways to reset the toolbar system.

- Complete reset
  - This removes all customizations from the menu system and returns the menu system to initial settings.
- Macro item reset
  - This removes all macro menu customization and tools, but retains any product menu customization.

To perform a complete reset:

1. Right click on the toolbar
2. Select Customize
3. Select the Toolbars tab
4. Select any toolbar from the list
5. Press the 'Reset' button
6. You will be given a warning message. Respond with Yes.
7. Close the Customize dialog
8. Restart SA

To perform a macro menu reset:

1. From the 'Tools' menu select the 'Macros' popup
2. From the 'Macros' popup select the 'Macro Projects' dialog
   a. *If a particular macro project is causing problems, uncheck it's active state in the dialog, and press Apply.*
3. Choose the icon at the top left of the dialog with the X through a menu – it has the tooltip 'Reset macro menu customizations'
4. You will be given a warning message. Respond with Yes.
   a. *If a particular macro project was causing problems, recheck it's active state in the dialog, and press Apply.*
5. Close the Macro Projects dialog
6. Restart SA (not essential, but some macro projects may not reload their menus until after a restart)

What are the New Guidelines for Adding Macro Items to Menus Programmatically?

How Do I Add a Macro to a Standard System Architect Menu ?

The main SA2001.Application method to support the adding of macro items is:

InsertMacroItemInMenu(MacroName as String, MacroItemCaption as String, InMenuTitleCaption as String, [BeforeMenuItemCaption as string]) as long

This function adds the MacroName (which comprises - "<Project>.<Module>.<Subroutine>") using the caption MacroItemCaption to the SA/User menu titled InMenuTitleCaption, before the menu item titled BeforeMenuItemCaption. If no Before item is specified then the item is added to the end of the menu. If the before item is #TOP# then the item is inserted at the top of the menu.

The function returns 0 if successful, otherwise none zero.

In other words:

```
   Set App = New SA2001.Application
x = App.InsertMacroItemInMenu("MyProject.MyModule.MySub",
"&Test Item", "&Tools","&Macros")
```

This inserts the item 'Test Item' in the Tools menu, before the Macros popup item. When the item is clicked on, the subroutine 'MySub' is executed in the module 'MyModule' in the project 'MyProject'.

For this to succeed the subroutine specified MUST already exist. Also, the captions MUST be specified exactly as in the menu.

For example,Tools is represented as the string &Tools

An ampersand implies an underscore under the following letter. This IS case sensitive.

The subroutine name, should NOT have ()'s following it.

The adding of BMPs to menu items is done by the AssignBMPtoMacroItem method.

AssignBMPtoMacroItem(MacroName as String, BMPFileName as String) as long

This function returns zero if successful, or none zero otherwise.

This should be done prior to adding the macro item to the menu - but need only be done once.

For example:

```
Set App = New SA2001.Application

x = App.AssignBMPtoMacroItem("MyProject.MyModule.MySub",
"C:\Piccy.BMP")

x = App.InsertMacroItemInMenu("MyProject.MyModule.MySub",
"&Test Item", "&Tools","&Macros")
```

This will add the piccy.bmp to the MySub macro, then add it to the menus as in the previous example. Note that the customize menus (right click on menus) option will now include the Piccy.BMP and the caption '&Test Item', if you wished to put the menu item anywhere else in your menus.

Using the Events

The following SA2001.Application events may assist macro code when deciding when to make menu visibility changes:

- MainMenuUpdate: When the whole menu is affected
- MethodMenuUpdate: When dictionary type menu items are updated
- ToolsMenuUpdate: When tools type menu items are updated
- ReportsMenuUpdate: When report type menu items are updated
- App_ShutDown: When SA shuts down

Events are NO LONGER required to keep user items in the menus. However if you wish to remove items from a menu, the following sections of code provide the outline for such event handling. (Note: The Project is called 'MyProject')

MODULE - AutoExec

```
' Main module which maintains the VBA based event handler
Dim EventHandler As EventCls
' Main Subroutine which starts the event handler
Sub Main()
  Set EventHandler = New EventCls     ' Create an event handler
  Set EventHandler.App = Application  ' Connect the event handler to SA
  ' Perform any additional start up work here
  ' This is a good place to add you BMPs and create your PopUp menus
  InitBMPs
  InitPopUps
End Sub

Sub InitBMPs()
  Dim x As Integer
  x =
Application.AssignBMPtoMacroItem("MyProject.MyModule.PRINTDIA
GRAMS", "SAWORD.BMP")
End Sub

Sub InitPopUps()
  Dim x As Integer
  ' NB: Once a popup is created, it remains until it is removed by
'RemovePopupMenu', and can be added/removed from menus at will.

  ' Create the popup menu with its bitmap
  x = Application.CreatePopUpMenu("Sample
Macros","SAWORD.BMP")
  ' Add the item to the popup
  x =
Application.InsertMacroItemInMenu("MyProject.MyModule.PRINTDIA
GRAMS", "&Print a Diagram", "Sample Macros")
End Sub
```

CLASS - EventCls

```
Public WithEvents App As Application  ' The application which will raise
events

Private Sub Class_Initialize()
  ' No need to do anything
End Sub

Private Sub Class_Terminate()
  ' No need to do anything
End Sub
```

```
Private Sub App_ReportsMenuUpdate()
  ' Reports menu has been redrawn
  Dim x As Long
  x = App.SetSeparatorBefore("&Report Generator...", "&Reports",
True)

  ' Insert a menu item in the 'Reports' menu, before the 'Report
Generator...' item, called 'Print all Diagrams' which calls
PrintDiagrams2
  x =
App.InsertMacroItemInMenu("MyProject.MyModule.PRINTDIAGRAMS
2", "Print all &Diagrams", "&Reports", "&Report Generator...")

  ' Insert our popup menu 'Sample Macros' in the 'Reports' menu,
before the 'Print all Diagrams' item.
  ' NB: You should not keep destroying and creating popups here,
unless you have a good reason - do this once at initialization.
  x = App.InsertPopupMenuItemInMenu("Sample Macros", "&Reports",
"Print all &Diagrams")
End Sub
```

MODULE - MyModule

```
Public Sub PrintDiagrams()
' User Code to run
MsgBox "Print Diagrams"
End Sub
Public Sub PrintDiagrams2()
' User Code to run
MsgBox "Print Diagrams2"
End Sub
```

How Do I Add a Popup ?

Use the CreatePopUpMenu method with the name you wish to use for your popup. And a BMP if desired. The popup is added to the root collection of popups in SA.

When you leave SA you no longer need your event handler class to remove any added popups using RemovePopUpMenu

How Do I Add Macro Items to a Popup ?

In exactly the same way as you do for adding to SA menus.

How do I Add a Popup to an SA Menu ?

You may add a popup to a System Architect menu in exactly the same way you add a macro item to a menu, except, use InsertPopupMenuItemInMenu and do not specify a macro name.

For example:

InsertPopupMenuItemInMenu(PopUpName as String, InMenuTitleCaption as String, [BeforeMenuItemCaption as string]) as long

How Do I Add a Separator to a Menu Item ?

Use SetSeparatorBefore, provide the menu name and item name, then set True for a separator and false for no separator.

How Do I Remove an Item from a Menu ?

You can remove popups or menu items using the same method: use RemoveItemFromMenu specify the name of the menu and the name of the item.

Note: You can only remove items that you have added. For example, System Architect menu items cannot be removed. The items are not actually removed, they are instead hidden. This allows users to customize the items to be elsewhere in the menu system. Note you can specify “” for the Menu name, as the Menu name is now ignored. The item will be hidden wherever it is in the menu system.

What Does RemovePopUpMenu Do ?

Since all popups and tools which are added to the menu system will remain even after SA has been restarted, it may become necessary to remove a popup menu which is no longer required. The App.RemovePopUpMenu(<PopupName>), method will totally remove the popup from the menu system.

If this method is used in a macro, say at SA shutdown, the popup and all its customization will be removed everytime SA shuts down. This would have the effect, that customization of this popup is not kept between SA sessions.

4

System Architect V9, V10, and Later Conversion

Introduction

This chapter provides details on converting V8 encyclopedias (dBASE) to V9 (SQL Server) and later. It also describes the workflow changes that users who have been using System Architect V8 and before will encounter when using the V9 or later version of the product.

Converting Encyclopedias

Encyclopedias created with versions of System Architect prior to V9 (which were in a proprietary format based on dBase) need to be converted to SQL Server or MSDE. We repeat the steps presented in Chapter 2, on how to do this. Later in this chapter, details are provided on changes that may be necessary to user-created VBA macro files.

Preparatory Step if You Have Custom Graphics

The conversion process is as follows:

1. If you have custom .bmp and .wmf files associated with an encyclopedia, you must perform two preparatory steps:

   a. Use Windows Explorer or equivalent, to copy the custom .bmp and .wmf files associated with the V8 encyclopedia to be converted to the images folder of the main System Architect software directory (<C>:\Program Files\Telelogic\System Architect Suite\System Architect\Images).

   b. Copy the usrprops.txt file of the V8 encyclopedia that contains custom images, and paste it into the main System Architect directory (<C>:\Program Files\Telelogic\System Architect Suite\System Architect), replacing the usrprops.txt file that is there.

   These two steps provide a shortcut to making the SQL Server or MSDE encyclopedia that you create, ready for conversion

Convert Encyclopedias By Using Merge Facility

2. In System Architect, create a new encyclopedia in SQL Server or MSDE. (This new encyclopedia will now already contain all necessary custom pictures and usrprops.txt.). Procedures for creating a new encyclopedia in SQL Server or MSDE are provided in the on-line help or in the Installation Guide.

3. Use System Architect's Encyclopedia Merge utility to ***merge*** all information from the V8 (dBase) encyclopedia into the V9 encyclopedia, as follows:

   a. In System Architect, select **Tools, Merge**.

   b. In the **Merge/Extract** dialog, click the **Browse** button. The **Open Encyclopedia** dialog will open.

c. In the **Open Encyclopedia** dialog, toggle on **dBase Encyclopedia** in the **Existing** tab, and navigate to the path of the V8 encyclopedia to be converted. Select **Open**.

d. Back in the **Merge/Extract** dialog, click on the **Everything** button.

e. Keep the **If Items are identical** choice at its default "**Use the more recent**".

f. Click **OK** in the Merge/Extract dialog. The encyclopedia will be converted for use with System Architect V9 – it will become a SQL Server database.

Modification of VBA Macros and SA/Word Macros

Users who have built their own VBA macros for use with System Architect, or their own SA Word VBA macros (which use the sawd2001.dll file), need to be aware of certain changes that have been made to System Architect's IMF calls, and may need to modify their code appropriately. Details are provided in the following sections.

Changes to VBA Macros

Changes necessary to VBA macros for use in System Architect V9 can be summarized as follows:

- **Microsoft SQL Server Encyclopedia Access:** In version 9 of System Architect the concept of a network lock file does not exist with respect to Microsoft SQL server encyclopedias. To ensure some form of exclusive use of the encyclopedia in order to avoid locking and collisions issues, two new functions were implemented: *CriticalRegion* and *AccessDefinition.*
- **ISAImf Objects:** Existing macros that use the *SA2001.ISAImf* object must have their Set statement changed to prevent a 'Type mismatch' run-time error.
- **SA2001.Encyclopedia.Fullname Property:** If a macro has used the *path* to the Encyclopedia *folder* to locate and access a file, such as 'D0000001.wmf' or 'Usrprops.txt', then it will need to be modified for use with V9 encyclopedias, which are maintained within SQL Server databases and store such files in a Files table. The required file will need to be *exported* before it can be used. The ISAImf object has an *SAFileExport* function that allows this.
- **ISAImf.SAGetEncyclopediaPath:** For the same reasons as those given for not using .Fullname (above), the *GetEncyclopediaPath* call cannot be used to locate and access encyclopedia files.
- **Importing Files:** In some cases it may be necessary to import files into the encyclopedia, so that an internal reference can be made. An example of this is when the .SetField() subroutine is used to link a picture symbol with its image file.

Details on the changes necessary for the situations described above are provided in the paragraphs that follow.

Microsoft SQL Server Encyclopedia Access

In version 9 of System Architect the concept of a network lock file does not exist with respect to Microsoft SQL server encyclopedias.

To ensure some form of exclusive use of the encyclopedia in order to avoid locking and collisions issues, two new functions were implemented: **CriticalRegion** and **AccessDefinition**.

The following sequences of code in VBA macros should be replaced, wherever possible, by calls to AccessDefinition (described below) with the appropriate directive flags.

- FindDefinition, CreateDefinition
- SAFindDefinition, CreateDefinition, OpenDefinition
- FindDefinition, OpenDefinition
- FindDefinition, OpenLockDefinition
- CreateDefinition (where FindDefinition is not preceding it)

Any block of code relying on the fact that it had exclusive access to the encyclopedia (by locking it) and has modified records without locking them first, **must** be altered so that a lock is placed on the record before modifying it (OpenLockDefinition) and a call to CloseDefinition is present.

If after replacing the above types there are still sequences remaining, they should be bounded by a Critical Region.

We recommend that a full analysis of the VBA code be done before any changes are made.

1. CriticalRegion:

The **CriticalRegion** function has been implemented in order to avoid collisions on a SQL Server encyclopedia in a network environment. The function offers a form of exclusive access to the encyclopedia.

A critical region is an exclusive lock taken on a system resource so that any other process that attempts to start a critical region will wait until the current one is completed, thus allowing content sensitive processes to be single threaded. The call to end a Critical Region should only be made if the call to begin the Critical Region succeeded.begins or ends a critical region.

int CriticalRegion(
BOOL bLock, /* Input: action to take – start or stop */ WORD IOErrorId)/* Input: Id for error reporting*/

Parameters:

bLock

Boolean indicating whether to stop (FALSE) or to start (TRUE) the critical region

IOErrorId

Number ranging between 8600 and 9000 used for IO error reporting.

Return Values:

If the function succeeds the return value is SUCCESS, otherwise the value is DB_ERROR.

2. AccessDefinition:

The **AccessDefinition** function finds or creates, opens, or open-locks a definition.

int AccessDefinition(

LPSAIDENT ddPrevDef,/* Input: previous definition */

LPCTSTR lpName,/* Input: definition name */

LPWORD lpwMinorType,/* Input: definition type */

DWORD dwCommand,/* Input: combination of commands */

LPSAIDENT lpidDefinitionRet,/* Output: accessed definition id */

LPSAHANDLE lphDefHandleRet,/* Output: accessed definition handle */

LPWORD lpwIsDefNew)/* Output: indicates the creation of the definition */

Parameters:

ddPrevDef

Pointer to the previous definition's ddId. It can be NULL.

lpName

Name of definition to be opened. It can be a qualified name.

LpwMinorType

Pointer to the minor type of the definition to be opened.

dwCommand

Specifies the action to take. The following values can be combined by using the bitwise OR operator:

Value	**Meaning**
IMF_DEFACCESS_CREATE	If definition attempting to be accessed couldn't be found, create it.
IMF_DEFACCESS_OPEN	Open definition.
IMF_DEFACCESS_LOCK	Lock the definition being opened. Used in conjunction with IMF_DEFACCESS_OPEN.

lpidDefinitionRet

Pointer to the opened/created definition's ddId.

lphDefHandleRet

Pointer to the opened/created definition's handle.

lpwIsDefNew

Pointer to a variable indicating whether or not we created this definition.

Return Values

If the function succeeds, the return value is IM2XE_OK.

If the function fails, the return value is IM2XE_*. For information on the returned error, consult IM2XDEF.H located in the SADESIGN directory.

Remarks

The command to find the definition is implied. In other words this function can be invoked with a *dwCommand* of 0 to find a definition.

The AccessDefinition function sets its own critical region and was designed to replace sequences of the following types:

- FindDefinition, CreateDefinition
- SAFindDefinition, CreateDefinition, OpenDefinition
- FindDefinition, OpenDefinition
- FindDefinition, OpenLockDefinition
- CreateDefinition

When AccessDefinition is required to create a definition, it will actually write a record to the database. The function CreateDefinition did not (and still does not) do this; a call to SaveDefinition was needed to actually write the record to the database. Thus, any extant code sequence which used to call CreateDefinition (creates a memory definition block) and later decide whether to call SaveDefinition (writes out the record) or not, will have to be modified so that, if it is decided not to actually create this record, the code must now call DeleteDefinition to remove the unwanted record.

VBA macros that use IMF calls may require adjustment. The following IMF calls have been affected by the support for SQL Server databases:

- SAGetEncyclopediaPath
- Encyclopedia.Name
- Encyclopedia.Fullname

SAGetEncyclopediaPath: The problem with using this ISAImf call is the same as for Encyclopedia.FullName, instead of a folder reference, a file reference is supplied. This is due to the fact that an encyclopedia is now referenced through a .UDL file and not stored within a folder. This is likely to be used simply to provide information and should not present a problem.

Encyclopedia.Name: This has been difficult to trace – but normally used as information about the name of an Encyclopedia. The output will identify the UDL filename and not a subdirectory.

It is interesting to note that the ISAImf call SAGetEncyName returns the SQL Server's Database Name and not simply the UDL filename.

Encyclopedia.FullName: **As for SAGetEncyclopediaPath, a filename is returned and not a subdirectory name.**

ISAImf Objects

Existing macros that use the SA2001.ISAImf object must have their Set statement changed to prevent a 'Type mismatch' run-time error, as follows:

Previous Code:

If the existing macro had code as follows:

```
Dim imfSA As SA2001.ISAImf
..
Set imfSA = SA2001.Application
```

New Code:

It should be changed to the following for V9:

```
Dim imfSA As SA2001.ISAImf
..
Set imfSA = appSA.Interface("ISAImf")
```

SA2001.Encyclopedia.Fullname property

If a macro has used the ***path*** to the Encyclopedia ***folder*** to locate and access a file, such as 'D0000001.wmf' or 'Usrprops.txt', then it will need to be modified for use with V9 encyclopedias, which are maintained within SQL Server databases and store such files in a Files table.

The required file will need to be **exported** before it can be used. The ISAImf object has an **SAFileExport** function that allows this.

For example, if the code in the macro is as follows:

```
..
Let strEncyclopediaPath = appSA.Encyclopedia.FullName
Let strUSRProps = strEncyclopediaPath & "UsrProps.txt"
```

In System Architect V9, the .Fullname property returns the UDL filename and its path. It might be preferable to use the **.Path** property that returns only the path to the UDL file. The path used to store the exported file could be any folder with write access.

The **SAFileExport** function takes the internal file reference and the external path and filename as parameters, returning 0 if the export is successful.

```
Let strPath = appSA.Encyclopedia.Path
Let strUSRProps = strPath & "USRProps.txt"
Let lngIMFerror = imfSA.SAFileExport("USRProps.txt",
strUSRProps)
```

ISAImf.SAGetEncyclopediaPath

For the same reasons as those given for not using .Fullname (above), the **GetEncyclopediaPath** call cannot be used to locate and access encyclopedia files. The SAFileExport function should be used, as described above.

Importing Files

In some cases it may be necessary to import files into the encyclopedia, so that an internal reference can be made. An example of this is when the .SetField() subroutine is used to link a picture symbol with its image file, as follows

```
Call symPicture.SetField(SYMFLD_U_S1_ZPPICFILE,
strFilePath)
```

The **SAFileImport** function takes the external path and filename and the internal file reference as parameters, returning 0 if the import is successful.

```
Let lngIMFerror = imfSA.SAFileImport(strFilePath,
strInternalName)
```

```
..
Call symPicture.SetField
(SYMFLD_U_S1_ZPPICFILE, strInternalName)
```

Changes to SA Word Template Files

SAGetEncyclopediaPath

If an SA Word template uses the SAGetEncyclopediaPath function provided by sawd2001.dll file, to locate encyclopedia files, possibly for inclusion in the document, then these files need to be exported. The reason for this is that in V9, encyclopedias are maintained within SQL Server databases, not encyclopedia directory paths.

The **SAFileExport** function will allow the export of an internal file such as a diagram's .wmf file.

```
Let intRet =
SAGetEncyclopediaPath(strEncyclopediaPath, 255)

Let strFilePath = strEncyclopediaPath & strFilename

..

ActiveDocument.InlineShapes.AddPicture
Anchor:=Selection.Range, _ FileName:= strFilePath,
LinkToFile:= False, _ SaveWithDocument:=True
```

For System Architect V9, it will be necessary to identify a suitable location for a temporary, exported file. Once the picture of the diagram has been inserted into the Word document the file will normally be deleted. The SAFileExport function takes the internal file reference and the external path and filename as parameters, returning 1 if the export is successful.

```
Let strPath = "C:\temp\" ' or other suitable path

Let strFilePath = strPath & strFilename

Call SAFileExport(strFilename, strFilePath)

..

ActiveDocument.InlineShapes.AddPicture
Anchor:=Selection.Range, _

FileName:= strFilePath, LinkToFile:=False, _

SaveWithDocument:=True
```

Workflow Changes

This section describes workflow changes that users of System Architect V8 and before will encounter when using System Architect V9, V10, or later.

1. License Manager Must Run on Stand-Alone Machines with System Architect

If you are running System Architect on a stand-alone machine (such as a laptop), you will need to install and run License Manager on that machine before running System Architect. Please see the Installation Guide.pdf for full instructions.

2. Creating and Opening Encyclopedias

Creating an Encyclopedia:

Creating an encyclopedia in System Architect V9 differs from previous versions of the product. You either create an encyclopedia on an MSDE server, or a SQL Server 2000 server. Full details are specified in the on-line help and in Chapter 3 of the Installation Guide. This manual is provided in .pdf format on the installation Volume 5 Documentation CD. If you do not have access to the Documentation CD, please contact your System Administrator.). A brief summary is provided here:

1. Select the **File, Encyclopedia Open** command.
2. In the **New** tab of the **Open a Project** dialog, specify a **Server Name**.

   For SQL Server 2000:

   Your System Administrator should provide you with this information -- which server is being used to create System Architect encyclopedias.

   For MSDE:

   With MSDE installed on your local machine, the name of your machine should be represented in the **Server Name** drop-down list. You may either select the name of your machine, or **local** – both result in opening the MSDE server installed on your local machine. For MSDE is installed on the network, you must select the appropriate server.

3. Select a method for Authentication, as follows:

<u>For SQL Server on the Network:</u>

Select SQL Server, and fill in the Username and Password that your system administrator
has provided to enable you to access the network server.

Important Note:

To create/open System Architect encyclopedias on SQL Server 2000 in a network environment, the System Administrator **must set up appropriate access rights** on SQL Server 2000 for all System Architect users. SA users will not be able to create encyclopedias unless they have proper rights to create databases on the server. They wont be able to open an encyclopedia unless they have rights to the database where the encyclopedia resides.

<u>Access Rights for SQL Server 2000:</u>

Server Role: For users to be able to create an encyclopedia on a SQL Server 2000 server, they must either have a Server Role of System Administrator or Creators. This is granted to the user by the System Administrator through SQL Server 2000 Enterprise Manager.

Database Access: For users to be able to open an encyclopedia and read and create definitions within it, they must have Database Access of db_datareader and db_datawriter. This is granted to the user by the System Administrator through SQL Server 2000 Enterprise Manager.

<u>For MSDE on your Local Machine:</u>

a. If you are using a Windows NT-based operating system (Windows 2000, Windows XP, Windows NT, etc), you may use either Windows NT authentication or SQL Server authentication, as follows:

   i. Toggle on Windows NT authentication, or

   ii. Toggle on SQL Server, and in the Login Information group, type in a Username of sa
   (for System Administrator), and leave the Password blank.

b. If you are using Windows 98 or Windows ME, you must use SQL Server authentication.
   Toggle on SQL Server, and in the Login Information group, type in a Username of sa

(for System Administrator), and leave the Password blank.

4. Specify the name of the database to create on that server. A Universal Data Link (UDL) file is created which contains the connection string to the server and database. The file can be created anywhere. Once created, the UDL file always correlates to a specific System Architect encyclopedia on a specific server (so it makes sense to name it the same as the encyclopedia you are creating).

Opening an Encyclopedia:

Once created, opening an existing encyclopedia is performed via System Archtiect's Open a Project dialog (File, Open Encyclopedia), and selecting the .udl file of the encyclopedia. You may use browse button of the Existing tab in the Open a Project dialog to find and select the path and .udl filename of the encyclopedia, or use the Recent tab to select recent projects.

3. Encyclopedia Maintenance

Verify and Repair -- you may still run the Verify and Repair facility on an encyclopedia by selecting it from the Tools menu.

Dictionary Update -- you may still run the Dictionary Update facility on an encyclopedia by selecting it from the Tools menu.

Reorganize -- the Reorganize command is no longer available from the Tools menu. The Reorganize command used to reorganize and compact a dBase encyclopedia. This function is now performed by ***shrinking*** the database, either using SQL Server's Enterprise Manager tool, or Telelogic's SAEM (Enterprise Manager) tool for MSDE. Instructions on how to shrink a database are provided in the on-line help of each respective tool.

Encyclopedia Comparisons Using SAEncomp -- SAEnComp, the encyclopedia compare facility, cannot currently work directly on encyclopedias created on SQL Server or MSDE in System Architect V9. To compare encyclopedias using SAEnComp, you must extract those encyclopedias to a dBase version using System Architect's Extract facility (Tools, Extract command). A **Working Document** on how to perform this is provided in the downloads area of the IBM Telelogic website.

4. Customizing an Encyclopedia's Metamodel

Editing USRPROPS.TXT:

The User Properties (USRPROPS.TXT) file is used to modify and extend the metamodel of an encyclopedia. It now resides in the Files table of the encyclopedia database. To edit it, you must perform the following steps:

1. Select Tools, Customize User Properties, Export USRPROPS.TXT (Encyclopedia).
2. In the Export User Properties dialog, select a directory to export the file to and click OK. The file will be placed in the specified directory, and automatically opened in Notepad (you may change this setting via Tools, Session Options).
3. Edit the file as appropriate. (For more information about editing the USRPROPS.TXT file, refer to the System Architect Extensibility Guide manual which is provided in .pdf format on the installation **Volume 5 Documentation CD**. If you do not have access to the Documentation CD, please contact your System Administrator.)
4. Select Tools, Customize User Properties, Import USRPROPS.TXT (Encyclopedia) to import the file back into the Files table of the encyclopedia.
5. Reopen the encyclopedia (File, Encyclopedia Open command) to make your changes take effect.

Images on Diagrams:

Images Depicting Symbols (USRPROPS.TXT Depiction Command):

User-defined images (WMF files to depict symbols on diagrams and BMP files for their respective representation on toolbars) are now housed in the Files table of an encyclopedia database. Such graphic files referenced in the USRPROPS.TXT of an encyclopedia must be imported into the Files table of the encyclopedia using the Encyclopedia File Manager command (Tools, Encyclopedia File Manager), or SQL Server 2000's Enterprise Manager program, or if you are using MSDE, the System Architect Encyclopedia Manager utility (SAEM).

Picture File Copy:

Copying a picture from an external directory onto a System Architect diagram (Draw, Picture, File Copy) creates a copy of the bitmap (.bmp) or metafile (.wmf) specified, and places that copy into the **Files** table of the encyclopedia database. The name of the picture copy is provided by System Architect, with a naming convention of Pnnnnnnn.bmp or Pnnnnnnn.wmf (where nnnnnnn = a sequential 7-digit number).

Picture File Reference:

You may also create a reference to a picture in an external directory, as you could with previous versions of System Architect (Draw, Picture, File Reference). System Architect creates a reference to the bitmap or metafile specified (it ***does not*** create a copy in the **Files** table of the encyclopedia database). If the picture is removed or moved from the external directory where it resides, the image of it on the System Architect diagram will disappear.

Encyclopedia File Manager:

There is a new Encyclopedia File Manager command in System Architect, accessed via the Tools menu. This command enables you to import/export files to/from the **Files** table, **Error_Log** table, **Info_Log** table, and **Network_Log** table of an encyclopedia.

The Files Table:

The Files table contains the following files:

- **Graphics that You Paste or Copy Into an Encyclopedia:** A WMF or BMP file is created automatically every time you paste or file-copy a picture-type symbol into a diagram. One WMF or BMP is created for each picture. You can turn off the automatic creation of WMF files in SA2001.INI
- **Graphics that You Specify Via Depictions Clause in USRPROPS.TXT:** System Architect enables you to specify new graphics for symbols drawn in an encyclopedia. You specify this using the Depictions clause in USRPROPS.TXT. You may specify new .BMP files for images on toolbars, and new .WMF files for use on diagrams. To make this happen, you must make the necessary changes to USRPROPS.TXT, and import the respective .BMP and .WMF files into the Files table of the encyclopedia.
- **The Property Set Files:** Three property set files determine the metamodel and the diagram and property sets that are turned on for an encyclopedia. They are as follows:

SAPROPS.CFG – created by IBM Specifies the metamodel of the encyclopedia.

USRPROPS.TXT – users add code to this file to specify metamodel extensions that they require.

SADECLAR.CFG – specifies the diagram types and property sets that are turned on for the encyclopedia. Is set by the Tools, Customize Method Support command in System Architect.

- **The AUTOEXEC.CSV File:** When you start a new project, the AUTOEXEC.CSV is automatically imported into the Files table. AUTOEXEC.CSV contains the Trigger Templates, which can be used as the basis for defining triggers. Complete information on the use of triggers can be found in the on-line help for the Schema Generator.
- **Format File (AUTOEXEC.STY, or 'style sheet'):** The format file, which determines the look of symbols that are drawn on diagrams. In particular, it refers to a symbol's size, shape, line thickness, font, text justification, etc. You can change these symbol characteristics to fit your project's needs.

  It is not mandatory that the Format file be housed in the Files table. You may place the Format File as follows:

  - You may put it in a central directory and then specify that path in the SA2001.ini file for each user. To do this you open the SA2001.ini file using a text editor such as Notepad (the sa2001.ini file is stored in the<C>:Documents and Settings\<user profile name>\Local Settings\Application Data\Telelogic\System Architect, and then create a setting such as FORMATFILE = <C>:\Program Files\Telelogic\System Architect Suite\System Architect\Autoexec.sty. In this case the name Autoexec.sty is arbitrary; you may name it something else, as long as the extension remains .sty (ie, Payroll.sty).
  - You may name the format file AUTOEXEC.STY and import a copy into the Files table of all your project encyclopedias (using System Architect's Tools, Encyclopedia File Manager command). The format file loaded in the FILES table of the encyclopedia is auto-loaded upon opening of the encyclopedia. It must be named Autoexec.sty

The Error_Log Table:

The Error_Log table contains the error log of all errors reported during operation of System Architect. This includes errors reported during most import/export operations of the tool, including reverse and forward engineering.

The Info_Log Table:

The Info_Log table lists the encyclopedia tests and thier results. The file gets created in the encyclopedia being tested and can also be opened with any text editor.

The Network_Log Table:

The Network_Log table contains all messages reported regarding network use of System Architect's multi-user version. These messages can sometimes prove useful in tuning a network. A Network Log option in the SA2001.ini file may be set to turn on or off network logging.

5. System Architect Encyclopedia Manager (SAEM)

IBM provides a utility program, System Architect Encyclopedia Manager (SAEM) for users creating encyclopedias on MSDE. The SAEM program provides various functions to System Architect users so that they may perform external tasks on their encyclopedias, such as shrinking, attaching, and deleting databases on a server. Users creating encyclopedias on SQL Server 2000 may use SQL Server's Enterprise Manager program to perform these functions.

The SAEM program is installed separately and comes with its own help file.

6. Version Control and PVCS

Version control of an encyclopedia to PVCS is now performed outside of System Architect. Since an encyclopedia now exists as a database in SQL Server 2000, you use SQL Server 2000's capabilities to version a database to PVCS. A **Working Paper** on this topic is available on the Downloads page of www.telelogic.com.

7. SABindery 6

SABind 6 is no longer supported in System Architect's bindery. The functionality provided by SABind 6 is now done outside of System Architect, using SQL Server 2000 database access functionality. A **Working Paper** on this topic is available on the Downloads page of www.telelogic.com.

5

System Architect V8 Conversion

Introduction

This chapter provides an overview of new functionality in System Architect V8 that requires a conversion for work done in previous versions of the product, specifically System Architect 2001 version 7 and earlier.

Overview of What's Changed

Essentially, conversion has two broad functions to perform:

- to assign a GUID to every object,
- convert any existing UML model elements to the new.

This chapter describes these two broad functions and then lists the warning message that can be generated by the conversion process.

GUID Part of the Conversion

Every diagram, symbol and definition in the encyclopedia being converted will be given a Global Unique Identifier or GUID. Note that this applies to **all** objects, whether they are part of UML or otherwise. The GUID is a 64 character long value provided by Microsoft code. The value is supposed to be unique throughout the known universe and for all time.

Normally, this will not be visible to the user. It is also not declared in SAPROPS (unless it is a key property – see later). However, you can see it in the browser details window if you run with the command line option

–fulldetails

Later, when the reporting system has been modified, it will also be reportable.

As stated before, the user will not normally be aware of an object's GUID. However, Associations in the new UML support are keyed by their GUID and this fact is declared in SAPROPS. Thus GUIDs are described for the Association definition and anything keyed by Association. However, even under these circumstances, the user will neither enter a GUID nor be expected to do anything much with it. Its value is, for all practical purposes, arbitrary. We will probably not even make it visible in SAEDIT when an Association is being edited.

UML Part of Conversion

All UML work is now done in a new set of diagrams. UML models are built within the following diagrams: Use Case, Class, Activity, Sequence, Collaboration, State, Component, Deployment. The old diagram types were all preceded by the UML acronym, for example, UML Use Case, UML Class, UML Activity, etc.

New Types

New definition types have been created to match the UML 1.4 spec. For example, a state in a state diagram no longer contains a list of OMT Process, rather it contains a list of OO Action. In this example, the conversion will take values found in the OMT Process list and make them OO Actions.

Representational Consistency

Of major emphasis in this new release is that relationships drawn between classifiers (ie, classes, Use Cases, etc) are now maintained within the encyclopedia; they are no longer diagram-specific. In layman's terms, what this means is that if you draw an association between class A and class B on one diagram, then create a new diagram within the same package, and drop down class A and class B, the same association will be drawn between them. System Architect ensures that diagrams are adjusted when opened and at other synchronizing events to reflect the addition and deletion of UML Classifiers and relationship lines (ie Associations, Inheritance, etc).

Because previous versions of System Architect did not enforce representational consistency amongst UML models, it was possible for the user to model inconsistent information – for example, model that Class A inherits from Class B on one diagram, and model that Class B inherits from Class A on another diagram. The conversion will only represent one of these two scenarios in the new encyclopedia (the first one encountered). The conversion will also present the user with warning information that this has occurred. If this is not the way the user wishes to model the information, it is up to the user to adjust the converted encyclopedia, or adjust the old encyclopedia prior to conversion to eliminate inconsistencies.

Packages

The package construct now plays a more important role for UML diagrams and definitions. Packages provide a grouping structure and a namespace for UML elements.

Consequently, when you create a UML diagram, you must specify what package the diagram is in. When you create a definition of a UML classifier, you must specify what package the definition will belong to.

The conversion will place all UML diagrams and definitions into a new package named Package1. It is up to the user to rename this package, and/or to create new packages and move diagrams and definitions to the new packages after conversion.

Changes to Diagram, Symbol, and Definition Types

The UML conversion is done using completely new ranges of type numbers for the new UML model element types.

	Old UML Name	***New UML Name***
Diagrams	UML Activity	Activity
	UML Class	Class
	UML Collaboration	Collaboration
	UML Component	Component
	UML Deployment	Deployment
	UML Sequence	Sequence
	UML State	State
	UML Use Case	Use Case
Symbols	Event	Message/Stimulus
	Communicates With	Use Case Association
	Uses	Includes
Definitions	Instance	Association
	OMT Process	OO Process
	UML Communicates	Use Case Association
	UML Dependency	Dependency
	UML Event	Message/Stimulus
	UML Extends	Extends
	UML Package	Package
	UML Uses	Includes

All other type names remain the same.

Messages

Errors

The following messages can appear if conditions are encountered which cause the conversion to be abandoned.

E3021 - Error encountered creating diagram < type> <name>

E3022 - The properties of the source encyclopedia are incorrect.

E3023 - The properties of the target encyclopedia are incorrect.

E3024 - Could not open diagram < type> <name>

E3025 - Diagram < type> <name> could not be saved

E3026 - Diagram < type> <name> could not be closed

E3027 - Could not delete old diagram for < type> <name>

Warnings

The following messages can appear if conditions are encountered which cause the conversion to create conditions other than those that the user might have expected.

W2011 - Constructor < key> collides with Method < key>. The constructor's properties will be merged into the method.

W2012 - Class <name 1> already exists. Conversion will create Class < name 2> for Link Attribute <name 1>.

W2013 - Illegal recursion encountered. <type> < qualified name> will be deleted.

W2014 - Link Attributes are no longer supported. Link Attribute < name> will be transformed into a Class.

W2015 - Class < name> is linked in more than one configuration. Copies of it will be created.

W2016 - Definition <type> <name> property <property name> would be overwritten. Retaining prior value.

W2017 - The <symbol type> <symbol name> on <diagram type> diagram <diagram name> is improperly linked. The link will be broken.

W2018 - Association <association name> on <diagram type> diagram <diagram name> had <node type> <node name> linked to it in contention with an earlier encountered linked item. The link will be broken.

W2019 - The <line type> line named <line name> on <type> diagram <name> was incorrectly connected to the <node type>

<node name> and the <node type> <node name>. It will be removed.

W2020 - Realizes < name> is connected in more than one configuration. Copies of it will be created.

W2021 - UML Communicates < name> is connected in more than one configuration. Copies of it will be created.

W2022 - Property “Formal Parameters” of <class name> < method name> contains duplicate parameter names. Parameter model elements not created.

Advices

The following messages can appear and are intended merely to inform the user. The condition being reported is not unexpected.

I1042 – Deleting definition <type> <name> which is used only by UML

Packages (General)

The package construct now plays a more important role for UML diagrams and definitions. Packages provide a grouping structure and a namespace for UML elements.

In System Architect version 7 and before, packages were used simply as a drawing element. UML classifiers drawn within the confines of a package symbol, or on a diagram that was a child of a package symbol, did not ‘belong’ to the package. In version 8 of System Architect, the package definition ‘contains’ UML classifiers such as classes and Use Cases, and their contents (such as class attributes, class methods, and Use Case steps). When you create a classifier such as a class, you must specify the package it belongs to.

Consequently, in Version 8 when you create a diagram, you must specify what package the diagram is in. When you create a definition of a UML classifier, you must specify what package the definition will belong to.

Packages Within Packages

Package names must be unique within an encyclopedia. A Package can refer to another Package as its “parent” Package, but that reference is not key. Although the user cannot use the same name for two or more packages, we can still present a hierarchy of packages in the browser.

Specifying What Package a Classifier Belongs To

The amount of information you have to fill in for an element will differ depending on the method you use to add the element – for example, if you create a new class definition outside of a diagram (say from **Dictionary, New**), you will need to specify the package the class belongs to; if you add a class by drawing it on a diagram (which already knows what package it represents), the class definition automatically gets keyed by the diagram’s package.

Browser Presentation

UML diagrams, symbols, and definitions are presented in two different views, depending on which tab of the browser you look at.

UML Tab

The UML tab of the System Architect browser now presents all UML diagrams and constituent definitions within a package structure.

All Methods Tab

The All Methods tab of the browser continues to present a flat view of the diagrams and definitions by diagram and definition type.

Packages Conversion (General)

On conversion, System Architect V8 places all diagrams and all contained symbols and definitions within a package named Package 1. Users may rename the default package in the converted encyclopedia:

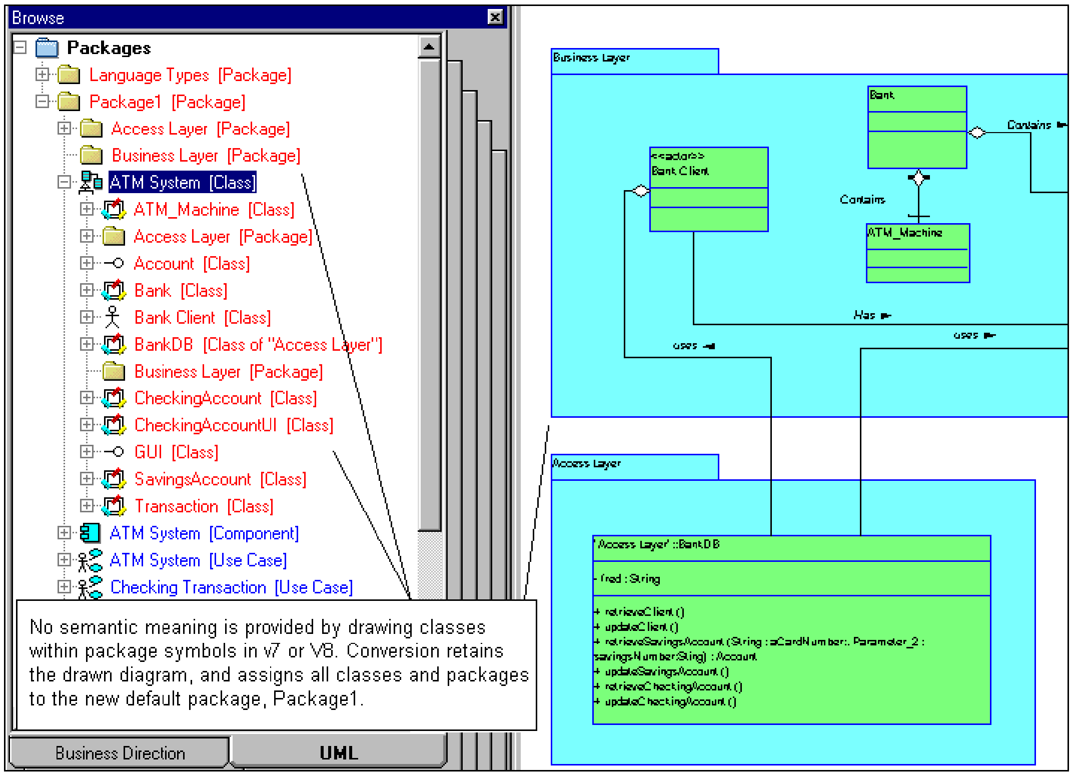
Diagrams and Definitions Placed in Package1 During Conversion

Behavior in SA/2001 v7	Behavior in V8
Packages could be drawn on UML diagrams, but they did not provide ownership of any UML classifiers such as actors or Use Cases. They were simply symbols drawn on diagram workspace.	A Package property has been added to the diagram definition of all UML diagrams. Within the UML tab of the browser, each diagram is displayed under the package it 'belongs to'. UML classifiers (classes, Use Cases, etc) and their constituant properties (ie, attributes, methods, Use Case Steps, etc) are 'keyed' to a package. Package provides a 'namespace' for these items.
What the Conversion Does	
On conversion, System Architect V8 places all UML diagrams and all contained symbols and definitions within a package named Package1. Users may rename the default package in the converted encyclopedia, and they may also move diagrams and definitions to a different package. Diagrams may be moved by simply changing the package name in the diagram's properties. Definitions may be moved by using the browser to copy them, paste them into the different package, and then deleting the old definition.	

Symbols Drawn Within a Package Symbol

Behavior in SA/2001 v7	Behavior in V8
Users could draw a package symbol on a diagram workspace, enlarge its area by dragging on its handlebars, and draw other symbols such as Use Cases, classes, etc, within it. Drawing symbols within a package symbol did not provide any special semantic meaning.	Users can still draw symbols within a package like they did in earlier versions. Similarly, System Architect V8 still does not recognize such drawing to mean the semantic of ownership. Thus, if a Use Case or class is moved inside or outside of a Package, System Architect does not rekey that Use Case or class, or, indeed, make any semantic change to the user's model.
What the Conversion Does	
If a Package symbol had been drawn on a diagram, it is retained on that diagram during conversion. Symbols drawn within the package are also retained. However, the symbols do not get 'keyed' to the package they are drawn within. The package symbol drawn on the diagram becomes part of the default Package1.	

Figure 5-1. Conversion places all UML diagrams and defintions within default package, Package1. Symbols drawn within package symbols do not provide any semantic meaning of ownership.



No semantic meaning is provided by drawing classes within package symbols in v7 or V8. Conversion retains the drawn diagram, and assigns all classes and packages to the new default package, Package1.

Use Case Diagram

The Use Case diagram has had a number of changes.

- The name of the diagram type has been changed from **UML Use Case** to **Use Case**.
- Use Cases and their associated definitions are now contained within a package definition, which provides them with a namespace. Use Case diagrams are also contained within a package, and represented thusly within the UML tab of the browser.
- Representational consistency is maintained for relationships between actors and Use Cases (via the Use Case Association) and between Use Cases (via the Extends, Includes, and Generalization relationships).
- The **Use Case Association** has replaced the **Communicates With** line to show a relationship between an Actor and a Use Case, with applicable representational consistency behavior included.
- Use Cases themselves may now be related by the following relationship lines:
    - **Extends** (renamed from **UML Extends**)
        - You may now also specify the step of the extension for a Use Case Extends line.
    - **Includes** relationship (replaces **Uses)**
    - **Generalization** (new).
- Use Case Steps for a Use Case are now presented in a Grid, as are Pre- and Post-Conditions.

Packages in Use Case Diagrams

Diagrams and Definitions Placed in Package1 During Conversion

Behavior in SA/2001 v7	Behavior in V8
Packages could be drawn on Use Case diagrams, but they did not provide ownership of any UML classifiers such as actors or Use Cases. They were simply symbols drawn on diagram workspace.	A Package property has been added to the Use Case diagram definition. Within the UML tab of the browser, the diagram is displayed under the package it 'belongs to'. Use Cases, and their constituant properties (ie, Use Case Steps, etc) are 'keyed' to a package. Package provides a 'namespace' for these items.
What the Conversion Does	
On conversion, System Architect V8 places all Use Case diagrams and all contained symbols and definitions within a package named Package1. Users may rename the default package in the converted encyclopedia, and they may also move diagrams and definitions to a different package. Diagrams may be moved by simply changing the package name in the diagram's properties. Definitions may be moved by using the browser to copy them, paste them into the different package, and then deleting the old definition.	

Symbols Drawn Within a Package Symbol

Behavior in SA/2001 v7	Behavior in V8
Users can draw a package symbol on a diagram workspace, enlarge its area by dragging on its handlebars, and draw other symbols such as Use Cases, etc, within it. Drawing symbols within a package symbol did not provide any special semantic meaning.	Users can still draw symbols within a package like they did in earlier versions. Similarly, System Architect V8 still does not recognize such drawing to mean the semantic of ownership. Thus, if a Use Case is moved inside or outside of a Package, System Architect does not rekey that Use Case or, indeed, make any semantic change to the user's model.
What the Conversion Does	
If a Package symbol had been drawn on a Use Case diagram, it is retained on that diagram during conversion. Symbols drawn within the package are also retained. However, the symbols do not get 'keyed' to the package they are drawn within. The package symbol drawn on the diagram becomes part of the default Package1.	

Communicates With Line Replaced with Use Case Association

Behavior in SA/2001 v7	Behavior in V8
Communicates With line drawn between Actors and Use Cases.	*Use Case Association* line drawn between Actors and Use Cases.
What the Conversion Does	
All *Communicates With* lines drawn between Actors and Use Cases in System Architect V7 are replaced with a *Use Case Association* line.	

Representational Consistency

Representational consistency has been implemented for the Use Case Associtaion line, which is drawn between an Actor and a Use Case on a Use Case diagram.

Behavior in SA/2001 v7	Behavior in V8
Communicates With relationship line could be drawn between Actor A and Use Case A on one diagram, yet on another diagram, the same Actor and Use Case could be drawn with no relationship drawn between them.	If a Use Case Association line is drawn between Actor A and Use Case A on one diagram, and the same Actor and Use Case are drawn on another diagram, the same Use Case Association relationship is automatically drawn between them.
What the Conversion Does	
Represents the same Use Case Association line between an Actor and a Use Case wherever the combination are drawn. If conversion finds an actor-Use Case pair that should have a Use Case Association line drawn between them, it issues a warning message: **W2021** - UML Communicates < name> is connected in more than one configuration. Copies of it will be created.	

Use Case *Uses* Relationship Changed to *Include*

The Uses relationship line (specified in earlier versions of UML) has been changed to Include, as specified in the most recent versions of UML (versions 1.3 and 1.4).

Behavior in SA/2001 v7	**Behavior in V8**
Uses relationship line could be drawn between Use Cases to show inclusion.	Include relationship can be drawn between Use Cases to show inclusion.
What the Conversion Does	
Any Uses relationship drawn between Use Cases is replaced by the Include relationship.	

Representational Consistency

Representational consistency has been implemented for the Include relationship line, which is drawn between Use Cases on a Use Case diagram.

Behavior in SA/2001 v7	**Behavior in V8**
Uses relationship line could be drawn between Use Case A and Use Case B on one diagram, yet on another diagram, the same two Use Cases could be drawn with no relationship, or a different type of relationship, drawn between them.	If an Include relationship line is drawn between Use Case A and Use Case B on one diagram, and those same two Use Cases are drawn on another diagram, the same Include relationship is automatically drawn between them.
What the Conversion Does	
Represents the same Include relationship line between a pair of Use Cases wherever the pair is drawn.	

Recursion of Includes (was Uses)

There is nothing under the old UML processing to prevent a user from expressing recursive usage in Use Cases. So when in conversion such recursion is discovered, one of the Uses lines is arbitrarily “discarded” so that the recursion is avoided.

Behavior in SA/2001 v7	**Behavior in V8**
Uses relationship line could be drawn from Use Case A to Use Case B on one diagram, yet on another diagram, the Uses relationship could be drawn in the opposite direction – from Use Case B to Use Case A. This is illegal recursion.	An Include relationship drawn from Use Case A to Use Case B will be consistently represented everywhere that pair of Use Cases appears. If the two Use Cases are drawn on a diagram, the relationship is automatically drawn.
What the Conversion Does	
Because previous versions of System Architect did not enforce representational consistency amongst UML models, it was possible for the user to model inconsistent information – for example, model that Use Case A uses Use Case B on one diagram, and model that Use Case B uses Use Case A on another diagram. The conversion will only represent one of these two scenarios in the new encyclopedia (the first one encountered). The conversion will also present the user with warning information that this has occurred. If this is not the way the user wishes to model the information, it is up to the user to adjust the converted encyclopedia, or adjust the old encyclopedia prior to conversion to eliminate inconsistencies. **WARNING:** When an Includes is added, it is checked to ensure that no recursive relationship between the involved Use Cases results. If it does, the addition is rejected – the Uses line is discarded – and a warning message is generated: **W2013** - Illegal recursion encountered. <type> < qualified name> will be deleted.	

Use Case *Extends* Relationship

The Extends relationship line now provides an Extension Point property. Representational consistency is now provided for Extends lines.

Behavior in SA/2001 v7	**Behavior in V8**
Extends relationship line could be drawn between Use Cases to show extension. No extension point could be specified on the line.	Include relationship can be drawn between Use Cases to show inclusion. Extension point can now be specified on the line.
What the Conversion Does	
No conversion necessary.	

Representational Consistency

Representational consistency has been implemented for the Extends relationship line, which is drawn between Use Cases on a Use Case diagram.

Behavior in SA/2001 v7	**Behavior in V8**
Extends relationship line could be drawn between Use Case A and Use Case B on one diagram, yet on another diagram, the same two Use Cases could be drawn with no relationship, or a different type of relationship, drawn between them.	If an Extends relationship line is drawn between Use Case A and Use Case B on one diagram, and those same two Use Cases are drawn on another diagram, the same Extends relationship is automatically drawn between them.
What the Conversion Does	
Represents the same Extends relationship line between a pair of Use Cases wherever the pair is drawn. Checks for recursion (see below).	

Extends Recursion

There is nothing under the old UML processing to prevent a user from expressing recursive extension in Use Cases. So when in conversion such recursion is discovered, one of the Extends lines is arbitrarily “discarded” so that the recursion is avoided.

Behavior in SA/2001 v7	**Behavior in V8**
Extends relationship line could be drawn from Use Case A to Use Case B on one diagram, yet on another diagram, the Extends relationship could be drawn in the opposite direction – from Use Case B to Use Case A. This is illegal recursion.	An Extends relationship drawn from Use Case A to Use Case B will be consistently represented everywhere that the pair of Use Cases appears. If the two Use Cases are drawn on a diagram, the relationship is automatically drawn.
What the Conversion Does	
Because previous versions of System Architect did not enforce representational consistency amongst UML models, it was possible for the user to model inconsistent information – for example, model that Use Case A extends Use Case B on one diagram, and model that Use Case B extends Use Case A on another diagram. The conversion will only represent one of these two scenarios in the new encyclopedia (the first one encountered). The conversion will also present the user with warning information that this has occurred. If this is not the way the user wishes to model the information, it is up to the user to adjust the converted encyclopedia, or adjust the old encyclopedia prior to conversion to eliminate inconsistencies. **WARNING:** When an Extends is added, it is checked to ensure that no recursive relationship between the involved Use Cases results. If it does, the addition is rejected – the Extends line is discarded – and a warning message is generated: **W2013** - Illegal recursion encountered. <type> < qualified name> will be deleted.	

Use Case Generalization Relationship

Behavior in SA/2001 v7	Behavior in V8
Generalization relationship between Use Cases not supported.	The Generalization relationship, drawn between Use Cases, is now supported in System Architect V8.
What the Conversion Does	
No conversion necessary.	

Class Diagram

There have been significant changes to the Class Diagram and the symbols and definitions contained within it.

- The name of the diagram type has been changed from **UML Class** to **Class**.
- Classes and their associated definitions are now contained within a package definition, which provides them with a namespace. Class diagrams are also contained within a package, and represented thusly within the UML tab of the browser.
- Representational consistency is maintained for relationships between classes and other classifiers drawn on a Class diagram.
- The manner in which associations are handled has been significantly changed. Much of the information about an association has been moved from its ‘Associative’ dialog to the definition dialog. The association definition provides information about each association end – one for each end of the class it attaches. So a normal association between two classes contains two association ends (is a binary association).
- N-ary Associations: The definition of an association between three classes (ternary association) contains three association ends, the defintion of an n-ary association contains n association ends. An n-ary association is now formed by drawing an association from a class to an existing association. Because of this, the toolbar no longer includes the N-ary Association symbol.

Packages on Class Diagrams

In general, a Package property has been added to a Class diagram. The class diagram ‘belongs’ to this package, and is displayed under it in the UML tab of the browser.

Diagrams and Definitions Placed in Package1 During Conversion

Behavior in SA/2001 v7	Behavior in V8
Packages could be drawn on Class diagrams, but they did not provide ownership of any UML classifiers such as classes. They were simply diagrammatic symbols.	A Package property has been added to the Class diagram definition. Within the UML tab of the browser, the diagram is displayed under the package it ‘belongs to’. Classes, and their constituant properties (ie, attributes, methods, etc) are ‘keyed’ to a package. Package provides a ‘namespace.
What the Conversion Does	
On conversion, System Architect V8 places all class diagrams and all contained symbols and definitions within a package named Package1. Users may rename the default package in the converted encyclopedia, and they may also move diagrams and definitions to a different package. Diagrams may be moved by simply changing the package name in the diagram’s properties. Definitions may be moved by using the browser to copy them, paste them into the different package, and then deleting the old definition.	

Symbols Drawn Within a Package Symbol

Behavior in SA/2001 v7	Behavior in V8
Users can draw a package symbol on a diagram workspace, enlarge its area by dragging on its handlebars, and draw other symbols such as classes, etc, within it. Drawing symbols within a package symbol did not provide any special semantic meaning.	Users can still draw symbols within a package like they did in earlier versions. Similarly, System Architect V8 still does not recognize such drawing to mean the semantic of ownership. Thus, if a class is moved inside or outside of a Package, System Architect does not rekey that class or, indeed, make any semantic change to the user's model.
What the Conversion Does	
If a Package symbol had been drawn on a class diagram, it is retained on that diagram during conversion. Symbols drawn within the package are also retained. However, the symbols do not get 'keyed' to the package they are drawn within. The package symbol drawn on the diagram becomes part of the default Package1.	

Class Stereotypes

User-Defined Depictions Based on Sterotype Value

New functionality is provided in System Architect V8 which allows the user to dictate how a symbol is drawn based on a property within that symbol's definition. For UML classes, this property is stereotype. This functionality is usable in other contexts also (for example, you may specify how an elementary business process symbol is drawn based on some property).

The new functionality is enacted as follows:

1. An optional extension to any entry in a List in the properties files (SAPROPS.CFG and USRPROPS.TXT) identifies to SA how to draw symbols whose definitions have that value for the appropriate property. E.g. in UML, the List "Class Stereotypes" against the entry for "actor", the user indicates how to draw an actor (which is any Class whose stereotype is "actor") by identifying the metafile to be used.
2. Whenever SA is drawing a symbol whose definition has a property whose value is taken from such a list and the property's value has such a meta-file indicated, SA utilizes that meta-file provided the user has indicated that this special drawing mechanism is to be used for that symbol.
3. SA "inflates" its toolbox and "Draw" menu to display multiple instances of a symbol type in this category to accommodate all the different applicable meta-files.
4. As was done for Stereotype for Classes in the old UML support, symbols "dropped" on a diagram by this mechanism have the equivalent symbol property set to the value implied by the drawing tool utilized. Again like Class Stereotype, this value is then copied into the definition.
5. A definition may have no more than one property whose value controls how symbols are to be drawn.
6. The prior mechanism for drawing the various "flavors" of Class has been discarded as has all the differing symbol types for these flavors – except, of course, the basic Class symbol.

When a symbol is drawn in this special manner, SA assumes that all text is "outside".

Example

To enable this function, the **Depictions** clause is used within a **List** in usrprops.txt. For example, here is an example of how a class stereotype is handled in saprops.cfg:

```
List "Class Stereotypes"
{

Value "actor"depictions {diagram images\slctact.wmf
menu images\slctact.bmp}
…
}
```

An **Images** folder within every encyclopedia now houses .bmp's or .wmf's to support this function. Out of the box, all class symbols are now handled this way.

User-Defined Class Depictions Based on Stereotype

Behavior in SA/2001 v7	Behavior in V8
Classes with certain stereotypes were depicted differently on the diagram workspace. This functionality was hard-coded in the product.	All classes are depicted in the toolbar by a user-defined .bmp file and on the diagram workspace by a user-defined .wmf file. These picture files are housed in a new directory located within each encyclopedia, named **Images**.
What the Conversion Does	
Images folder is provided in the converted encyclopedia; classes are converted to be represented by the new regime.	

Template and Parameterized Classes

In earlier versions of System Architect, the parameterized class was treated differently than a regular class – it had a separate symbol on the toolbar, and once drawn, was known as a parameterized class symbol rather than a class symbol. In the new version of System Architect, V8, all classes are now treated simply as different stereotypes of class.

Behavior in SA/2001 v7	Behavior in V8
Parameterized Class was a separate symbol on the diagram toolbar. Once drawn, it appeared as a class with a little dotted rectangle on the top right of the main rectangle. Other special types of classes -- Class Template, Instantiated Class, Class Utility, Parameterized Class Utility and Instantiated Class Utility – were drawn by drawing a class and changing the stereotype to the applicable value.	Parameterized Class (Class Template), Instantiated Class, Class Utility, Parameterized Class Utility and Instantiated Class Utility are now all drawn by selecting the class symbol and specifying the appropriate stereotype.
What the Conversion Does	
Parameterized Class (Class Template), Instantiated Class, Class Utility, Parameterized Class Utility and Instantiated Class Utility are all treated as different stereotypes of class. However, Classes whose stereotypes include the word “parameterized” have the little dotted rectangle on the top right of their main rectangle drawn for them.	

Associations on Class Diagrams

A number of changes have been made to the handling of associations on class diagrams:

- Representational consistency is now maintained for associations on an encyclopedia level.
- The name of an association's definition has been changed from Instance to Association.
- Most properties describing an association have been moved from the Associative dialog to the definition dialog of an association.
- The association definition is now composed of the Association End definitions of the classes the association connects -- each Association End identifies the Class which is fulfilling that end's role.
- The user is allowed to associate three or more Classes by first associating two Classes and then associated additional Classes to that Association.
- The definition type behind symbols of type N-ary Association has been changed from Class to Association (which has been "renamed" from Instance).
- The key of an Association comprises Name and a GUID called the "Association GUID". The Name may be omitted. (The Association GUID should not be confused with the GUID – a property that is added to all definitions and diagrams during conversion as a unique universal identifier. The GUID used in the key of an Association is in addition to the GUID assigned as a universal identifier.)
- The key to an Association End is the Package and Name of the Class fulfilling the Role, the Name and GUID of the Association and the Name and Role GUID of the Association End.

Representational Consistency

Associations drawn between classes are now maintained within the encyclopedia; they are no longer diagram-specific. In layman's terms, what this means is that if you draw an association between class A and class B on one diagram, then create a new diagram within the same package, and drop down class A and class B, the same association will be drawn between them. System Architect ensures that diagrams are adjusted when opened and at other synchronizing events to reflect the addition and deletion of UML classes and associations.

Behavior in SA/2001 v7	Behavior in V8
Association line could be drawn between class A and class B on one diagram, yet on another diagram, the same classes could be drawn with no relationship drawn between them.	If an Association line is drawn between class A and class B on one diagram, and the same classes are drawn on another diagram, the same Association line is automatically drawn between them.
What the Conversion Does	
Represents the same Association line between two classes wherever the combination are drawn.	

Association Definition

Behavior in SA/2001 v7	Behavior in V8
Association definition was called Instance. Most information about an association, however, was held in the Associative dialog.	Association definition is now called Association. The Association definition contains the Association End definitions – there is one Association End definition for each class the association connects (most associations are binary).
What the Conversion Does	
Information about an association is moved from the Associative dialog to the respective Association End definition.	

N-ary Definitions

Behavior in SA/2001 v7	Behavior in V8
User selected three or more classes and chose the Relate with N-ary command from the Draw menu. The classes all became connected. An n-ary diamond symbol was drawn at the junction of the lines connecting the classes. The lines connecting the classes were association lines.	User creates an association between two classes, then draws an association line from a third class to the association line between the first two classes. The association lines connect, and an n-ary diamond symbol is automatically drawn. The definition of the diamond symbol, or any junction on the line is the same – it is a single association with three (or more) association ends.
What the Conversion Does	
The definition of an N-ary has been changed from Class to Association. During conversion, SA creates a Class containing that material that was in the old N-ary and makes that Class the Association Class of the Association. Two possible problems must be considered which arise because the UML Methodology requires the Class and the Association to have the same name. • The original N-ary might not have been named, but SA cannot allow the Class to not have a name. In this case SA assigns a name. • The original N-ary may have had the same name as an extant Class. In this case also, SA assigns a name.	

N-ary Association Lines

Behavior in SA/2001 v7	Behavior in V8
Under the old UML support, the diamond-shaped N-ary symbol (which was defined as a class) was connected to its classes by Association lines.	The definition of the diamond symbol, or any junction connecting it to a class is the same – it is defined as a single association with three (or more) association ends.
What the Conversion Does	
Conversion turns every Association line that connected an N-ary symbol to a class, into an Association Role line, which is a new symbol type. The Association Role line is given an Association Role definition, which is a new definition type. A warning message is generated: **W2019** - The <line type> line named <line name> on <type> diagram <name> was incorrectly connected to the <node type> <node name> and the <node type> <node name>. It will be removed.	

Multiple N-ary Association Configurations

Behavior in SA/2001 v7	Behavior in V8
Under the old UML support, an N-ary (defined by a Class) could be use to associate various configurations of Classes. On one diagram, it could have been used to associate Classes A, B and C and on another (or even, conceivably, on the same diagram!) it could have been used to associate Classes D, E, F and G. Furthermore, the user might have used that same Class as the Association Class for one or more binary (ordinary) Associations. This was a consequence of not enforcing representational consistency.	This behavior cannot be modeled anymore because of representational consistency.
What the Conversion Does	
When conversion encounters multiple "configurations"[1] like this, it copies the original Class as many times as needed such that each original configuration has its "own" Class. Naturally, the copies have modified names to ensure uniqueness.	

Link Attributes Linked to Binary (Ordinary) Associations

[1] In this context, a "configuration" is an unordered set of associated Classes. Thus, if Class X is used (as the definition of an N-ary) in one place to associate Classes A, B and C and in another place it associates the same three Classes (regardless of apparent order), that is considered the same configuration. It is the same configuration even if some of the associative data, such as Cardinality (Multiplicity), differ. If, however, that same N-ary is used to associate Classes A, B, C and D or Classes A, B and D or **any** collection of Classes other than A, B and C, then that is deemed a different configuration.

Behavior in SA/2001 v7	**Behavior in V8**
A link attribute symbol on the toolbar enabled a user to draw a link attribute symbol on the diagram. The user could then select this symbol, select an association line, and choose an attach command in the Draw menu to attach the link attribute to the association.	Link Attributes are really only special cases of Association Classes.
What the Conversion Does	
Conversion converts Link Attributes to Classes containing its Class Attributes and makes the new class the Association Class of the Association. A warning message is generated: **W2014** - Link Attributes are no longer supported. Link Attribute < name> will be transformed into a Class. If no Class already exists with the same name as the Link Attribute, the new Class is given that name. If the Link Attribute was not named, a new name is allocated for the class created. If such a Class already exists, a new name is allocated and a message is generated: **W2012** - Class <name 1> already exists. Conversion will create Class < name 2> for Link Attribute <name 1>. The Class will be represented on the diagram in place of the Link Attribute and will be linked to its Association.	

Link Attributes Linked to N-ary (Diamond) Symbol

Behavior in SA/2001 v7	Behavior in V8
Under the old UML support, System Architect allowed Link Attributes to be linked to the N-ary (diamond shaped) symbol. Although SA allowed this, it did not make much sense because the N-ary was a Class and so had no need for a Link Attribute.	Link Attributes are only special cases of Association Classes.
What the Conversion Does	
Conversion unlinks the Link Attribute and generates a warning message:	
W2017 - The <symbol type> <symbol name> on <diagram type> diagram <diagram name> is improperly linked. The link will be broken.	
Conversion also converts the link attribute to a Class (see above) and emits a warning message:	
W2014 - Link Attributes are no longer supported. Link Attribute < name> will be transformed into a Class.	

Link Attributes Attached to N-ary Associations

Behavior in SA/2001 v7	**Behavior in V8**
Under the old UML support, Association lines connected the associated classes to the N-ary (diamond) symbol. System Architect allowed the user to link Link Attributes to those Association lines, even though this did not make semantic sense.	Incorrect behavior can no longer be drawn.
What the Conversion Does	
When this condition is encountered, conversion unlinks the Link Attribute (which is being converted to a Class – see above.) and emits a warning message: **W2017** - The <symbol type> <symbol name> on <diagram type> diagram <diagram name> is improperly linked. The link will be broken. Conversion also converts the link attribute to a Class (see above) and emits a warning message: **W2014** - Link Attributes are no longer supported. Link Attribute < name> will be transformed into a Class.	

Classes Attached to N-ary Associations

Behavior in SA/2001 v7	Behavior in V8
Under the old UML support, Association lines connected the associated classes to the N-ary (diamond) symbol. System Architect allowed the user to link classes to those Association lines, even though this did not make semantic sense.	Incorrect behavior can no longer be drawn.
What the Conversion Does	
When this condition is encountered, conversion unlinks the class and emits a warning message: **W2017** - The <symbol type> <symbol name> on <diagram type> diagram <diagram name> is improperly linked. The link will be broken.	

Classes Linked to Multiple Associations

Behavior in SA/2001 v7	**Behavior in V8**
Under the old UML, SA allowed the user to link a Class (as the "Association Class") to one Association and also link it to another (either on a different diagram or on the same diagram using a different Class symbol for the Class). The user may also have used it as the definition for one or more N-aries. This makes no sense.	Incorrect behavior can no longer be drawn.
What the Conversion Does	
During conversion, every time a Class is discovered linked (or used as an N-ary) in more than one configuration of associated Classes, copies of that Class are taken to fulfill every different configuration. Symbol names of the Classes are changed as needed and a warning message is emitted: **W2015** - Class < name> is linked in more than one configuration. Copies of it will be created.	

An Association with Multiple Linkages

Behavior in SA/2001 v7	Behavior in V8
Under the old UML, the user could link a Link Attribute to an Association and could link a Class (as the "Association Class") to that same Association – albeit on a different diagram or, if on the same diagram, to a different Association line for the Association. Indeed, (s)he might have linked more than one Class and/or more than one Link Attribute to the same Association.	Incorrect behavior can no longer be drawn.

What the Conversion Does

Conversion does the following:

- If an Association is encountered with more than one linked Class, one of the Classes (arbitrarily) is taken as the future Association Class and the remainder are treated as though they were not linked and a message is emitted.

  W2018 - Association <association name> on <diagram type> diagram <diagram name> had <node type> <node name> linked to it in contention with an earlier encountered linked item. The link will be broken.

- If an Association is encountered with more than one linked Link Attribute and no linked Class, one of the Link Attributes (arbitrarily) is used to provide the name of the future Association Class. The contents of **all** the Link Attributes are appended to this new Class and a message is emitted.
- If an Association is encountered with both a Class linked as the Association Class and one or more Link Attributes linked then the contents of the Link Attributes (i.e. their Class Attributes) are appended to those of the Class.

Inheritance on Class Diagram

Rename Is Subclass Relationship to Inherits From

Behavior in SA/2001 v7	Behavior in V8
Is Subclass line drawn from subclass to superclass.	*Inherits From* line drawn from subclass to superclass.
What the Conversion Does	
All *Is Subclass* lines are replaced with *Inherits From* line.	

Representational Consistency

Behavior in SA/2001 v7	Behavior in V8
Is Subclass relationship line could be drawn between class A and class B on one diagram, yet on another diagram, the same two classes could be drawn with no relationship, or a different type of relationship, drawn between them.	If an Inherits From relationship line is drawn between class A and class B on one diagram, and those same two classes are drawn on another diagram, the same Inherits From relationship is automatically drawn between them.
What the Conversion Does	
Represents the same Inherits From relationship line between a pair of classes wherever the pair is drawn. Checks for recursion (see below).	

Recursion of Inheritance/'Is Subclass'

There is nothing under the old UML processing to prevent a user from expressing recursive inheritance in Classes. So when in conversion such recursion is discovered, one of the Is Subclass lines is arbitrarily "discarded" so that the recursion is avoided.

Behavior in SA/2001 v7	**Behavior in V8**
Is Subclass relationship line could be drawn from class A to class B on one diagram, yet on another diagram, the *Is Subclass* relationship could be drawn in the opposite direction – from class B to class A. This is illegal recursion.	An *Inherits From* relationship drawn from class A to class B will be consistently represented everywhere that the pair of classes appears. If the two classes are drawn on a diagram, the relationship is automatically drawn.
What the Conversion Does	
Because previous versions of System Architect did not enforce representational consistency amongst UML models, it was possible for the user to model inconsistent information – for example, model that class A inherits from class B on one diagram, and model that class B inherits from class A on another diagram. The conversion will only represent one of these two scenarios in the new encyclopedia (the first one encountered). The conversion will also present the user with warning information that this has occurred. If this is not the way the user wishes to model the information, it is up to the user to adjust the converted encyclopedia, or adjust the old encyclopedia prior to conversion to eliminate inconsistencies. **WARNING:** When an Inheritance is added, it is checked to ensure that no recursive relationship between the involved classes results. If it does, the addition is rejected – the Inherits line is discarded – and a warning message is generated: **W2013** - Illegal recursion encountered. <type> < qualified name> will be deleted.	

Realizes Relationship on Class Diagram

Representational Consistency

Behavior in SA/2001 v7	Behavior in V8
Realizes relationship line could be drawn between class A and class B on one diagram, yet on another diagram, the same two classes could be drawn with no relationship, or a different type of relationship, drawn between them.	If a Realizes relationship line is drawn between class A and class B on one diagram, and those same two classes are drawn on another diagram, the same Realizes relationship is automatically drawn between them.
What the Conversion Does	
Represents the same Realizes relationship line between a pair of classes wherever the pair is drawn. Whenever conversion adds a Realizes line between two classes (because they are connected via a Realizes line on another diagram), it generates a warning message: **W2020** - Realizes < name> is connected in more than one configuration. Copies of it will be created. Conversion also checks for recursion (see below).	

Realizes Recursion

There is nothing under the old UML processing to prevent a user from expressing recursive realization in Classes. So when in conversion such recursion is discovered, one of the Is Subclass lines is arbitrarily "discarded" so that the recursion is avoided.

Behavior in SA/2001 v7	Behavior in V8
Realizes relationship line could be drawn from class A to class B on one diagram, yet on another diagram, the *Realizes* relationship could be drawn in the opposite direction – from class B to class A. This is illegal recursion.	A *Realizes* relationship drawn from class A to class B will be consistently represented everywhere that the pair of classes appears. If the two classes are drawn on a diagram, the relationship is automatically drawn.
What the Conversion Does	
Because previous versions of System Architect did not enforce representational consistency amongst UML models, it was possible for the user to model inconsistent information – for example, model that class A *realizes* class B on one diagram, and model that class B *realizes* class A on another diagram. The conversion will only represent one of these two scenarios in the new encyclopedia (the first one encountered). The conversion will also present the user with warning information that this has occurred. If this is not the way the user wishes to model the information, it is up to the user to adjust the converted encyclopedia, or adjust the old encyclopedia prior to conversion to eliminate inconsistencies. **WARNING:** When a Realizes is added, it is checked to ensure that no recursive relationship between the involved classes results. If it does, the addition is rejected – the *Realizes* line is discarded – and a warning message is generated: **W2013** - Illegal recursion encountered. <type> < qualified name> will be deleted.	

State Diagram

The following changes have been made to the UML State diagram:

- The diagram type name has been changed from **UML State** diagram to **State** diagram.
- A Package property has been added to the State diagram's definition.
- Certain properties of a state definition have changed:

  OMT Process has been changed to OO Action

  An OO Action now has the following properties: When (On entry, On exit, Do) and Type (Action, Send Event). In previous versions, On Entry and On Exit were simple text properties within a state definition.

  Lists of State Variables, OO Actions, and Internal Transitions have been put into grids.
- Certain properties of a state transition have changed:

  OO Transition now has a list of OO Actions, labeled Effect

  OO Transition now has properties Trigger Event and Parameters.

  OO Transition's Condition has been renamed to Guard Condition.

  OO Transition now has properties Send Event, Send Parameters, and Send Target.
- Four new symbols have been added to the diagram. They are the **Synchronization Bar**, **Junction, Synch State** and **Dynamic Choice Point**. The **And Connector** has been removed. All of the new symbols are defined by **OO State**.

Sequence Diagram

A number of changes have been made to the UML Sequence diagram:

- The name of the diagram type was changed from UML Sequence Diagram to Sequence Diagram.
- A Package property has been added to the diagram definition.
- The diagram is synchronized with its partner Collaboration diagram only when the user invokes a "Synchronize" menu entry. Consequently, the Synchronization property of this diagram became redundant, and was removed.
- The manner in which Foci of Control are dealt with has been completely changed. There is now a separate symbol for Focus of Control over which the user has total control. They may be "stacked".
- Event line on the Sequence diagram has been renamed to Message/Stimulus line.

Focus of Control

Behavior in SA/2001 v7	Behavior in V8
Previously in System Architect, focus of control bars (or activation bars) were automatically drawn on the object lifelines based on the placement of event lines drawn to the object lifeline. The user had little or no control over the length or position of the line.	In System Architect V8, the focus of control symbol is now a separate symbol that a user may select from the toolbar and attach to an object lifeline. The user may increase or decrease the length of the line as he or she sees fit.
What the Conversion Does	
Conversion maintains all focus of control bars that existed on the old Sequence diagram. Users may find multiple focus of control bars on object lifelines in the converted encyclopedia, and may want to delete some of these bars and lengthen one bar to compensate.	

Symbol Properties Relocated to Their Definitions

A number of properties that had been housed in the Symbol properties tab of the object lifeline's definition have been moved to the definition of the object itself. What this means is that these properties are now held globally for the definition of the object, not on a symbol-by-symbol basis.

Behavior in SA/2001 v7	**Behavior in V8**
The following properties were housed in an object’s symbol properties tab: • Create Action • Destroy Action	The following properties are now housed in an object’s definition: • Create Action • Destroy Action
What the Conversion Does	
Appropriate values of the properties are migrated to the respective property within the link’s definition.	

Collaboration Diagram

A number of changes were made to the UML Collaboration diagram:

- The name of the diagram was changed from **UML Collaboration** diagram to **Collaboration** diagram.
- A Package property has been added to the diagram's definition.
- The diagram is synchronized with its partner Sequence diagram only when the user invokes the "Synchronize" menu entry. Consequently, the Synchronization property of this diagram was removed.

Symbol Properties Relocated to Their Definitions

A number of properties that had been housed in the Symbol properties tab of the Link line's definition have been moved to the definition of the line itself. What this means is that these properties are now held globally for the definition of the link line, not on a symbol-by-symbol basis.

Behavior in SA/2001 v7	Behavior in V8
The following properties were housed in a **link** line's symbol properties tab: • From Visbility • From Shared • To Visibility • To Shared	The following properties are housed in a **link** line's definition: • From Visbility • From Shared • To Visibility • To Shared
What the Conversion Does	
Appropriate values of the properties are migrated to the respective property within the link's definition.	

Activity Diagram

A number of changes were made to the UML Activity diagram:

- The name of the diagram was changed from **UML Activity** diagram to **Activity** diagram.

Component Diagram

The following changes have been made to the UML Component diagram:

- The name of the diagram has been changed from UML Component diagram to Component diagram.
- A Package property has been added to the diagram's definition. This package property enables a user to specify what package the diagram is contained in.

Packages on Component Diagrams

In general, a Package property has been added to a Component diagram. The Component diagram ‘belongs’ to a package specified in this property, and is displayed under it in the UML tab of the browser.

Diagrams and Definitions Placed in Package1 During Conversion

Behavior in SA/2001 v7	**Behavior in V8**
Packages could be drawn on Component diagrams, but they did not provide ownership of any UML classifiers such as classes or components. They were simply diagrammatic symbols.	A Package property has been added to the Component diagram definition. Within the UML tab of the browser, the diagram is displayed under the package it ‘belongs to’. Of the UML classifiers that can be drawn on a Component diagram, components are not ‘keyed’ to package, classes are.
What the Conversion Does	
On conversion, System Architect V8 places all component diagrams and all contained symbols and definitions within a package named Package1. After conversion, users may rename the default package in the converted encyclopedia, and they may also move diagrams and definitions to a different package. Diagrams may be moved by simply changing the package name in the diagram’s properties. Class definitions may be moved by using the browser to copy them, paste them into the different package, and then deleting the old definition.	

Symbols Drawn Within a Package Symbol

Behavior in SA/2001 v7	Behavior in V8
Users could draw a package symbol on a diagram workspace, enlarge its area by dragging on its handlebars, and draw other symbols such as components, interfaces, etc, within it. Drawing symbols within a package symbol did not provide any special semantic meaning.	Users can still draw symbols within a package like they did in earlier versions. Similarly, System Architect V8 still does not recognize such drawing to mean semantic ownership. Thus, if an interface is moved inside or outside of a Package, System Architect does not rekey that interface or, indeed, make any semantic change to the user's model.
What the Conversion Does	
If a Package symbol had been drawn on a class diagram, it is retained on that diagram during conversion. Symbols drawn within the package are also retained. However, the symbols do not get 'keyed' to the package they are drawn within. The package symbol drawn on the diagram becomes part of the default Package1.	

Deployment Diagram

The following changes have been made to the UML Component diagram:

- The name of the diagram has been changed from UML Deployment diagram to Deployment diagram.
- A Package property has been added to the diagram's definition. This package property enables a user to specify what package the diagram is contained in.

Class Definition

There have been a number of changes to elements of a class definition:

- The way that method parameters is dealt with has been significantly changed.
- Constructors are now a special form of a method.
- The way that language properties are dealt with is significantly different.

Method Parameters

Methods and Formal Parameters

Under the old Methodology, Methods (and Constructors) were keyed by their Formal Parameters. The Formal Parameters property was simply a text property, which the user entered. This created difficulties because minor differences in the manner in which Formal Parameters was typed (say, more or less spacing) gave the appearance of differing parameterization when no such difference was intended. To deal with this problem and to support the planned future forward and reverse engineering, a different regime is to be employed. Under the new UML support, the Formal Parameters property will still be a key to Method, but the user will longer enter it. Instead, during normal operation of System Architect, the user will maintain a new property called Parameters, which will be a ListOf a new definition called Formal Parameter. System Architect will then construct the Formal Parameters property from the list property.

When a Method is encountered during conversion, the Formal Parameters property is analyzed and the Parameters property is constructed. The necessary Formal Parameter definitions are created too. However, no attempt is made to rekey the Method by changing the property Formal Parameters, if such a change is implied by the property Parameters. Any such “re-keying” will happen automatically next time the user edits the Method. Conversion is not undertaking the re-keying because it is expected that frequently the new key would clash with that of an existing Method for the reason stated earlier.

Method Parameterization

Behavior in SA/2001 v7	**Behavior in V8**
In previous versions of System Architect, users added parameters to a method by typing in a text string, with each parameter separated by commas.	There is a new definition type named Formal Parameter. Users now add parameters to a method by selecting from or adding to a list of Formal Parameters.
What the Conversion Does	
The conversion takes each value found within the text string of a method’s parameters and create a new Method Parameter definition for each, assigning it to the method.	

The Demise of Constructors

Constructors are not supported as a separate type in the new meta-model. (Constructors are simply special cases of Methods that must have the same name as the Class they belong to.) When a Class is encountered in conversion, its two list properties “Operations” and “Constructors” are merged. Duplication could arise when a Method has been given a Name equal to that of the Class and has its Formal Parameters equal to that of an existing Constructor. Under these circumstances, the Method is retained, augmented by the Constructor’s properties being appended the Method’s properties.

Method Constructors

Behavior in SA/2001 v7	**Behavior in V8**
Methods and constructors were supported as separate types in a class definition.	Constructors are not supported as a separate type in the new meta-model. (Constructors are simply special cases of Methods that must have the same name as the Class they belong to.)
What the Conversion Does	
When a Class is encountered in conversion, its two list properties “Operations” and “Constructors” are merged. **Warning:** Duplication could arise when a Method has been given a Name equal to that of the Class and has its Formal Parameters equal to that of an existing Constructor. Under these circumstances, the Method is retained, augmented by the Constructor’s properties being appended the Method’s properties. A warning message is emitted: **W2011** - Constructor < key> collides with Method < key>. The constructor’s properties will be merged into the method.	

Language Properties

Language Properties

The way that class types and method return types are handled has been changed. Previously, System Architect provided a language-specific set of types – there was a list of C++ types, a list of Java types, a list of CORBA IDL types, etc. It was up to the user to intelligently specify the proper types for their classes – they weren't prevented from specifying a class attribute as having a Java type of Int, and a C++ type of char.

For V8 of System Architect, the following is true:

- A class is tied to a programming language
- A 'type' is actually represented as a class that has that type name (ie, a class named Integer)
- 'Type' classes are contained within a package for the programming language to which they belong. The programming language packages are contained within the Language Types package. There is a programming language package for Java, C++, Visual Basic, etc.

The conversion will map each class, attribute, and method to only one language type. The conversion handles each class, attribute, and method individually. If, for example, the conversion finds that a class has values for its C++ properties, and no values for other language properties, it will map the class to C++. If a class has values for its Java properties, and no values for other language properties, it will map the class to Java. If the class has values for more than one language, the conversion will use the language property set in the Customize Method Support of the existing encyclopedia as the arbitrator. If more than one language is set in the Customize Method Support of the existing encyclopedia, then the conversion arbitrarily chooses Java as the mapping language.

Enumerated Types

Behavior in SA/2001 v7	Behavior in V8
C++ Enumerated Type was a separate definition type. A class definition contained a property entitled C++ Enumerated Types.	The definition C++ Enumerated Type has been removed. The property of a class definition entitled C++ Enumerated Type has also been removed. The property Value List, which was part of the C++ Enumerated Type definition, has been added to the class definition.
What the Conversion Does	
Conversion creates a Class for each Enumerated Type and moves the value of the property Value List from the *Enumerated Type* to that *Class.* The *Enumerated Type* itself is not written forward.	

Programming Language for a Class

Behavior in SA/2001 v7	Behavior in V8
A user could specify multiple language properties for a single class.	A *Class* and its dependents are devoted to a single language at any point in time. Usually the property Programming Language is chosen by the user when the Class is first created, within the new property, Programming Language. The Programming Language can be specified as Java, CORBA, C++, or Visual Basic. Programming Language may also be set by F/R engineering according to the language it is dealing with. Once set, the user may not change the value of this property.
What the Conversion Does	
Specifies a single language for each class converted. Conversion looks to see if properties for only one language have been set for the class, and if this is so, it chooses this language as the language for the class.	

Class Attribute Types

Behavior in SA/2001 v7	**Behavior in V8**
Users could specify an attribute type value for each language that System Architect supported. Each language type was a separate property. It was up to the user to enforce standardization of types. However, there was nothing stopping the user from specifying inconsistent types for an attribute – for example, that an attribute had a C++ type of char and a Java type of int.	The different type related properties in the *Class Attribute* definition for different languages have been combined. Each class attribute has the following common type properties: • Type Pkg • Pre Type • Type • Post Type The following type properties that existed in previous versions of System Architect have been eliminated: • C Storage Type • CORBA Type • Delphi Type • Delphi Array Element Type • Delphi Set Type • JavaAttribute Type • PWB Data Type • VB Variable Type • Generic Data Type • Dynasty Type
What the Conversion Does	
Conversion sets the values of "Type Pkg", "Pre Type", "Type" and "Post Type" in accordance with the appropriate “<language> Type” property in the original and sets the value of “Type Pkg”. The appropriate “<language> Type” property and value for “Type Pkg” is determined by the logic shown in the table below.	

Value of “Programming Language” in Owning Class	**Property to be used**	**Value for “Type Pkg”**	**Convert or Move**
Java	“Java Attribute Type”	“Java Types”	Convert
C ++	“C Storage Type”	“C++ Types”	Convert
Visual Basic	“VB Variable Type” or “Generic Data Type” – depending on which is first found non-blank	“VB Types”	Convert
CORBA	“CORBA Type”	“CORBA IDL Types”	Move
Delphi	“Delphi Type” or “Delphi Array Element Type” or “Delphi Set Type” – depending on which is first found non-blank	“Delphi Types”	Move
PWB	“PWB Data Type”	“PWB Types”	Move
Dynasty	“Dynasty Type”	“Dynasty Types”	Move

If there is no value of “Programming Language” in the owning Class, a value will be determined (and set in the property) by the **Language Determining Algorithm** specified below:

Language Determining Algorithm: Each of the Class's Class Attributes is inspected in the order in which they appear in the Class. Each attribute's properties of the form "<language> Type" is then be inspected in order of the entries in the above table. As soon as one is encountered that is non-blank, that language is adopted as the Class's "Programming Language".

"Convert" means that the algorithm for analyzing a parameter value should be used to isolate the pre-type, type and post-type and the up to 3 appropriate values for "Pre Type", "Type" and "Post Type" inserted. "Move" means that the value is to be moved directly into "Type" and the properties "Pre Type" and "Post Type" not completed.

Method Return Types

Behavior in SA/2001 v7	Behavior in V8
Users could specify a method return type value for each language that System Architect supported. Each language type was a separate property. It was up to the user to enforce standardization of types. However, there was nothing stopping the user from specifying inconsistent return types for a method – for example, that a method had a C++ return type of char and a Java return type of int.	The different return type related properties in the *method* definition for different languages have been combined. Each method has the following common return type properties: • Type Pkg • Pre Type • Type • Post Type The following type properties that existed in previous versions of System Architect have been eliminated: • (C++) Return Type • CORBA Return Type • Delphi Return Type • JavaMethod Type • PWB Return Type • VB Return Type
What the Conversion Does	
Conversion sets the values of "Type Pkg", "Pre Type", "Type" and "Post Type" in accordance with the appropriate "<language> Type" property or "Return Type" property in the original and supplies a suitable value for "Type Pkg". The appropriate "<language> Type" property and value for "Type Pkg" is determined by the logic shown in the table below.	

Value of "Programming Language" in Owning Class	**Property to be used**	**Value for "Type Pkg"**	**Convert or Move**
Java	"Java Method"	"Java Types"	Convert
C ++	"Return Type"	"C++ Types"	Convert
Visual Basic	"VB Return"	"VB Types"	Convert
CORBA	"CORBA Type"	"CORBA IDL Types"	Move
Delphi	"Delphi Return Type"	"Delphi Types"	Move
PWB	"PWB Return"	"PWB Types"	Move
Dynasty	none	none	-

If there is no value of "Programming Language" in the owning Class, a value will be determined (and set in the property) by the **Language Determining Algorithm** below:

Language Determining Algorithm: Each of the Class's Class Attributes will be inspected in the order in which they appear in the Class. Each attribute's properties of the form "<language> Type" will then be inspected in order of the entries in the above table. As soon as one is encountered that is non-blank, that language will be adopted as the Class's "Programming Language".

"Convert" means that the algorithm for analyzing a parameter value should be used to isolate the pre-type, type and post-type and the up to 3 appropriate values for "Pre Type", "Type" and "Post Type" inserted. "Move" means that the value is to be moved directly into "Type" and the properties "Pre Type" and "Post Type" not completed.

Modifying SA/Word Reports

Changes must be made to SA/Word reports to make them work with V8. Most changes deal with renaming of Diagrams, Symbols, and Definitions. Changes also involve the retrieving of attribute types, method return types, and method parameters.

Diagram, Symbol, and Definition Name Changes

There were a number of name changes to UML-related diagrams, symbols, and definitions in System Architect V8. A complete list is provided on page 5-6. You should refer to this list and make the applicable name change to code within any SA/Word macro that calls out one of these diagrams, symbols, or definitions.

Example of Diagram Name Changes:

For example, if you try to run a report that used to get UML Class diagrams, you will receive a message that there are no class diagrams in this encyclopedia. You must go into the code of the macro and change references to "UML Class" to "Class". Similarly, "UML Sequence" is now "Sequence", "UML Use Case" is now "Use Case", etc.

Use Case Diagram Specific Changes

For Use Case Associations, beware that “Uses” is now “Includes” and “Communicates With” is now “Use Case Association”.

If you have a report that gets actors, you should change "Actor" to "Class" and from this you can get the property "StereoType" (for Actor, it returns "actor").

To retrieve objects related to Actor, reports used to use the following relationship numbers:

```
SAOpenRelatedObject(SymHdl, 8)
SAOpenRelatedObject(SymHdl, 7)
```

Reports should now use the following relationship numbers to retrieve objects related to Actor:

```
SAOpenRelatedObject(SymHdl, 6)
SAOpenRelatedObject(SymHdl, 9)
```

Sequence Diagram Specific Changes

To start, remember that if you have a report that gets the event line on a Sequence diagram you must change the symbol "Event" to "Message/Stimulus".

Method Names:

To get a method name on a Message/Stimulus line on a Sequence diagram, you simply get the Operation property of that line, and get the name of the method definition that populates that property, as the following code snippet shows:

```
Property = szGetProperty(ObjectHdl, "Operation")
HdlRet = SAOpenDefinition(stringDefType, Property)
Name = szGetProperty(HdlRet, "Name")
```

Method Formal Parameters:

Formal Parameters used to be property of a method containing a text string; it is now a property of a method that contains a list of the formal parameter definition. To get the formal parameters of a method on a Message/Stimulus line, use code similar to the following:

```
Property = szGetProperty(ObjectHdl, "Operation")
HdlRet = SAOpenDefinition(stringDefType,
stringDefName)
Property = szGetProperty(HdlRet, "Formal
Parameters")
```

Method Return Types:

If your report got the return type of a method on an event line, beware that the return type used to be language-specific, so that you were getting a Java return type or a C++ return type, etc. In V8 there is now only one return type. The following snippet of code shows how a return type specific to Java (JavaMethodType) used to be obtained (Old) and how it is now obtained (New):

Old:

```
MethodName = szGetMethodName(ObjHdl)
HdlRet = SAOpenRelatedObject(ObjHdl, RelType 7)
Name = szGetProperty(HdlRet3, "Name") until Name
= MethodName
```

```
JavaMethodType = szGetProperty(HdlRet3,
"JavaMethodType")
```

New:

```
Operation = szGetProperty(ObjHdl, "Operation")
HdlRet = SAOpenDefinition("Method", Operation)
Return Type = szGetProperty(ObjHdl, "Return Type")
```

Converting User-Defined Matrices

The Matrix Editor has undergone substantial changes in System Architect version 8.5. Because of this, User-Defined matrices created with earlier versions need to be updated using the procedures below. In part, the procedure requires that you customize a VBA macro and run it against your encyclopedia. Standard Matrices, which are shipped with System Architect, will be updated automatically once existing encyclopedias are opened.

This section provides the VBA macro code that you need to update your UDMs. By their nature, user-defined matrices vary from client to client. Therefore, System Architect cannot automatically update all UDMs. Instead, we provide the macro code to let you to update your matrices. System Architect cannot provide a function that updates them for you automatically.

What has changed?

The System Architect program files used to create matrices and to display them on the SA menu have changed. Consequently, older UDMs will not be recognized or displayed on the SA menu without performing the procedures described below.

In earlier versions, to create a user defined Matrix it was necessary to create a new Matrix definition in USRPROPS.TXT. The properties used in these older definitions have changed, so they must be migrated to their new equivalents. The translation of these is relatively simple. The following table illustrates the translation:

Old Property	**New Property**
Name	RowDefintions
Key Information	ColumnDefintions
Model (used in entity matrices)	ColumnDefintionsKey1

*** Note – some matrices create their definitions the other way around (i.e. Name = ColumnDefinition and Key Information=RowDefinition). You need to verify how the definitions were created before setting up the conversion macro.

Performing the Conversion

Before running the macro, you must prepare the encyclopedia by running the System Architect Encyclopedia conversion utility.

1. Click **Tools**, and select **Conversion**.
2. Open the updated encyclopedia.

Converting User-Defined Matrices

Conversion is a four-step process in which you perform the following steps:

Step 1: Backup your existing encyclopedia!

Step 2: Modify your definition properties in USRPROPS.TXT to reflect the new properties.

Step 3: Copy the property data from the old properties to the new properties.

Step 4: Modify the USRMATRX.XML file to recognize your matrix and display it on the menu.

Step 1

Ensure you have backed up the encyclopedia. This will save a great deal of time, should any of the following steps fail.

Step 2

The sample properties below show the old definition properties and the new definition properties to which they must be changed.

The Old Definition:

```
DEFINITION "UOB/ENTITY"
{
PROPERTY "Model"
{ KEY EDIT ONEOF "Project Data Model" RELATE BY "is part
of"
DEFAULT "Pool" READONLY }
PROPERTY "Description"
  { EDIT Text LENGTH 255 HELP "Appears in the cell of a
Matrix" }
PROPERTY "Key Information"
{ EDIT Text LENGTH 100 KEY READONLY }
PROPERTY "Intersection?"
{ EDIT Boolean LENGTH 1 }
}
```

The New Definition :

```
DEFINITION "UOB/ENTITY"
{
```

```
PROPERTY "RowDefinition"
{ KEY EDIT OneOf "Unit of Behavior" RELATE BY "is part
of" }
PROPERTY "ColumnDefinitionKey1"
{ KEY EDIT OneOf "Project Data Model" RELATE BY "nothing"
}
PROPERTY "ColumnDefinition"
{ KEY EDIT OneOf "Entity" RELATE BY "is part of"}
{KEYED BY { Model:ColumnDefinitionKey1, Name } }
PROPERTY "Description"
{ EDIT Text LENGTH 255 HELP "Appears in the cell of a
Matrix" }
}
PROPERTY "Intersection?"
{ EDIT Boolean LENGTH 1 }
}
```

Note that changing the USRPROPS.TXT file will NOT delete the properties from your encyclopedia. It will merely hide them from view.

If in the future you wish these properties to become visible again, simple add the properties removed by this translation.

Step 3

Having changed the property set to it is also necessary to copy the actual property data from the original property to the new property. This job is done by a simple piece of VBA code.

1. Cut and paste the attached Sample Code from the following section, into a VBA macro module.
2. For every Entity Matrix definition add the following line to the MatrixConversion Subroutine:

   ConvertEntityMatrix "DEFINITIONNAME"

3. For every Normal Matrix definition add the following line to the MatrixConversion Subroutine:

   ConvertMatrix "DEFINITIONNAME"

4. Examples of these lines can be seen, commented out, in the MatrixConversion Subroutine.
5. Run the macro ‘MatrixConversion’

On completion the message box “Completed” will be displayed. All Matrix property data should now have been copied across to the new property names.

Sample Code to Convert Matrix Definitions.

Copy the code below into a macro and modify accordingly.

```
Private Sub MatrixConversion()

    ConvertEntityMatrix "UOB/ENTITY", True
    ConvertEntityMatrix "EBP/ENTITY", True
    ConvertEntityMatrix "LOCATION/ENTITY", False
    ConvertMatrix "REQUIREMENT/PROCESS", False
    ConvertMatrix "LOCATION/APPLICATION", False
    ConvertMatrix "BUSINESS/ORGANIZATION", False
    ConvertMatrix "RECEIVER/SENDER", False
    ConvertMatrix "BEHAVIOR/STAKEHOLDER", False
    ConvertMatrix "CULTURE/STAKEHOLDER", False
    ConvertMatrix "EBP/ROLE", False
    ConvertMatrix "WORKGROUP/ROLE", False
    ConvertMatrix "DIMENSION/PRODUCT", False
    ConvertMatrix "ORGUNIT/ORGUNIT", False
    ConvertMatrix "UOB/ROLE", True
End Sub

Private Sub ConvertEntityMatrix(Name As String,
ReverseRow As Boolean)
    Dim sao As SAObjects
    Dim sad As Definition
    Set sao =
Application.Encyclopedia.GetFilteredDefinitions("",
GetDefinitionType(Name))
    sao.ReadAll
    For Each sad In sao
        sad.SetProperty "ColumnDefinitionKey1",
quote(sad.GetProperty("Model"))
        If Not ReverseRow Then
            sad.SetProperty "RowDefinition",
quote(sad.Name)
            sad.SetProperty "ColumnDefinition",
quote(sad.GetProperty("Key Information"))
        Else
            sad.SetProperty "RowDefinition",
quote(sad.GetProperty("Key Information"))
            sad.SetProperty "ColumnDefinition",
quote(sad.Name)
        End If
        If sad.GetProperty("Description") <> "" Then
            sad.SetProperty "Intersection?", "T"
        End If
        sad.Save
    Next
End Sub

Private Sub ConvertMatrix(Name As String, ReverseRow As
Boolean)
    Dim sao As SAObjects
    Dim sad As Definition
    Set sao =
Application.Encyclopedia.GetFilteredDefinitions("",
GetDefinitionType(Name))
```

```
    sao.ReadAll
    For Each sad In sao
        If Not ReverseRow Then
            sad.SetProperty "RowDefinition",
quote(sad.Name)
            sad.SetProperty "ColumnDefinition",
quote(sad.GetProperty("Key Information"))
        Else
            sad.SetProperty "RowDefinition",
quote(sad.GetProperty("Key Information"))
            sad.SetProperty "ColumnDefinition",
quote(sad.Name)
        End If
        If sad.GetProperty("Description") <> "" Then
            sad.SetProperty "Intersection?", "T"
        End If
        sad.Save
    Next
End Sub

Private Function quote(Str As String) As String
    Dim i As Long
    If Left(Str, 1) = Chr(34) And Right(Str, 1) = Chr(34)
Then
        ' assume string is quoted already
        quote = Str
    Else
        If InStr(Str, " ") <> 0 Or InStr(Str, ".") <> 0 _
            Or InStr(Str, "/") <> 0 Or InStr(Str, "+") <>
0 Then
            quote = Chr(34) & Str & Chr(34)
        Else
            quote = Str
        End If
    End If
End Function
```

Step 4

Follow the new Matrix editor help for adding a new Matrix to the USRMATRX.XML file.

Reference

This section provides advanced reference material for understanding how the conversion is undertaken.

Changes to the Property File

All the new types have default names beginning “New UML”. However, if you look at the new SAPROPS (version 8), you will see the following right near the beginning:

HIDE statements for old UML diagrams

Even though the new code supports the old UML in the same fashion as before, these statements effectively turn the old UML off. Later, we may turn it off internally – i.e. not rely on SAPROPS for this. However, at the moment, it is convenient to be able to run either old or new UML with the new code. The old UML can be run with the new code on a new (converted) encyclopedia by simply using an old (version 7) SAPROPS file instead of the new SAPROPS file.

RENAME statements for new UML diagrams

The new UML diagrams are then renamed to the names used in the UML methodology version 1.4. The renaming of the Use Case diagram is done only if UML is enabled. If it is, the original Use Case diagram is renamed to Jacobsen Use Case which frees up its name for use by the New UML Use Case diagram.

RENAME statements for old UML definitions

If UML is active, rename statements for most of the definitions used by the old UML are next. This frees up their names for use by the new UML definitions.

RENAME statements for new UML definitions

If UML is active, the new UML definitions are then renamed (mostly) to the names used in the UML methodology version 1.4. They mostly use the names vacated by the old UML definitions.

There are a few instances where the name used in the UML methodology version 1.4 could not be used because that would clash with another methodology.

RENAME statements for old UML symbols

If UML is active, rename statements for most of the symbols used by the old UML are next. It is not actually necessary to free up their names for use by the new UML symbols, because symbol type names need be unique only within diagram type. Thus, they are currently commented out and eventually may be removed entirely.

RENAME statements for new UML symbols

If UML is active, the new UML symbols are then renamed to the names used in the UML methodology version 1.4. They mostly use the names vacated by the old UML definitions.

After the renaming, the new SAPROPS is much as you would expect, except that all the old UML material has been excised and the new UML material has been added all together at the end.

When conversion runs, it does some basic checks on the property files. The file in the old encyclopedia directory is checked for conformance with what conversion expects the old UML meta-model to be. Similarly, the file in System Architect working directory is checked for conformance with what conversion expects the new meta-model to be. If a problem is found with either file and there is UML material to convert, conversion is abandoned.

Removed Material

Some of the definition types that are used to support the (old) UML methodology were usable in one or more other OO methodologies as well, e.g. Class and Use Case whereas some types were exclusive to UML. Conversion removes the latter, which comprises:

- **Activity Model**
- **Action State**
- **Activity State Variable**
- **Action Transition**
- **Component**
- **Node**
- **Object State**
- **Realizes**
- **UML Communicates**
- **UML Dependency**
- **UML Event**
- **UML Extends**
- **UML Package**
- **UML Uses**

New Material

Every UML diagram found in the encyclopedia will have a new equivalent diagram created for it. The old one will be deleted. Usually the material on a new diagram will look just like that on the old, but there are exceptions.

Most definitions that are of types that could have been used to support (directly or indirectly) UML diagrams also have equivalents generated for them. The exceptions are described below. The old definitions may or may not be deleted (see later).

6

Introduction to SA/2001 Conversion

Introduction

This chapter provides an overview of new functionality in System Architect 2001 that requires a conversion when moving to SA/2001 from SA versions 4.0 and/or 3.1.

Overview of What's Changed

In general there are two areas of change in System Architect 2001 that requires conversion for work done in earlier versions:

- Long Names (up to 80 characters) are now allowed for Symbols, Definitions, and Diagrams (the previous limit was 31 characters), and

- Data modeling has changed significantly.

When conversion is run against an encyclopedia created with a previous version of System Architect, a separate conversion routine is run for the each of the two topics above.

Special Note for SA/Catalyst Users

In addition, changes have been made to some of the names of business modeling diagrams and definitions for the Catalyst methodology. This only affects users of SA/Catalyst who have made changes to the properties file (USRPROPS.TXT). Users who have modified properties for SA/Catalyst should read the conversion instructions in chapter 2, and refer to the name changes in chapter 5 of this manual to see if the name changes affect them.

Special Note for SA/BPR Users

In addition SA/BPR users should be aware that IDEF1X data models are not directly converted to the new data modeling format. They can be converted if they are first mapped to physical diagrams in the SA/BPR tool, converted to physical diagrams in SA/2001, then mapped to ER diagrams in SA/2001. In addition, Swimlanes from IDEF and Organizational Units from the Catalyst methodology are both present in SA/2001. SA/BPR users who have modified the properties of a Swimlane definition through USRPROPS.TXT should pay special attention to the conversion instructions for SA/BPR users in Chapter 2.

Long Names

System Architect 2001 supports names up to a maximum of 80 characters long, compared to the old limit of 31 characters. The conversion routine converts all underlying fields so that they can hold the additional characters. Existing names of symbols will not change. You will, after the conversion process is complete, be able to rename old names to new names that are up to 80 characters long.

Internal Timestamps

When an encyclopedia is converted from SA 4.0 to SA/2001, all internal timestamps are preserved. However, in some cases you may find that the external display of a given timestamp appears to be inaccurate, seeming to be off by exactly one or two or three hours.

The difference results from SA 4.0 being a 16-bit application, while SA/2001 is a 32-bit application. When Microsoft upgraded from one environment to the other, they also changed their handling of national and international time zones. In particular, the 16-bit environment's default assumption is that everyone lives and works in the Pacific time zone, whereas the 32-bit environment uses the Windows date/time properties to determine where you live or work.

Here is a typical result: Assume that a user in New York creates a diagram at 8:00 AM using 16-bit SA 4.0. After he converts and inspects the same diagram using 32-bit SA/2001, the diagram's timestamp will be displayed as 11:00 AM.

Data Modeling Changes

Data modeling functionality has been greatly enhanced in System Architect 2001. Data models built with previous versions of System Architect must be run through a conversion routine to make them compatible with data modeling in System Architect 2001. Details about the changes to data modeling functionality and the specifics of what the conversion program does are provided in chapters 3 and 4 of this manual. An overview is provided here.

Models and Subject Areas

You may now partition your project encyclopedia into models, each represented by a model Entity Relation diagram, which is further represented by one or more Subject Area diagrams. A change made to a Subject Area diagram is automatically reflected in the Model Diagram, and vice versa.

What this means for a user converting an existing encyclopedia is that all of your ER diagrams must be made to report to a "model". During the conversion routine, all ER diagrams encountered in the old encyclopedia are made to report to a new "model" named "**Data Model**". Users may not specify the default; however, after the conversion is run, they may do a global rename on the new "model" property.

Referential Integrity

New data modeling functionality includes on-the-fly referential integrity. The rules of data modeling are enforced – 'ghost-buster' symbols prohibit you from dropping a relationship line into space, or from forming an illegal relationship. Entity types are automatically transformed as required based on the relationship you draw. Relationships between entities are maintained for whatever diagram they appear on within a model, within the encyclopedia.

This means a lot for a user converting an existing encyclopedia. For example, if the conversion program finds a line unconnected at one or both ends, it will connect it (as long as the line has a definition that describes its 'from' and 'to' entities or tables) or delete it. If the conversion program encounters relationships that conflict with one another, it will maintain the first one encountered, and make all subsequent conflicting ones non-identifying.

Relationship Lines

In previous versions of System Architect, much of the information on a relationship line was held in the Associative properties dialog. In System Architect 2001, the Associative properties dialog has been removed, and all information about the relationship line has been moved to its Definition.

Global vs Associative Modeling

In previous versions of System Architect, a user had the option of modeling with the Global technique versus the Associative technique. In global modeling, there was only one instance of a data element in an encyclopedia. In Associative modeling, an attribute in an entity was 'associated with' an underlying data element, enabling you to create an underlying data dictionary of data elements that could be reused in different entities within the encyclopedia. However, there were no restrictions that attribute information had to correlate with data element information – it was up to the user to make sure that the information was consistent.

In System Architect 2001, global modeling no longer exists. All modeling is done similar to what was called the 'Associative' technique, but now, also, the tool enforces certain consistencies between the data. Data elements provide the core data dictionary entry. Attributes enable the modeler to enter instance data, such as primary key, foreign key, and nullity information. Data structures provide the modeler with a way to group data elements. Data domains may be used optionally to provide data elements with inherited definitions.

Global Models

What this means for a user converting an existing encyclopedia is that a 'Global' encyclopedia will be converted into an 'Associative' one.

Associative Models

Previous versions of System Architect did not automatically enforce consistency between definitions of attributes and their underlying data elements. Because of this, the conversion program may run into the following situations:

- Attributes for which data elements have not been assigned.
- Attributes for which *data properties* (physical properties such as data type or type qualifiers) do not agree with the data properties of the underlying data element.

Any conflicts will prompt SA to produce a new data element:

- If a data element is not provided, the created data element will be given the name and properties of the attribute.
- If a data element already exists having the same name as the attribute, the created data element will have a -01, -02, etc. extension.
- If properties are overridden which should not be, a new data element will be created having the name of the old data element with a -01, -02, etc. extension. The properties of the created element will include the properties of the attribute and the properties inherited from the data.

Changes to Property Sets

In System Architect 2001, the logical ER diagram provides the user with generic values for some properties, including data type. It is in the physical diagram that the user specifies DBMS-specific values. In previous versions of System Architect, the user may have entered DBMS-specific values into a logical ER diagram.

During conversion, some DBMS-specific properties on a logical ER diagram will be merged into a single generic property. Please refer to Chapter 4 of this manual for specific property mappings.

Symbols and Definitions Removed from ER and Physical Diagrams

A number of symbols have been removed from the data modeling in System Architect 2001, replaced by properties within certain definitions.

Access Path/Index Symbols: These symbols have been removed. The conversion validates Attribute/Column selection and adds access path/index name to list of indices in the entity/table definition.

Database Symbols: These symbols have been removed. The conversion fills in the name of the database in the diagram properties of the physical diagram.

Element Symbol: Previous versions of System Architect allowed for drawing of Element symbols, which represented individual data elements. The conversion adds into the entity definition all of the elements *attached* to entities via relation lines. Note also that users may have had this symbol defined as something else. If the conversion finds that the symbol is defined by something else, it makes the appropriate transformation.

Data Symbol: Previous versions of System Architect allowed for drawing of Data symbols, which represented individual data structures. The conversion adds into the entity definition all of the data structures attached to entities via relation lines. If the Data symbol is unattached, the conversion transforms it into an entity. Note also that many users used this symbol as an entity (redefining the symbol as being defined by entity via USRPROPS.TXT). If the symbol has been assigned the entity definition, the conversion transforms the symbol into an entity.

Relation Diamond Symbol: Previous versions of System Architect allowed for drawing of Relation Diamond symbols. The conversion looks at the connections to the relation symbol, and transform the relation based on the number of relation lines and the cardinality at the ends that connect to the entities:

- If there are only 2 relation lines and both ends are many, the relation diamond and connected relation lines are replaced with a non-specific relation line.
- If there are only 2 relation lines and one end is zero or one, and the other end is many, the relation diamond and connected relation lines are replaced with a relation line whose type corresponds to the relation diamond's definition (identifying or non-identifying); the cardinality is set to match the ends that attach to the entity.
- If there are more than 2 relation lines, the relation diamond is replaced with an associative entity.

XOR Connectors: Previous versions of System Architect allowed for use of XOR connectors within the data model. The conversion replaces the XOR structure with multiple relation lines, and appropriate exclusionary arcs, and determines From/To information based on the cardinality at Entity side of the attached lines.

If all lines are designated as Many on the Entity side, the XOR structure is transformed into an entity.

Business Modeling Name Changes

A number of name changes have been made to diagrams, definitions, and reports in the product to replace the name Catalyst with the term Business Enterprise Modeling.

Changes to USRPROPS.TXT

Users who have modified Catalyst properties through USRPROPS should make the appropriate name changes to their USRPROPS based on the procedures provided in Chapter 2, and the information provided in Chapter 5.

Maintaining Catalyst Name

For Catalyst users wishing to keep the Catalyst name, a USRPROPS.TXT file that will switch the name back is available from IBM. Please contact your sales representative to obtain it.

7

Changes to Data Modeling Behavior

Introduction

This chapter describes changes to data modeling behavior and capabilities and how the conversion deals with these issues.

Overview

This section descibes how data modeling differs in System Architect V8, and System Architect 2001, versus the data modeling found in the 3.1 and 4.0 family of System Architect products (System Architect 3.1 and 4.0, SA/Object Architect 4.0, SA/Data Architect 4.0, and SA/Catalyst 4.0). This section also describes what the conversion program does to deal with each functionality point.

You may use this chapter as a guide to understand how your data models will be converted.

Relationship Lines in Data Model Must Be Connected on Both Ends

SA 3.1/4.0 Behavior	SA/2001 Behavior
Data relationship lines may be unconnected at one or both ends	Lines must be connected to an entity at both ends. Ghost-buster lies on drawing tool until line is correctly connected; user can hit Escape to discontinue line drawing.
What the Conversion Does	
A line that is unconnected at one or both ends will be transformed into a "dog[1]," retaining its underlying definition When this line is connected properly, it is transformed to a relationship line. Default: if a definition exists for the line which has values for both the source entity and target entity, the line will be connected.	

1 A graphic type line with no meaning

Cyclic Relationships

There are three types of relationships that are involved here:

- Cyclic Identifying Relationships
- Entity Participating as a Sub-entity in Super/Sub Relationship and in an Identifying Relationship
- Sub-Entity Is Parent of Itself

Cyclic Identifying Relationships

Behavior in SA 3.1 and 4.0	**SA/2001 Behavior**
Cyclic identifying relationships are allowed within diagrams; FK propagation does not behave correctly when these relationships exist	Cyclic identifying relationships are allowed within one diagram, because during design more than one user's subject area diagrams may cause conflicting relationships when the model is refreshed. Modify the "ghost-buster" dialog to display the triangle with an exclamation point inside. If the line is dropped, add a graphic comment that states 'This line is in a cyclic relationship. It will not automatically be repaired if the cycle is repaired elsewhere.' FK propagation is not allowed on an ERD with these relationships; ERD cannot be used to create a PDM. Ghost-busters displayed from Rules Check can be cleared, even if the user chooses to leave the cyclic relationships in place.
What the Conversion Does	
The pre-conversion report includes the names of all lines participating in any cyclic relationships, and explains that the diagram cannot be used as the source of a PDM.	

Entity May Be a Sub-Entity and Participate in an Identifying Relationship

SA 3.1/4.0 Behavior	SA/2001 Behavior
An entity may participate as a sub-entity in a super/sub relationship, and in an identifying relationship	These relationships are allowed within one diagram, as it is understood that during design more than one user's subject area diagrams may cause conflicting relationships when the model is refreshed.
What the Conversion Does	
The pre-conversion report includes the names of all lines participating in any cyclic relationships, and explains that the diagram cannot be used as the source of a PDM.	

Sub-Entity Is the Parent of Itself

SA 3.1/4.0 Behavior	SA/2001 Behavior
A sub-entity is the parent of itself. E.g., Entity B is the sub-entity of entity A, which is the sub-entity of entity Z. Entity B is the parent of entity Z.	These relationships are allowed within one diagram, as it is understood that during design more than one user's subject area diagrams may cause conflicting relationships when the model is refreshed.
What the Conversion Does	
The pre-conversion report includes the names of all lines participating in any cyclic relationships, and explains that the diagram cannot be used as the source of a PDM.	

Relationship Lines

SA 3.1/4.0 Behavior	SA/2001 Behavior
A non-identifying relationship, where one end is optional, the optional end is assumed to be the child. The line may have been drawn starting from either entity, and the graphic source of the line is ignored by foreign key generation and schema generation.	The graphic start of the line is the "source," the graphic end is the "target." The child end of the line may be of any cardinality; the parent end may be one (identifying) or optional one (non-identifying). In a parent-child relationship, the parent is the source entity; the child is the target entity. Non-specific lines (M:N) have both a source and target relationship.
What the Conversion Does	
Looks at all relationships and, as necessary, changes to 4.0 storage mechanism, using 3.1 rules for determining parentage. Super-sub and M:N relationships are converted.	

SA 3.1/4.0 Behavior	SA/2001 Behavior
Relationships can be "unmarked" and/or "unknown."	Identifying relationships must be of known cardinality; non-identifying relationships may be "unknown". Relationship must be determined using the line type chosen from the Toolbox, or the cardinality chosen from the dialog.
What the Conversion Does	
If line is identifying: unmarked/unknown:M converted to M:N unmarked/unknown:1 converted to [not determined] unmarked/unknown:unmarked/unknown converted to "dog" If line is non-identifying: unmarked/unknown:M not changed unmarked/unknown:1 not changed unmarked/unknown:unmarked/unknown not changed	

Optional Cardinality Is Allowed

SA 3.1/4.0 Behavior	SA/2001 Behavior
"Optional cardinality" in an Identifying relationship is allowed. FK propagation probably does not work correctly.	The parent end of an identifying relationship cannot be optional. Rules Check marks identifying lines with optional cardinality as erroneous. The dialog for the line automatically modifies an identifying relationship line to non-identifying if the user checks that the line is optional.
What the Conversion Does	
Advises the user of all identifying lines that have optional cardinality; the default transformation of these lines will make them mandatory.	

General Items No Longer Supported in SA/2001

The following items are no longer supported in System Architect:

- Relationship Lines Can No Longer Be Attached to XOR and AND Symbols
- Data Structure Symbol No Longer Supported
- Relation Diamond Not Supported
- Symbol for Data Element No Longer Supported
- Access Path Symbol Not Supported
- Index Symbol Not Supported
- Tables Are No Longer Created Which Point to Simply to Source Entity, Source Class, or Source Logical View
- Attribute's Data Type No Longer Overrides Data Type of Underlying Data Element or Underlying Data Domain
- Attributes Derived from a Given Data Element May Override the Values for Nullity
- All Attributes Must Now Be Derived From a Data Element
- Data Elements Derived from Data Domains Can No Longer Have Different Values for Data Type, Type Qualifier, and Nullity

Relationship Lines Can No Longer Be Attached to XOR and AND Symbols

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
In versions 3.1n through 3.1A8, the XOR symbol was placed on a PDM to indicate that a super-entity had been merged to the sub-entities. All relationships that had gone to the super in the source ERD were connected to the XOR symbol. In versions 3.1A9 through 4.0, all relationships were connected to the tables created from the sub-entities. In addition, it is possible that users placed XOR and AND symbols and drew relationships between them and entities, tables or other ERD and PDM symbols. FK sources referred to an entity that did not exist on the PDM; Schema Generator would not run.	XOR and AND symbols do not appear on the toolbox. If they are copied from another diagram, relationship lines cannot be attached to them Merging of properties from the super to the subs during the ERD-to-PDM conversion works correctly. XOR and AND symbols are not used as substitutes for entities.
Conversion Issues	
The conversion report explains that the symbol is useless, and that the user must re-create the PDM from the ERD; no schema generation is allowed from the PDM in which XOR or AND symbols act as the target and/or source of constraints.	

Data Structure Symbol No Longer Supported

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
Data symbol used as a replacement for the entity symbol, with "defined by Entity" notation in USRPROPS. Data symbol placed on the diagram to indicate a data structure.	Data symbol not supported. Data structures not contained in entities. When a defined data structure is dragged into an entity, its component data elements are added individually to the entity.

Conversion Issues

If the data symbol is defined as an entity, it is converted to an entity.

Add a property to the data structure attribute, "OneOf 'Source data structure'", completing the value.

If the symbol is pointing to an Data Structure definition, then enum the bottom level Data Elements retaining the pointers, producing Entity and Attribute definitions. If the Entity name already exists, rename the new symbol and new Definition to name + tie-breaker suffix.

Note that data structure symbols may be attached to one or more entities with relationship lines. Those lines must be converted as relationships in the new diagram.

Relation Diamond Not Supported

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
Relation diamond used instead of relationship line.	Relation diamond not supported
Conversion Issues	
If the relation diamond is defined as an entity, it is converted to an entity. If the relation diamond was not re-defined, and it has only one line as source and one to which it is the target, transform it to a relation line. If it participates in two 1:M relationships, and it is on the "one" side, change it to a single M:N relationship line. If the relation diamond was not re-defined, and it has more than two lines entering or leaving, transform it to an associative entity.	

Symbol for Data Element No Longer Supported

SA 3.1/SA 4.0 Behavior	**SA/2001 Behavior**
Data element symbol used to visually depict the elements of the entity to which it is attached.	Data element symbol is not supported.
Conversion Issues	
If the data element symbol is defined as an entity, it is converted to an entity. If the symbol has not been re-defined in USRPROPS, leave it on the diagram as a "dog," as well as any lines that may have been used to attach it to other symbols. ERD's with this symbol will not be converted to PDM.	

Access Path Symbol Not Supported

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
Access path symbol used to indicate indices; user can enter attributes in the access path which are not in the entity to which it is attached.	Access path symbol not supported; entities have 2 properties: primary key, and indexes. The attributes in the index must be in the entity.
Conversion Issues	
The Access Path is left on the diagram as a "dog," and does not display the attributes; the lines connecting the Access Path to the entity are converted to "dogs." If all the attributes in the Access Path are in the entity, convert the access path to an index. Attributes in the Access Path, but not in the entity, are added to the entity. Both Schema Generator and Foreign Key generation ignore "dogs."	

Index Symbol Not Supported

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
Index symbol on PDM used to indicate indices.	Index symbol not supported. The columns in the index must be in the table.
Conversion Issues	
The Index symbol is left on the diagram as a "dog," and does not display the columns ; the lines connecting the Index to the table are converted to "dogs." If all the columns in the Index are in the Table, convert the Index to an internal index. Columns in the 3.1/4.0 index, but not in the table, are added to the entity. Both Schema Generator and Foreign Key generation ignore "dogs." Creates Primary Key access path for each entity; create Primary Key index for each table. See property conversion document for details.	

Tables Are No Longer Created Which Point Simply to Source Entity, Source Class, or Source Logical View

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
In many versions prior to 3.1J8, tables are created pointing to "source entity," "source class," and/or "source logical view."	Tables have explicit columns.
Conversion Issues	
Table definition is constructed with columns generated from attributes or class attributes in all referenced "source xxx" objects. Some tables may also have columns; these columns will be retained as is.	

Attribute's Data Type No Longer Overrides Data Type of Underlying Data Element or Underlying Data Domain

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
Attribute's data types may override the type of the derived-from data element; Data element's data types may override the type of the derived-from data domain; SQL data types inappropriate for the stated target database; Conflicting data types specified for multiple target databases allowed as values of attributes.	SQL data type mapped into SQL '92 set. Data types are specified for the data element and cannot be overridden at the attribute level. Data types specified for a data domain cannot be overridden at the data element level. SQL data type, precision/length, nullity, and column name are non-DBMS specific, so long as the target database is SQL. AS/400, dBase, Paradox, and other non-SQL databases have their own set of data types, which are supported on definition pages appropriately named.

Conversion Issues
Copies AS/400, dBase and other non-SQL data type as is; continues to support them in Logical to Physical generation. In non-conversion processing, so long as the ERD supports physical properties, non-SQL target database properties are displayed and converted to PDM. E.g., the dBase user may see dBase data types and enter them. If the dBase user enters a value in the SQL data type property, and there is no value in the dBase data type property, the SQL value is converted correctly to dBase when the PDM is generated.

Attributes Derived from a Given Data Element May Override the Values for Nullity

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
In the associative model, many attributes may be defined with the same name, but different values. These attributes may or may not derive from the same data element, and their values may or may not have over-ridden those of the underlying data element.	Attributes derived from a given data element may override the values for nullity. The attribute name, nullity, and column name may vary based on the entity. Attribute name, column name, and nullity are the same as the underlying data element. When a data element's name is changed, those three characteristics that remain identical to those of the data element will also be changed.
Conversion Issues	

If a data element exists with the same name as the attribute, and the same values for all properties except description, the conversion sets the data element property of the attribute to that data element.

If a data element exists with the same name as the attribute, whose property values are different, the conversion creates a new data element for the attribute with a tie-breaker name.

If an attribute exists in one entity more than once, with the same name, and different property values, the conversion issues a tie-breaker name to the duplicate(s). If an attribute exists in one entity more than once, with the same name, and the same property values, the conversion deletes the duplicate(s).

All Attributes Must Now Be Derived From a Data Element

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
Attributes may be defined without an underlying data element.	All attributes are derived from a data element.
Conversion Issues	
If no data element exists for the attribute, a data element will be created with the same name as the attribute, and the attribute's data element name property will be set.	

Data Elements Derived from Data Domains Can No Longer Have Different Values for Data Type, Type Qualifier, and Default Value

SA 3.1/SA 4.0 Behavior	SA/2001 Behavior
Data domains have exactly the same properties as data elements. Data elements may be derived from data domains, and have different values for the properties data type, type qualifier, and nullity.	Data types specified for a data domain cannot be overridden at the data element level. Data domains are not keyed by a specific model, and can be used by any data element in the encyclopedia.
Conversion Issues	
Reference to an undefined data domain is removed from the data element. If the data element refers to a data domain whose property values (specifically, the values for data type, data type qualifiers, and default value) differ, the reference is removed. If the data element refers to a data domain and the property values (specifically, the values for data type, data type qualifiers, and default value) are the same, the property values are left alone. ***(more in next cell of table)***	

Conversion Issues
If a domain contains properties that are no longer in a domain definition within SA/2001, the values of those properties are not brought into the new domain definition. However, if the property had a value in the *attribute* or *data element* definition in the old encyclopedia, those values will be brought into the SA/2001 encyclopedia within the corresponding attribute or data element definition. For example, if, in an old encyclopedia you had a value for a Constraint within an attribute and also in it's undelying domain, the value for the attribute's constraint is brought over into the attribute definition in SA/2001, but not into the domain definition (since domain does not have a Constraint property in SA/2001).

8

Specific Data Modeling Properties Changes

Introduction

This chapter provides a detailed list of specific changes to properties of definitions of System Architect's data modeling, and how the conversion program handles each situation.

Overview

The conversion program performs the following:

- Convert ER diagrams to Subject Area ERDs.
- Populate existing Physical DM diagrams with columns based on "Source Entity" and "Logical View" information.
- Some DBMS-specific properties will be merged into a single generic property. Where a default property has been specified, SA will check if a non-default value has been defined in any of the existing DBMS-specific property fields. If a value other than the default has been specified, SA will assume it to be the true value of the field.
- Previously generic properties which have become DBMS-specific will be replicated into the appropriate fields.

The following pages list the specific actions that the conversion program takes when converting definition properties.

Entity Relation Diagram Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Database		None	REMOVED
DB2 Table Name		None	REMOVED
DBMS	DBMS	Makes 'logical'	Optional
Freeze date	Freeze date	None	
Freeze time	Freeze time	None	
Frozen by	Frozen by	None	
	Full Model	Makes "F"	NEW
Logical/Physical		None	REMOVED
Notes	Notes	None	
Version number		None	REMOVED

Physical Data Model Diagram Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
	Database	Fills in name of the database specified by the database symbol. If database symbol is not named, fillss in "Unspecified"	NEW - REQUIRED
DBMS	DBMS	Makes "Default DBMS" if not specified*	REQUIRED
	Freeze date	None	NEW
	Freeze time	None	NEW
	Frozen by	None	NEW
	Notes	None	NEW
	Source Diagram Name	Inserts Diagram Name	NEW - REQUIRED

* Based on User Option to specifies the default DBMS prior to beginning conversion. Identifying Relation Symbol in Entity Relation (New)

Nonidentifying Relation Symbol in Entity Relation (NEW)

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
	Force Non-identifying		
	From Entity	Fills in with "From Entity" from Relation Line Symbol	
	To Entity	Fills in with "To Entity" from Relation Line Symbol	

Access Path Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Clustered	Clustered	When creating unique index from Attribute/Data Element, makes T if Cluster Index = T	
DATABASE			REMOVED
DB2 BUFFERPOOL	DB2 BUFFERPOOL	Copies from Entity definition when creating PK Index	
DB2 CLOSE	DB2 CLOSE	Copies from Entity definition when creating PK Index	
DB2 DEFER	DB2 DEFER	Copies from Entity definition when creating PK Index	
DB2 DSETPASS	DB2 DSETPASS	Copies from Entity definition when creating PK Index	
DB2 ERASE	DB2 ERASE	Copies from Entity definition when creating PK Index	
DB2 FREEPAGE	DB2 FREEPAGE	Copies from Entity definition when creating PK Index	
DB2 PCTFREE	DB2 PCTFREE	Copies from Entity definition when creating PK Index	
DB2 PRIQTY	DB2 PRIQTY	Copies from Entity	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
		definition when creating PK Index	
DB2 SECQTY	DB2 SECQTY	Copies from Entity definition when creating PK Index	
DB2 SUBPAGES	DB2 SUBPAGES	Copies from Entity definition when creating PK Index	
DB2 USING Name	DB2 USING Name	Copies from Entity definition when creating PK Index	
DB2 USING Type	DB2 USING Type	Copies from Entity definition when creating PK Index	
dBase Index Expression	dBase Index Expression	None	
Description	Description		REDEFINED
Duplicates	Duplicates	None	
Entity Name	Entity Name	None	KEY. PARENT designation removed, relate by changed. New relationship " is indexed by" replaces keyed by*
IGNORE_DUP_KEY	IGNORE_DUP_KEY	Fills in with Primary Key IGNORE_DUP_KEY value from entity when creating PK Index	
	INFORMIX CONSTRAINT	Fills in with value from entity when	Gray if PK not = T moved from Entity

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
	CHECK	creating PK Index	
Ingres Compression	Ingres Compression	Fills in with value from entity when creating PK Index	
Ingres Fills Factor	Ingres Fills Factor	Fills in with value from entity when creating PK Index	
Ingres Leaf Fills	Ingres Leaf Fills	Fills in with value from entity when creating PK Index	
Ingres Location	Ingres Location	Fills in with value from entity when creating PK Index	
Ingres Noduplicates	Ingres Noduplicates	Fills in with value from entity when creating PK Index	
Ingres Nonleaf Fills	Ingres Nonleaf Fills	Fill in with value from entity when creating PK Index	
Ingres Storage Structure	Ingres Storage Structure	Fills in with value from entity when creating PK Index	
	Message Name	When creating unique index from Attribute/Data Element, copies from Attribute/Data Element definition. When Creating PK Index, fills in with Primary Key Message Name value from Entity.	moved from attribute/data element/entity

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
ORACLE INITRANS	ORACLE INITRANS	Fills in with ORACLE INDEX INITRANS value from entity when creating PK Index	
ORACLE MAXTRANS	ORACLE MAXTRANS	Fills in with ORACLE INDEX MAXTRANS value from entity when creating PK Index	
ORACLE NOSORT	ORACLE NOSORT	None	
ORACLE PCTFREE	ORACLE PCTFREE	Fills in with ORACLE INDEX PCTFREE value from entity when creating PK Index	
ORACLE STORAGE PARAMETERS	ORACLE STORAGE PARAMETERS	Fills in with ORACLE INDEX STORAGE PARAMETERS value from entity when creating PK Index	
ORACLE TABLESPACE	ORACLE TABLESPACE	Fills in with ORACLE INDEX TABLESPACE value from entity when creating PK Index	
OWNER		None	REMOVED
	Primary Key	Set=T when creating PK Index	NEW
	Primary Key Constraint Name	Fills in with value from entity when	Gray property if PK not = T

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		creating PK Index. Use PK Constraint Name if given, else use PK Name	moved from Entity If SQL Server/Sybase - use DBMS Specific
	Primary Key Name	Fills in with value from entity when creating PK Index	Gray property if PK not = T moved from Entity
SORTED_DATA	SORTED_DATA	Fills in with Primary Key SORTED_DATA value from entity when creating PK Index	
	SQL Server Constraint Name	Fills in with value from entity when creating PK Index. Fills in with value from attribute (if exists) or data element when creating Unique Index from attribute or data element.	PK Constraint name if PK = T only valid if Unique = T moved from Entity/attribute/data element
SQL Server Fillsfactor	SQL Server Fillsfactor	Fills in with value from attribute/data element when creating unique index. When Creating PK Index, fills in with SQL Server Primary Key Fillsfactor value from Entity.	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
SQL Server Segment Name	SQL Server Segment Name	Fills in with value from attribute/data element when creating unique index. When Creating PK Index, fills in with SQL Server Primary Key Segment value from Entity.	
	SYBASE10 Constraint Name	Fills in with value from entity when creating PK Index. Fills in with value from attribute (if exists) or data element when creating Unique Index from attribute or data element.	PK Constraint name if PK = T only valid if Unique = T moved from Entity/attribute/data element
SYBASE10 Fillsfactor	SYBASE10 Fillsfactor	Fills in with value from attribute/data element when creating unique index. When Creating PK Index, fills in with SYBASE10 Primary Key Fillsfactor value from Entity.	
SYBASE10 MaxRowsPerPage	SYBASE10 MaxRowsPerPage	Fills in with value from attribute/data element when	gray if not Sybase 11 SYBASE10

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		creating unique index. Fills in with value from entity when creating PK index.	Fillsfactor and SYBASE10 MaxRowsPerPage mutually exclusive
SYBASE10 Segment Name	SYBASE10 Segment Name	Fills in with value from attribute/data element when creating unique index. When Creating PK Index, fills in with SYBASE10 Primary Key Segment value from Entity.	
Unique	Unique	Makes T when creating PK or Unique Index	Makes T & gray if PK=T
Word Index	Word Index	None	

* In SA 3.1/SA 4.0 the access path is keyed by the entity to which it is attached -> Entity symbol connects to relation line symbol connects to Access Path symbol uses Access Path definition. In SA/2001, there is a new relationship -> Entity definition *is indexed by* Access Path definition. The Access Path symbol has been removed. During conversion, the appropriate fields are fills in on both sides of the relationship.

Attribute Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Alias	Alias	None	
AllowNull		Set Default Nullity	REMOVED
Business Unit		Moves to Data Element *	Property of Data
C Storage Type		Moves to Data Element *	Property of Data
C Storage Occurrences		Moves to Data Element *	Property of Data
Caption	Caption	None	
Case	Case	None	
Check		Moves to Data Element *	Property of Data
CheckMsgId		Moves to Data Element *	Property of Data
Cluster Index		Moves to Access Path **	Property of Index
Coding Usage		Moves to Data Element *	Property of Data
Coding Scheme		Moves to Data Element *	Property of Data
ColHdg	ColHdg	None	
Column Name	Column Name	If not given, fills in with mapped attribute name	Filled in by SA
Comments	Comments	None	
Data Element	Data Element	If not given, fills in with Data element	REQUIRED

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		being created	
Database Comment	Database Comment	None	
DataType		Maps to appropriate SQL Data Type	REMOVED
DB2 Character Type		Moves to Data Element *	Property of Data
dBase Data Type		Maps to appropriate SQL Data Type	REMOVED
dBase Field Name		Moves to Column Name	REMOVED
dBase Naming Prefix		If attribute references data element, renames attribute with prefix. If attribute references data structure, decomposes structure and prefix new attributes. (1 data element per attribute)	REMOVED
Decimals		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
DecPos		Moves to Data Element *	Property of Data

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Default	PowerBuilder Default	Copies value from Default into PowerBuilder Default	RENAMED
Default Name		Moves to Data Element *	Property of data
Default Nullity	Default Nullity	If Null, makes F If Not Null, makes T If Not Null Unique, makes T & makes Unique=T If Not Null with Default, makes T & makes System Default=T	REDEFINED
Default Value		Moves to Data Element *	Property of Data
Description	Description	None	
Display Format	Display Format	None	
	Display Length	Fills in from SQL Type Qualifier	NEW - for Screen Painter Default fills in with SQL Type Qualifier length
Display Picture		Moves to Data Element *	Property of Data
Edit		Moves to Data Element *	Property of Data
Edit Style	Edit Style	None	
Entity Name	Entity Name	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Field Length		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
FieldName		Moves to Column Name	REMOVED
Field Type		Maps to appropriate SQL Data Type	REMOVED
Function		Moves to Data Element *	Property of Data
Heading	Heading	None	
Heading Position	Heading Position	None	
Height	Height	None	
Identity	Identity	None	
Identity Qualifiers	Identity Qualifiers	None	
INFORMIX CONSTRAINT CHECK		Moves to Data Element *	Property of Data
INFORMIX Constraint Check Name		Moves to Data Element *	Property of Data
InterBase Array Dim		Moves to Data Element *	Property of Data
InterBase Character Set		Moves to Data Element *	Property of Data
InterBase Collate		Moves to Data Element *	Property of Data
InterBase Compute		Moves to Data	Property of Data

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Expr		Element *	
InterBase CONSTRAINT CHECK		Moves to Data Element *	Property of Data
InterBase Constraint Check Name		Moves to Data Element *	Property of Data
InterBase Segment Size		Moves to Data Element *	Property of Data
InterBase Sub Type		Moves to Data Element *	Property of Data
	Key	Makes T if @ found in Entity	NEW - if PK makes not null
	Key Component Number	Fills in with number following @ sign in entity definition.	Required for PK attributes.
Justify	Justify	None	
Label	Label	None	
Label Position	Label Position	None	
Length		Moves to SQL Type Qualifier if value doesnt exist there (length,precision) Copies to Display Length	REMOVED
Length Description		Moves to Data Element *	Property of Data
Logical Format		Moves to Data Element *	Property of Data

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Logical Length		Moves to Data Element *	Property of Data
Message Name		Moves to Access Path **	Property of Index
Naming Prefix		If attribute references data element, renames attribute with prefix. If attribute references data structure, decomposes structure and prefix new attributes. (1 attribute per data element)	REMOVED
Notes		Moves to Data Element *	Property of Data
Optionality		Moves to Data Element *	Property of Data
ORACLE CONSTRAINT CHECK		Moves to Data Element *	Property of Data
ORACLE Constraint Check Name		Moves to Data Element *	Property of Data
ORACLE Unique		If T, makes Unique T and create Unique Index for column If F, do nothing	REPLACED
Owner		Moves to Data	Property of Data

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		Element *	
Paradox picture		None	REMOVED
Progress Case-sensitive		Moves to Data Element *	Property of Data
Progress Column-Label	Progress Column-Label	None	
Progress Data-Type		Maps to appropriate SQL Data Type	REMOVED
Progress Decimals		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
Progress Desc		Moves to Data Element *	Property of Data
Progress Extent		Moves to Data Element *	Property of Data
Progress Format		Moves to Data Element *	Property of Data
Progress Help		Moves to Data Element *	Property of Data
Progress Index		Moves to Data Element *	Property of Data
Progress Initial		Moves to Data Element *	Property of Data
Progress Label	Progress Label	None	
Progress Mandatory		Set Nullity	REMOVED
Progress Valexp		Moves to Data	Property of Data

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		Element *	
Progress Valmsg		Moves to Data Element *	Property of Data
Progress View-As		Moves to Data Element *	Property of Data
Rule Name		Moves to Data Element *	Property of Data
SQL Data Type		Moves to Data Element *	Property of Data
SQL Server Constraint Name		Moves to Access Path **	Property of Index
SQL Server Check Constraint		Moves to Data Element *	Property of Data
SQL Server Check Constraint Name		Moves to Data Element *	Property of Data
SQL Server Default Constraint Name		Moves to Data Element *	Property of Data
SQL Server Fillsfactor		Moves to Access Path **	Property of Index
SQL Server Segment Name		Moves to Access Path **	Property of Index
SQL Server Unique		If T, makes Unique T If F, do nothing	REPLACED
SQL Type Qualifiers		Moves to Data Element *	Property of Data
Standard Messages		Moves to Data Element *	Property of Data
Storage Class		Moves to Data Element *	Property of Data

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Storage Picture		Moves to Data Element *	Property of Data
SYBASE10 Check Constraint		Moves to Data Element *	Property of Data
SYBASE10 Check Constraint Name		Moves to Data Element *	Property of Data
SYBASE10 Check Message Name		Moves to Data Element *	Property of DE
SYBASE10 Constraint Name		Moves to Access Path **	Property of Index
SYBASE10 Fillsfactor		Moves to Access Path **	Property of Index
SYBASE10 MaxRowsPerPage		Moves to Access Path **	Property of Index
SYBASE10 Segment Name		Moves to Access Path **	Property of Index
SYBASE10 Unique		If T, makes Unique T If F, do nothing	REPLACED
Synonym(s)		Moves to Data Element *	Property of Data
Text definition			
Unit of Measure		Moves to Data Element *	Property of Data
User Roles		Moves to Data Element *	Property of Data
Validation/Derivation		Moves to Data Element *	Property of Data
	Unique	Fills in based on DBMS-specific	NEW - replaces DBMS specific

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		unique properties. If any are T, makes it T.	unique
Validation Rule	Validation Rule	None	
Value for Null		Moves to Data Element *	Property of Data
WATCOM CONSTRAINT CHECK		Moves to Data Element *	Property of Data
Width	Width	None	

* If value already specified in Data Element, and is different, create new data element

** If Unique =T, create access path. Specifies index and constraint properties on access path definition.

Column Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Alias	Alias	None	
Attribute	Attribute	None	
Business Unit	Business Unit	None	
Caption	Caption	None	
C Storage Occurrences	C Storage Occurrences	None	
C Storage Type	C Storage Type	None	
Caption	Caption	None	
Case	Case	None	
Check	Check	None	
CheckMsgId	CheckMsgId	None	
Cluster Index		Creates appropriate index and specifies Cluster there	REMOVED
Coding Scheme	Coding Scheme	None	
Coding Usage	Coding Usage	None	
ColHdg	ColHdg	None	
Comments	Comments	None	
Database Comment	Database Comment	None	
Database Name	Database Name	None	KEY
DataType		Maps to appropriate SQL Data Type	REMOVED
DB2 Character Type	DB2 Character Type	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
dBase Data Type		Maps to appropriate SQL Data Type	REMOVED
dBase Field Name		Moves to Column Name	REMOVED
dBase Naming Prefix		If Column references data element or nothing, renames column with prefix. If column references data structure, decomposes structure and prefix new columns. (1 data element per column)	REMOVED
Decimals		Moves to SQL Type Qualifier if value doesn't exist there (length,precision)	REMOVED
DecPos	DecPos	None	
Default	PowerBuilder Default	Copies value from Default into PowerBuilder Default	RENAMED
Default Name	Default Name	None	grayed if Default Value provided
Default Nullity	Default Nullity	If Null, makes F If Not Null, makes T If Not Null Unique, makes T & makes	REDEFINED

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		Unique=T If Not Null with Default, makes T & makes System Default=T	
Default Value	Default Value	None	
Description	Description	None	
Display Format	Display Format	None	
	Display Length	Fills in with value from SQL Type Qualifier	NEW - for Screen Painter Default fills in with SQL Type Qualifier length
Display Picture	Display Picture	None	
Domain	Domain	None	
Edit	Edit	None	
Edit Style	Edit Style	None	
Field Length		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
FieldName		Moves to Column Name	REMOVED
Field Type		Maps to appropriate SQL Data Type	REMOVED
Function	Function	None	
Heading	Heading	None	
Heading Position	Heading Position	None	
Height	Height	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Identity	Identity	None	
Identity Qualifiers	Identity Qualifiers	None	
INFORMIX CONSTRAINT CHECK	INFORMIX CONSTRAINT CHECK	None	
INFORMIX Constraint Check Name	INFORMIX Constraint Check Name	None	
InterBase Array Dim	InterBase Array Dim	None	
InterBase Character Set	InterBase Character Set	None	
InterBase Collate	InterBase Collate	None	
InterBase Compute Expr	InterBase Compute Expr	None	
InterBase Constraint Check Name	InterBase Constraint Check Name	None	
InterBase CONSTRAINT CHECK	InterBase CONSTRAINT CHECK	None	
InterBase Segment Size	InterBase Segment Size	None	only valid if Data Type is BLOB
InterBase Sub Type	InterBase Sub Type	None	only valid if Data Type is BLOB
Label	Label	None	
Label Position	Label Position	None	
Length		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
		Copies to Display Length	
	Key	Makes T if @ found in Entity	NEW - if PK makes not null
	Key Component Number	Fills in with number following @ sign in table definition.	Required for PK attributes.
Justify	Justify	None	
Naming Prefix		If Column references data element or nothing, renames column with prefix. If column references data structure, decomposes structure and prefix new columns. (1 data element per column)	REMOVED
ORACLE CONSTRAINT CHECK	ORACLE CONSTRAINT CHECK	None	
ORACLE Constraint Check Name	ORACLE Constraint Check Name	None	
ORACLE Unique		If T, makes Unique T If F, do nothing	REPLACED
Owner Name	Owner Name		KEY
Progress Case-sensitive	Progress Case-sensitive	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Progress Column-Label	Progress Column-Label	None	
Progress Data-Type		Maps to appropriate SQL Data Type	REMOVED
Progress Decimals		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
Progress Desc	Progress Desc	None	
Progress Extent	Progress Extent	None	
Progress Format	Progress Format	None	
Progress Help	Progress Help	None	
Progress Index	Progress Index	None	
Progress Initial	Progress Initial	None	
Progress Label	Progress Label	None	
Progress Valexp	Progress Valexp	None	
Progress Valmsg	Progress Valmsg	None	
Progress View-As	Progress View-As	None	
Rule Name	Rule Name	None	
	Source Diagram Name		NEW - KEY
SQL Data Type	SQL Data Type	None	
SQL Type Qualifiers	SQL Type Qualifiers	None	
SQL Server Check Constraint	SQL Server Check Constraint	None	
SQL Server Check Constraint Name	SQL Server Check Constraint Name	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
SQL Server Default Constraint Name	SQL Server Default Constraint Name	None	
SQL Server Unique		If T, makes Unique T If F, do nothing	REPLACED
Storage Class	Storage Class	None	
Storage Picture	Storage Picture	None	
SYBASE10 Check Constraint	SYBASE10 Check Constraint	None	
SYBASE10 Check Constraint Name	SYBASE10 Check Constraint Name	None	
SYBASE10 Check Message Name	SYBASE10 Check Message Name	None	
SYBASE10 Unique		If T, makes Unique T If F, do nothing	REPLACED
	System Default	Makes T if Nullity = Non Null with default	NEW - due to Nullity redefined - available only if NOT NULL
Table Name	Table Name	None	KEY
	Unique	Makes T if any <DBMS>Unique is T	NEW - replaces DBMS specific unique
Validation Rule	Validation Rule	None	
WATCOM CONSTRAINT CHECK	WATCOM CONSTRAINT CHECK	None	
Width	Width	None	

Constraint Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
	Allow Nulls	Set from associative properties	NEW
Child Delete		None	REMOVED
Child Delete Error Code		None	REMOVED
Child Delete Error Message		None	REMOVED
Child Insert	Child Insert	None	
Child Insert Error Code	Child Insert Error Code	None	
Child Insert Error Message	Child Insert Error Message	None	
Child Update	Child Update	None	
Child Update Error Code	Child Update Error Code	None	
Child Update Error Message	Child Update Error Message	None	
Database Name	Database Name	None	**KEY**
Description	Description	None	
	Foreign Keys and Roles	Set from FKFrom and attribute name	NEW
Foreign Key Name		None	REMOVED
From Owner Name	From Owner Name	None	**KEY**
From Table	From Table	None	**KEY**
Identifying	Identifying	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Parent Delete	Parent Delete	None	
Parent Delete Error Code	Parent Delete Error Code	None	
Parent Delete Error Message	Parent Delete Error Message	None	
Parent Insert		None	REMOVED
Parent Insert Error Code		None	REMOVED
Parent Insert Error Message		None	REMOVED
Parent Update	Parent Update	None	
Parent Update Error Code	Parent Update Error Code	None	
Parent Update Error Message	Parent Update Error Message	None	
Relationship	Relationship	None	
	Source Diagram Name	None	NEW - **KEY**
SYBASE10 Message Name	SYBASE10 Message Name	None	
	To Cardinality	Sets from associative properties	NEW
To Owner Name	To Owner Name	None	**KEY**
To Table	To Table	None	**KEY**

Data Domain Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Alias		Copies to Data Element if not overridden in DE definition	Not Property of Domain
AllowNull		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Business Unit		Copies to Data Element if not overridden in DE definition	Not Property of Domain
C Storage Occurrences		Copies to Data Element if not overridden in DE definition	Not Property of Domain
C Storage Type		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Caption		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Case		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Check		Copies to Data Element if not overridden in DE	Not Property of Domain

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		definition	
CheckMsgId		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Cluster Index		Copies to Data Element if not overridden in DE definition	REMOVED
Coding Scheme		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Coding Usage		Copies to Data Element if not overridden in DE definition	Not Property of Domain
ColHdg		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Column Name		None	Not Property of Domain
Comments		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Database		None	REMOVED
DataType		Maps to appropriate SQL Data Type	REMOVED
DB2 Character Type	DB2 Character Type	None	
dBase Data Type		Maps to appropriate	REMOVED

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
		SQL Data Type	
dBase Field Name		None	REMOVED
Decimals		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
DecPos		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Default		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Default Name	Default Name	None	grayed if Default Value provided
Default Nullity		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Default Value	Default Value	None	
Description	Description	None	
Display Format		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Display Picture		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Domain		If multi-level domain in use, follows chain	Not Property of Domain

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		of inheritance copying properties to top level if not overridden in upper levels . Once the domains have been resolved, copies properties to data element definition.	
Edit		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Edit Style		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Field Length		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
Field Type		Moves to Column Name	REMOVED
FieldName		Maps to appropriate SQL Data Type	REMOVED
Function		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Future Only	Future Only	None	
Heading		Copies to Data Element if not overridden in DE definition	Not Property of Domain

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Heading Position	▨	Copies to Data Element if not overridden in DE definition	Not Property of Domain
Height	▨	Copies to Data Element if not overridden in DE definition	Not Property of Domain
Identity	▨	Copies to Data Element if not overridden in DE definition	Not Property of Domain
Identity Qualifiers	▨	Copies to Data Element if not overridden in DE definition	Not Property of Domain
IDEF1X Entity State	IDEF1X Entity State	None	IDEF1X
INFORMIX Constraint Check Name	INFORMIX Constraint Check Name	None	
INFORMIX CONSTRAINT CHECK	INFORMIX CONSTRAINT CHECK	None	
Initial Value	▨	None	REMOVED
InterBase Array Dim	▨	Copies to Data Element if not overridden in DE definition	Not Property of Domain
InterBase Character Set	InterBase Character Set	None	
InterBase Collate	▨	Copies to Data Element if not overridden in DE	Not Property of Domain

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		definition	
InterBase Compute Expr		Copies to Data Element if not overridden in DE definition	Not Property of Domain
InterBase CONSTRAINT CHECK	InterBase CONSTRAINT CHECK	None	
InterBase Constraint Check Name	InterBase Constraint Check Name	None	
InterBase Segment Size	InterBase Segment Size	None	
InterBase Sub Type	InterBase Sub Type	None	
Justify		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Label		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Label Position		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Length		Moves to SQL Type Qualifier if value doesnt exist there (length,precision) Copies to Display Length	REMOVED
Length Description		Copies to Data	Not Property of

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		Element if not overridden in DE definition	Domain
Logical Format		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Logical Length		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Message Name		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Model	Model	None	IDEF1X - KEY
Notes		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Optionality		Copies to Data Element if not overridden in DE definition	Not Property of Domain
ORACLE CONSTRAINT CHECK	ORACLE CONSTRAINT CHECK	None	
ORACLE Constraint Check Name	ORACLE Constraint Check Name	None	
ORACLE Unique		Copies to Data Element if not overridden in DE definition	Not Property of Domain

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Owner		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Paradox picture		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Case-sensitive		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Column-Label		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Data-Type		Maps to appropriate SQL Data Type	REMOVED
Progress Decimals		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
Progress Desc		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Extent		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Format		Copies to Data Element if not overridden in DE	Not Property of Domain

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		definition	
Progress Help		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Index		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Initial		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Label		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Mandatory		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Valexp		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress Valmsg		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Progress View-As		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Rule Name	Rule Name	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
SQL Data Type	SQL Data Type	None	
SQL Server Check Constraint	SQL Server Check Constraint	None	
SQL Server Check Constraint Name	SQL Server Check Constraint Name	None	
SQL Server Constraint Name		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SQL Server Default Constraint Name	SQL Server Default Constraint Name	None	
SQL Server Fillsfactor		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SQL Server Segment Name		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SQL Server Unique		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SQL Type Qualifiers	SQL Type Qualifiers	None	
Standard Messages		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Storage Class		Copies to Data Element if not overridden in DE definition	Not Property of Domain

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Storage Picture		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SYBASE10 Check Constraint	SYBASE10 Check Constraint	None	
SYBASE10 Check Constraint Name	SYBASE10 Check Constraint Name	None	
SYBASE10 Check Message Name	SYBASE10 Check Message Name	None	
SYBASE10 Constraint Name		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SYBASE10 Fillsfactor		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SYBASE10 MaxRowsPerPage		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SYBASE10 Segment Name		Copies to Data Element if not overridden in DE definition	Not Property of Domain
SYBASE10 Unique		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Synonym(s)		Copies to Data Element if not overridden in DE	Not Property of Domain

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		definition	
Table		None	REMOVED
Text definition		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Unit of Measure		Copies to Data Element if not overridden in DE definition	Not Property of Domain
User Defined DataType		Copies to Data Element if not overridden in DE definition	Not Property of Domain
User Roles		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Validation Rule		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Validation/Derivation		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Value for Null		Copies to Data Element if not overridden in DE definition	Not Property of Domain
Version		None	REMOVED
WATCOM CONSTRAINT	WATCOM CONSTRAINT	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
CHECK	CHECK		
Width		Copies to Data Element if not overridden in DE definition	Not Property of Domain

**** NOTE: Multi-level domains may be in use. If a Domain is specified for a data domain, the domains must be consolidated before beginning the conversion.

**** NOTE: Some properties that are not included in the SA/2001 Data Element definition are listed here to be moved to the data element definition. The conversion needs to be performed in 2 steps. First, copies all domain properties to the data element if they aren't overridden in the data element definition. Second, follow the appropriate conversion specification in the data element definition.

Data Element Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Alias		Moves to Attribute *	
AllowNull		Moves to Attribute *	
Business Unit	Business Unit	None	
C Storage Type	C Storage Type	None	
C Storage Occurrences	C Storage Occurrences	None	
Caption		Moves to Attribute *	
Case		Moves to Attribute *	
Check	Check	None	
CheckMsgId	CheckMsgId	None	
Cluster Index		Create appropriate access path and specifies Cluster there **	REMOVED
Coding Scheme	Coding Scheme	None	
Coding Usage	Coding Usage	None	
ColHdg		Moves to Attribute *	
Column Name		Moves to Attribute *	
Comments		Moves to Attribute *	
Database		None	REMOVED
Database Comment		Moves to Attribute *	
DataType		Maps to appropriate SQL Data Type	REMOVED
DB2 Character Type	DB2 Character Type	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
dBase Data Type		Maps to appropriate SQL Data Type	REMOVED
dBase Field Name		Moves to Column Name	REMOVED
dBase Naming Prefix		(Re)names attribute with prefix	REMOVED
Decimals		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
DecPos	DecPos	None	
Default		Moves to Attribute *	RENAMED PowerBuilder Default
Default Name	Default Name	None	grayed if Default Value provided
Default Nullity		Moves to Attribute *	
Default Value	Default Value	None	
Description	Description	None	
Display Format		Moves to Attribute *	
Display Picture	Display Picture	None	
Domain	Domain	None	
Edit	Edit	None	
Edit Style		Moves to Attribute *	
Entity Name		None	REMOVED
Field Length		Moves to SQL Type Qualifier if value doesnt exist there	REMOVED

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		(length,precision)	
Field Type		Maps to appropriate SQL Data Type	REMOVED
FieldName		Moves to Column Name	REMOVED
Function	Function	None	
Heading		Moves to Attribute *	
Heading Position		Moves to Attribute *	
Height		Moves to Attribute *	
Identity		Moves to Attribute *	
Identity Qualifiers		Moves to Attribute *	
INFORMIX CONSTRAINT CHECK	INFORMIX CONSTRAINT CHECK	None	
INFORMIX Constraint Check Name	INFORMIX Constraint Check Name	None	
Initial Value		None	REMOVED
InterBase Array Dim	InterBase Array Dim	None	
InterBase Character Set	InterBase Character Set	None	
InterBase Collate	InterBase Collate	None	
InterBase Compute Expr	InterBase Compute Expr	None	
InterBase CONSTRAINT CHECK	InterBase CONSTRAINT CHECK	None	
InterBase Constraint Check	InterBase Constraint Check	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Name	Name		
InterBase Segment Size	InterBase Segment Size	None	only valid if Data Type is BLOB
InterBase Sub Type	InterBase Sub Type	None	only valid if Data Type is BLOB
Justify		Moves to Attribute *	
Label		Moves to Attribute *	
Label Position		Moves to Attribute *	
Length		Moves to SQL Type Qualifier if value doesnt exist there (length,precision) Copies to Display Length	
Length Description	Length Description	None	
Logical Format	Logical Format	None	
Logical Length	Logical Length	None	
Message Name		Moves to Access Path definition **	Property of Unique Index
Naming Prefix		(Re)names attribute with prefix	REMOVED
Notes	Notes	None	
Optionality	Optionality	None	
ORACLE CONSTRAINT CHECK	ORACLE CONSTRAINT CHECK	None	
ORACLE Constraint Check Name	ORACLE Constraint Check Name	None	
ORACLE Unique		Moves to Attribute *	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		If T, makes Unique T If F, do nothing	
Owner	Owner	None	
Paradox picture		None	REMOVED
Progress Case-sensitive	Progress Case-sensitive	None	
Progress Column-Label		Moves to Attribute *	
Progress Data-Type		Maps to appropriate SQL Data Type	REMOVED
Progress Decimals		Moves to SQL Type Qualifier if value doesnt exist there (length,precision)	REMOVED
Progress Desc	Progress Desc	None	
Progress Extent	Progress Extent	None	
Progress Format	Progress Format	None	
Progress Help	Progress Help	None	
Progress Index	Progress Index	None	
Progress Initial	Progress Initial	None	
Progress Label		Moves to Attribute *	
Progress Mandatory		Used to set Attribute's default Nullity if value not given in Attribute or DE	REMOVED
Progress Valexp	Progress Valexp	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Progress Valmsg	Progress Valmsg	None	
Progress View-As	Progress View-As	None	
Rule Name	Rule Name	None	
SQL Data Type	SQL Data Type	None	
SQL Server Check Constraint	SQL Server Check Constraint	None	
SQL Server Check Constraint Name	SQL Server Check Constraint Name	None	
SQL Server Constraint Name		Moves to Access Path definition **	Property of Unique Index
SQL Server Default Constraint Name	SQL Server Default Constraint Name	None	
SQL Server Fillsfactor		Moves to Access Path definition **	Property of Unique Index
SQL Server Segment Name		Moves to Access Path definition **	Property of Unique Index
SQL Server Unique		Moves to Attribute * If T, makes Unique T If F, do nothing	
SQL Type Qualifiers	SQL Type Qualifiers	None	
Standard Messages	Standard Messages	None	
Storage Class	Storage Class	None	
Storage Picture	Storage Picture	None	
SYBASE10 Check Constraint	SYBASE10 Check Constraint	None	
SYBASE10 Check Constraint Name	SYBASE10 Check Constraint Name	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
SYBASE10 Check Message Name	SYBASE10 Check Message Name	None	
SYBASE10 Constraint Name		Moves to Access Path definition **	Property of Unique Index
SYBASE10 Fillsfactor		Moves to Access Path definition **	Property of Unique Index
SYBASE10 MaxRowsPerPage		Moves to Access Path definition **	Property of Unique Index
SYBASE10 Segment Name		Moves to Access Path definition **	Property of Unique Index
SYBASE10 Unique		Moves to Attribute * If T, makes Unique T If F, do nothing	
Synonym(s)	Synonym(s)	None	
	System Default	None	NEW - due to Nullity redefined - available only if NOT NULL
Table		None	REMOVED
Text definition		None	
Unit of Measure	Unit of Measure	None	
User Roles	User Roles	None	
Validation Rule		Moves to Attribute *	
Validation/Derivation	Validation/Derivation	None	
Value for Null	Value for Null	None	
Version		None	REMOVED
WATCOM CONSTRAINT	WATCOM CONSTRAINT	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
CHECK	CHECK		
Width		Moves to Attribute *	

* If value already specified in Attribute, do not replace

Data Struct Domain Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Business Owner	Business Owner	Copies to Data Structure if not already specified there.	
dBase Database Name		None.	
dBase Naming Prefix		Copies to Data Structure if not already specified there.	
Description	Description	None	
Domain		Copies to Data Structure if not already specified there.	
Entity Name		None	
Naming Prefix	Naming Prefix	Copies to Data Structure if not already specified there.	
Progress Desc		Copies to Data Structure if not already specified there.	
Progress Dump Name		Copies to Data Structure if not already specified there.	
Progress Valexp		Copies to Data Structure if not	

		already specified there.	
Progress Valmsg		Copies to Data Structure if not already specified there.	

Data Structure Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Business Owner	Business Owner	Copies to each Data Element in Data Structure if not already specified there.	
Caption	Caption	None	
dBase Database Name		None	REMOVED
dBase Naming Prefix		Apply to individual attributes fromed from the decomposition of the data structure if generic Naming Prefix is not provided.	
Description	Description	None	
Domain		If multi-level domain in use, follow chain of inheritance copiesing properties to top level if not overridden in upper levels . Once the domains have been resolved, copies properties to data structure definition.	REMOVED
Entity Name		None	REMOVED
Naming Prefix	Naming Prefix	Apply to individual attributes fromed	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		from the decomposition of the data structure.	
Progress Desc		None	REMOVED
Progress Dump Name		Copies to each Data Element in Data Structure if not already specified there.	REMOVED
Progress Valexp		Copies to each Data Element in Data Structure if not already specified there.	REMOVED
Progress Valmsg		Copies to each Data Element in Data Structure if not already specified there.	REMOVED

Entity Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Additional relationships	Additional relationships	None	
Archive and destruction	Archive and destruction	None	
Average	Average	None	
Business Description	Business Description	None	
Business Owner	Business Owner	None	
Caption	Caption	None	
CLUSTER		Moves to Access Path definition *	Property of PK Index
Cluster Primary Index		Moves to Access Path definition *	Property of PK Index
Comments	Comments	None	
Create Trailer	Create Trailer	None	
Data Font Name	Data Font Name	None	
DATABASE		None	REMOVED
Database Comment	Database Comment	None	
DB2 AUDIT	DB2 AUDIT	None	
DB2 BUFFERPOOL		Moves to Access Path definition *	Property of PK Index
DB2 CLOSE		Moves to Access Path definition *	Property of PK Index
DB2 Data Capture	DB2 Data Capture	None	
DB2 DSETPASS		Moves to Access	Property of PK

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		Path definition *	Index
DB2 EDITPROC	DB2 EDITPROC	None	
DB2 ERASE		Moves to Access Path definition *	Property of PK Index
DB2 FIELDPROC		None	REMOVED
DB2 FREEPAGE		Moves to Access Path definition *	Property of PK Index
DB2 OBID	DB2 OBID	None	
DB2 PCTFREE		Moves to Access Path definition *	Property of PK Index
DB2 PRIQTY		Moves to Access Path definition *	Property of PK Index
DB2 SECQTY		Moves to Access Path definition *	Property of PK Index
DB2 SUBPAGES		Moves to Access Path definition *	Property of PK Index
DB2 TABLESPACE	DB2 TABLESPACE	None	
DB2 USING Name		Moves to Access Path definition *	Property of PK Index
DB2 USING Type		Moves to Access Path definition *	Property of PK Index
DB2 VALIDPROC	DB2 VALIDPROC	None	
dBase Database Name		None	REMOVED
dBase Naming Prefix		Fills in Naming Prefix field	REMOVED
Delete Trigger Name	Delete Trigger Name	None	
Description	Description	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Function	Function	None	
Growth per period	Growth per period	None	
Headings Font Name	Headings Font Name	None	
	Indices	Fills in with names of all Access Path definitions that are Keyed by the entity.	NEW
INFORMIX CONSTRAINT CHECK		Moves to Access Path definition *	Property of PK Index
Informix DBSPACE	Informix DBSPACE	None	
Informix EXTENT	Informix EXTENT	None	
Informix Fragment	Informix Fragment	None	
Informix Lock Mode	Informix Lock Mode	None	
Informix NEXT	Informix NEXT	None	
INFORMIX Trigger	INFORMIX Trigger	None	
Ingres Compression		Moves to Access Path definition *	Property of PK Index
Ingres Fills Factor		Moves to Access Path definition *	Property of PK Index
Ingres Leaf Fills		Moves to Access Path definition *	Property of PK Index
Ingres Location		Moves to Access Path definition *	Property of PK Index
Ingres Noduplicates		Moves to Access Path definition *	Property of PK Index
Ingres Nonleaf Fills		Moves to Access Path definition *	Property of PK Index

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Ingres Storage Structure		Moves to Access Path definition *	Property of PK Index
Insert Trigger Name	Insert Trigger Name	None	
InterBase CONSTRAINT CHECK	InterBase CONSTRAINT CHECK	None	
InterBase Constraint Check Name	InterBase Constraint Check Name	None	
InterBase Trigger	InterBase Trigger	None	
Labels Font Name	Labels Font Name	None	
Location	Location	None	
Logical View			REMOVED
Maximum	Maximum	None	
Naming Prefix	Naming Prefix	None	
Normalize		None	REMOVED
Notes	Notes	None	
ORACLE CONSTRAINT CHECK	ORACLE CONSTRAINT CHECK	None	
ORACLE Constraint Check Name	ORACLE Constraint Check Name	None	
ORACLE INDEX INITRANS		Moves to Access Path definition (ORACLE INITRANS) *	Property of PK Index
ORACLE INDEX MAXTRANS		Moves to Access Path definition (ORACLE MAXTRANS) *	Property of PK Index

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
ORACLE INDEX PCTFREE		Moves to Access Path definition (ORACLE PCTFREE) *	Property of PK Index
ORACLE INDEX STORAGE PARAMETERS		Moves to Access Path definition (ORACLE STORAGE PARAMETERS) *	Property of PK Index
ORACLE INDEX TABLESPACE		Moves to Access Path definition (ORACLE TABLESPACE) *	Property of PK Index
ORACLE INITRANS	ORACLE INITRANS	None	
ORACLE MAXTRANS	ORACLE MAXTRANS	None	
ORACLE PCTFREE	ORACLE PCTFREE	None	
ORACLE PCTUSED	ORACLE PCTUSED	None	
ORACLE STORAGE PARAMETERS	ORACLE STORAGE PARAMETERS	None	
ORACLE TABLESPACE	ORACLE TABLESPACE	None	
Oracle Trigger	Oracle Trigger	None	
Owner	Owner	None	
Primary Index Name		When specified, use as "Name" when creating Access Path definition for PK index *	REMOVED -Is the name of the PK Index created by SA
Primary Key		Moves to Access	Property of PK

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Constraint Name		Path definition *	Index
Primary Key IGNORE_DUP_KEY		Moves to Access Path definition (IGNORE_DUP_KEY)*	Property of PK Index
Primary Key Message Name		Moves to Access Path definition (Message Name)*	Property of PK Index
Primary Key Name		Moves to Access Path definition (Primary Key Constraint Name)*	Property of PK Index
Primary Key SORTED_DATA		Moves to Access Path definition (SORTED_DATA)*	Property of PK Index
Progress Create Trigger File	Progress Create Trigger File	None	
Progress Delete Trigger File	Progress Delete Trigger File	None	
Progress Desc	Progress Desc	None	
Progress Dump Name	Progress Dump Name	None	
Progress Valexp	Progress Valexp	None	
Progress Valmsg	Progress Valmsg	None	
Progress Write Trigger File	Progress Write Trigger File	None	
Purpose	Purpose	None	
Record Format	Record Format	None	
Security measures	Security measures	None	
Source Class	Source Class	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
	Source Table		NEW
SQL Server Check Constraint	SQL Server Check Constraint	None	
SQL Server Check Constraint Name	SQL Server Check Constraint Name	None	
SQL Server Primary Key Fillsfactor		Moves to Access Path definition (SQL Server Fillsfactor)*	Property of PK Index
SQL Server Primary Key Segment Name		Moves to Access Path definition (SQL Server Segment Name)*	Property of PK Index
SQL Server Segment Name	SQL Server Segment Name	None	
SQL Server Trigger	SQL Server Trigger	None	
State indicator values	State indicator values	None	
SYBASE10 Check Constraint	SYBASE10 Check Constraint	None	
SYBASE10 Check Constraint Name	SYBASE10 Check Constraint Name	None	
SYBASE10 Check Message Name	SYBASE10 Check Message Name	None	
SYBASE10 MaxRowsPerPage		Moves to Access Path definition	Property of PK Index
SYBASE10 Primary Key Fillsfactor		Moves to Access Path definition (SYBASE10 Fillsfactor)*	Property of PK Index
SYBASE10 Primary Key Segment Name		Moves to Access Path definition	Property of PK Index

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
	▨	(SYBASE10 Segment Name)*	
SYBASE10 Segment Name	SYBASE10 Segment Name	None	
Synonym(s)	Synonym(s)	None	
Table Name	Table Name	None	
UNIQUE	▨		REMOVED
Update Trigger Name	Update Trigger Name	None	
User Roles	User Roles	None	
View Name	View Name	None	
View Owner	View Owner	None	
Volume	Volume	None	
WATCOM CONSTRAINT CHECK	WATCOM CONSTRAINT CHECK	None	
Watcom Trigger	Watcom Trigger	None	

* Create Access Path for PK for each entity. Moves PK Index and Constraint properties to Access Path definition.

Foreign Key Component Definition Properties

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
	Relation Name	Fills in with name of relation over which FK is being propagated.	
	From Entity	Fills in with the 'From Entity' specified for the relation line.	
	To Entity	Fills in with the 'To Entity' specified for the relation line.	
	Primary Key Name	Fills in with the name of the Attribute which is a PK in the parent entity (corresponds to FK in child).	
	Role Name	Fills in with the name of the attribute which is a FK in the child entity (corresponds to PK in parent).	
	Physical Role Name	Fills in with the value from the 'Column Name' field in the attribute which is the FK in the child entity.	

Index Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Clustered	Clustered	When creating unique index from Column, makes T if Cluster Index = T	
Database Name	Database Name	None	KEY
DB2 BUFFERPOOL	DB2 BUFFERPOOL	Copies from Table definition when creating PK Index	
DB2 CLOSE	DB2 CLOSE	Copies from Table definition when creating PK Index	
DB2 DEFER	DB2 DEFER	Copies from Table definition when creating PK Index	
DB2 DSETPASS	DB2 DSETPASS	Copies from Table definition when creating PK Index	
DB2 ERASE	DB2 ERASE	Copies from Table definition when creating PK Index	
DB2 FREEPAGE	DB2 FREEPAGE	Copies from Table definition when creating PK Index	
DB2 PCTFREE	DB2 PCTFREE	Copies from Table definition when creating PK Index	
DB2 PRIQTY	DB2 PRIQTY	Copies from Table definition when creating PK Index	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
DB2 SECQTY	DB2 SECQTY	Copies from Table definition when creating PK Index	
DB2 SUBPAGES	DB2 SUBPAGES	Copies from Table definition when creating PK Index	
DB2 USING Name	DB2 USING Name	Copies from Table definition when creating PK Index	
DB2 USING Type	DB2 USING Type	Copies from Table definition when creating PK Index	
dBase Index Expression	dBase Index Expression	None	
Description	Description		REDEFINED
Duplicates	Duplicates	None	
IGNORE_DUP_KEY	IGNORE_DUP_KEY	Fills in with Primary Key IGNORE_DUP_KEY value from table when creating PK Index	
	INFORMIX CONSTRAINT CHECK	Fills in with value from table when creating PK Index	Gray property if PK not = T moved from Entity
Ingres Compression	Ingres Compression	Fills in with value from table when creating PK Index	
Ingres Fills Factor	Ingres Fills Factor	Fills in with value from table when creating PK Index	
Ingres Leaf Fills	Ingres Leaf Fills	Fills in with value	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		from table when creating PK Index	
Ingres Location	Ingres Location	Fills in with value from table when creating PK Index	
Ingres Noduplicates	Ingres Noduplicates	Fills in with value from table when creating PK Index	
Ingres Nonleaf Fills	Ingres Nonleaf Fills	Fills in with value from table when creating PK Index	
Ingres Storage Structure	Ingres Storage Structure	Fills in with value from table when creating PK Index	
▨	Message Name	When creating unique index from Column, copies from Column definition. When Creating PK Index, fills in with Primary Key Message Name value from Table.	moved from Column/Table
ORACLE INITRANS	ORACLE INITRANS	Fills in with ORACLE INDEX INITRANS value from table when creating PK Index	
ORACLE MAXTRANS	ORACLE MAXTRANS	Fills in with ORACLE INDEX MAXTRANS value from table when creating PK Index	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
ORACLE NOSORT	ORACLE NOSORT	None	
ORACLE PCTFREE	ORACLE PCTFREE	Fills in with ORACLE INDEX PCTFREE value from table when creating PK Index	
ORACLE STORAGE PARAMETERS	ORACLE STORAGE PARAMETERS	Fills in with ORACLE INDEX STORAGE PARAMETERS value from table when creating PK Index	
ORACLE TABLESPACE	ORACLE TABLESPACE	Fills in with ORACLE INDEX TABLESPACE value from table when creating PK Index	
Owner Name	Owner Name	None	KEY
▨	Primary Key	Set=T when creating PK Index	NEW
▨	Primary Key Constraint Name	Fills in with value from table when creating PK Index. Use PK Constraint Name if given, else use PK Name	Gray property if PK not = T moved from Entity If SQL Server/Sybase - use DBMS Specific
▨	Primary Key Name	Fills in with value from table when creating PK Index	Gray property if PK not = T moved from Entity
SORTED_DATA	SORTED_DATA	Fills in with Primary Key	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		SORTED_DATA value from table when creating PK Index	
Source Access Path	Source Access Path	None	
	Source Diagram Name	Fills in with name of diagram where symbol appears	NEW - KEY
	SQL Server Constraint Name	Fills in with value from table when creating PK Index. Fills in with value from column when creating Unique Index from column.	PK Constraint name if PK = T only valid if Unique = T moved from Entity/attribute/data element
SQL Server Fillsfactor	SQL Server Fillsfactor	Fills in with value from column when creating unique index. When Creating PK Index, fills in with SQL Server Primary Key Fillsfactor value from table.	
SQL Server Segment Name	SQL Server Segment Name	Fills in with value from column when creating unique index. When Creating PK Index, fills in with SQL Server Primary Key	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		Segment value from table.	
	SYBASE10 Constraint Name	Fills in with value from table when creating PK Index. Fills in with value from column when creating Unique Index from a column.	PK Constraint name if PK = T only valid if Unique = T moved from Entity/attribute/data element
SYBASE10 Fillsfactor	SYBASE10 Fillsfactor	Fills in with value from column when creating unique index. When Creating PK Index, fills in with SYBASE10 Primary Key Fillsfactor value from table.	
SYBASE10 MaxRowsPerPage	SYBASE10 MaxRowsPerPage	Fills in with value from column when creating unique index. Fills in with value from table when creating PK index.	only valid for Sybase 11
SYBASE10 Segment Name	SYBASE10 Segment Name	Fills in with value from column when creating unique index. When Creating PK Index, fills in with SYBASE10 Primary Key	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		Segment value from table.	
Table Name	Table Name	None	KEY relate by changed. New relationship - is indexed by replaces keyed by*
Unique	Unique	Makes T when creating PK or Unique Index	Makes T & gray if PK=T
Word Index	Word Index		

* In SA 3.1/SA 4.0 the index is keyed by the table to which it is attached -> Table symbol connects to constraint line symbol connects to Index symbol uses Index definition. In SA/2001, there is a new relationship -> Table definition is indexed by Index definition. The Index symbol has been removed. During conversion, we must makes sure that the appropriate fields are filled in on both sides of the relationship.

Relationship Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Additional properties	Additional properties	None	
	Allow Nulls	Set from associative properties	NEW
Child Delete		None	REMOVED
Child Delete Error Code		None	REMOVED
Child Delete Error Message		None	REMOVED
Child Insert	Child Insert	None	
Child Insert Error Code	Child Insert Error Code	None	
Child Insert Error Message	Child Insert Error Message	None	
Child Update	Child Update	None	
Child Update Error Code	Child Update Error Code	None	
Child Update Error Message	Child Update Error Message	None	
Constraint Name	Constraint Name	None	
	Description	None	
	From Exclusion Char	None	NEW
	Foreign Keys and Roles	Set from FKFrom and attribute name	NEW
From Entity	From Entity	None	KEY
Growth per period	Growth per period	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Identifying	Identifying	None	
Notes	Notes	None	
Owner	Owner	None	
Parent Delete	Parent Delete	None	
Parent Delete Error Code	Parent Delete Error Code	None	
Parent Delete Error Message	Parent Delete Error Message	None	
Parent Insert		None	REMOVED
Parent Insert Error Code		None	REMOVED
Parent Insert Error Message		None	REMOVED
Parent Update	Parent Update	None	
Parent Update Error Code	Parent Update Error Code	None	
Parent Update Error Message	Parent Update Error Message	None	
Reverse Phrase	Reverse Phrase	None	
Role		If specified, use to set Foreign Keys and Roles.	
Role Prefix		If specified, use to set Foreign Keys and Roles.	
SYBASE10 Message Name	SYBASE10 Message Name	None	
Synonym(s)	Synonym(s)	None	
	To Cardinality	Set from associative	NEW

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
		properties	
To Entity	To Entity	None	KEY
	To Exclusion Char	None	NEW
User Roles	User Roles	None	

Super-sub Relation Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
	Additional properties	Copies from Relation definition	
	Description	Copies from Relation definition	
	Foreign Keys and Roles	Set from FK From and attribute name	NEW
	From Entity	Copies from Relation definition	KEY
	From Exclusion Char	None	
	Growth per period	Copies from Relation definition	
	Notes	Copies from Relation definition	
	Owner	Copies from Relation definition	
	Synonym(s)	Copies from Relation definition	
	To Entity	Copies from Relation definition	KEY
	To Exclusion Char	None	
	User Roles	Copies from Relation definition	

Non-specific Relation Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
	Additional properties	Copies from Relation definition	
	Allow Nulls	Set from associative properties	
	Description	Copies from Relation definition	
	From Entity	Copies from Relation definition	KEY
	From Exclusion Char	None	
	Growth per period	Copies from Relation definition	
	Notes	Copies from Relation definition	
	Owner	Copies from Relation definition	
	Reverse Phrase	Copies from Relation definition	
	Synonym(s)	Copies from Relation definition	
	To Cardinality	Set from associative properties	
	To Entity	Copies from Relation definition	KEY
	To Exclusion Char	None	
	User Roles	Copies from Relation definition	

Table Definition Properties

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Caption	Caption	None	
CLUSTER		Moves to Index definition *	Property of PK Index
Cluster Primary Index		Moves to Index definition *	Property of PK Index
Create Trailer	Create Trailer	None	
Data Font Name	Data Font Name	None	
DATABASE		None	REMOVED
Database Comment	Database Comment	None	
Database Name	Database Name	None	KEY
DB2 AUDIT	DB2 AUDIT	None	
DB2 BUFFERPOOL		Moves to Index definition *	Property of PK Index
DB2 CLOSE		Moves to Index definition *	Property of PK Index
DB2 Data Capture	DB2 Data Capture	None	
DB2 DSETPASS		Moves to Index definition *	Property of PK Index
DB2 EDITPROC	DB2 EDITPROC	None	
DB2 ERASE		Moves to Index definition *	Property of PK Index
DB2 FIELDPROC		None	REMOVED
DB2 FREEPAGE		Moves to Index definition *	Property of PK Index
DB2 OBID	DB2 OBID	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
DB2 PCTFREE		Moves to Index definition *	Property of PK Index
DB2 PRIQTY		Moves to Index definition *	Property of PK Index
DB2 SECQTY		Moves to Index definition *	Property of PK Index
DB2 SUBPAGES		Moves to Index definition *	Property of PK Index
DB2 TABLESPACE	DB2 TABLESPACE	None	
DB2 USING Name		Moves to Index definition *	Property of PK Index
DB2 USING Type		Moves to Index definition *	Property of PK Index
DB2 VALIDPROC	DB2 VALIDPROC	None	
dBase Database Name		None	REMOVED
dBase Naming Prefix		Fills in Naming Prefix field	REMOVED
Delete Trigger Name	Delete Trigger Name	None	
Description	Description	None	
Function	Function	None	
Headings Font Name	Headings Font Name	None	
	Indices	Fills in with names of all Index definitions that are Keyed by the table.	NEW
INFORMIX CONSTRAINT		Moves to Index definition *	Property of PK Index

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
CHECK			
Informix DBSPACE	Informix DBSPACE	None	
Informix EXTENT	Informix EXTENT	None	
Informix Fragment	Informix Fragment	None	
Informix Lock Mode	Informix Lock Mode	None	
Informix NEXT	Informix NEXT	None	
INFORMIX Trigger	INFORMIX Trigger	None	
Ingres Compression		Moves to Index definition *	Property of PK Index
Ingres Fills Factor		Moves to Index definition *	Property of PK Index
Ingres Leaf Fills		Moves to Index definition *	Property of PK Index
Ingres Location		Moves to Index definition *	Property of PK Index
Ingres Noduplicates		Moves to Index definition *	Property of PK Index
Ingres Nonleaf Fills		Moves to Index definition *	Property of PK Index
Ingres Storage Structure		Moves to Index definition *	Property of PK Index
Insert Trigger Name	Insert Trigger Name	None	
InterBase CONSTRAINT CHECK	InterBase CONSTRAINT CHECK	None	
InterBase Constraint Check Name	InterBase Constraint Check Name	None	
InterBase Trigger	InterBase Trigger	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Labels Font Name	Labels Font Name	None	
Naming Prefix		Apply to column name	REMOVED
ORACLE CONSTRAINT CHECK	ORACLE CONSTRAINT CHECK	None	
ORACLE Constraint Check Name	ORACLE Constraint Check Name	None	
ORACLE INDEX INITRANS		Moves to Index definition (ORACLE INITRANS) *	Property of PK Index
ORACLE INDEX MAXTRANS		Moves to Index definition (ORACLE MAXTRANS) *	Property of PK Index
ORACLE INDEX PCTFREE		Moves to Index definition (ORACLE PCTFREE) *	Property of PK Index
ORACLE INDEX STORAGE PARAMETERS		Moves to Index definition (ORACLE STORAGE PARAMETERS) *	Property of PK Index
ORACLE INDEX TABLESPACE		Moves to Index definition (ORACLE TABLESPACE) *	Property of PK Index
ORACLE INITRANS	ORACLE INITRANS	None	
ORACLE MAXTRANS	ORACLE MAXTRANS	None	
ORACLE PCTFREE	ORACLE PCTFREE	None	
ORACLE PCTUSED	ORACLE PCTUSED	None	
ORACLE STORAGE	ORACLE STORAGE	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
PARAMETERS	PARAMETERS		
ORACLE TABLESPACE	ORACLE TABLESPACE	None	
Oracle Trigger	Oracle Trigger	None	
OWNER		None	REMOVED
Owner Name	Owner Name	None	KEY
Primary Index Name		When specified, use as "Name" when creating Index definition for PK index *	REMOVED -Is the name of the PK Index created by SA
Primary Key Constraint Name		Moves to Index definition *	Property of PK Index
Primary Key IGNORE_DUP_KEY		Moves to Index definition (IGNORE_DUP_KEY)*	Property of PK Index
Primary Key Message Name		Moves to Index definition (Message Name)*	Property of PK Index
Primary Key Name		Moves to Index definition (Primary Key Constraint Name)*	Property of PK Index
Primary Key SORTED_DATA		Moves to Index definition (SORTED_DATA)*	Property of PK Index
Progress Create Trigger File	Progress Create Trigger File	None	
Progress Delete Trigger File	Progress Delete Trigger File	None	

SA 3.1/SA 4.0	SA/2001	Conversion	Comments
Progress Desc	Progress Desc	None	
Progress Dump Name	Progress Dump Name	None	
Progress Valexp	Progress Valexp	None	
Progress Valmsg	Progress Valmsg	None	
Progress Write Trigger File	Progress Write Trigger File	None	
Record Format	Record Format	None	
Source Class	Source Class	None	
	Source Diagram Name	Fills in with name of diagram that includes Table symbol. If table is included on more than one diagram, create duplicate definitions.	NEW - KEY
Source Entity	Source Entity	Inherit all entity properties that are not specified for Table. Create Columns based on entity attributes.	REDEFINED
Source View		Create columns based on selected attributes of entities that are specified as being included in the view.	REMOVED
SQL Server Check Constraint	SQL Server Check Constraint	None	
SQL Server Check	SQL Server Check	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Constraint Name	Constraint Name		
SQL Server Primary Key Fillsfactor		Moves to Index definition (SQL Server Fillsfactor)*	Property of PK Index
SQL Server Primary Key Segment Name		Moves to Index definition (SQL Server Segment Name)*	Property of PK Index
SQL Server Segment Name	SQL Server Segment Name	None	
SQL Server Trigger	SQL Server Trigger	None	
SYBASE10 Check Constraint	SYBASE10 Check Constraint	None	
SYBASE10 Check Constraint Name	SYBASE10 Check Constraint Name	None	
SYBASE10 Check Message Name	SYBASE10 Check Message Name	None	
SYBASE10 MaxRowsPerPage		Moves to Index definition *	Property of PK Index
SYBASE10 Primary Key Fillsfactor		Moves to Index definition (SYBASE10 Fillsfactor)*	Property of PK Index
SYBASE10 Primary Key Segment Name		Moves to Index definition (SYBASE10 Segment Name)*	Property of PK Index
SYBASE10 Segment Name	SYBASE10 Segment Name	None	
UNIQUE			REMOVED
Update Trigger	Update Trigger	None	

SA 3.1/SA 4.0	**SA/2001**	**Conversion**	**Comments**
Name	Name		
View Name	View Name	None	
View Owner	View Owner	None	
WATCOM CONSTRAINT CHECK	WATCOM CONSTRAINT CHECK	None	
Watcom Trigger	Watcom Trigger	None	

* Create Index for PK for each table. Moves PK Index and Constraint properties to Index definition.

9

Business Modeling Name Changes

Introduction

This chapter provides details of what diagram and definition names were changed within the business modeling provided by SA/Catalyst.

SA/Catalyst Name Changes

A number of name changes have been made to diagrams, definitions, and reports in the product to replace the name Catalyst with the term Business Enterprise Modeling.

Changes to USRPROPS.TXT

Users who have modified Catalyst properties through USRPROPS should make the appropriate name changes to their USRPROPS based on the information provided in this chapter.

SA/Catalyst Diagram Name Changes

Old Name	New Name
Application Architecture	System Architecture
Application Area Map	System Area Map
Application Context	System Context
Application/Subsys. Structure	System/Subsystem Structure
Conceptual Flow	Business Concept
Flow	
Functional Decomposition	
Logical Decision Making Chart	Decision Chart
Logical Process Hierarchy	Process Hierarchy
Network Concept	
Process Flow	Process Chart
Process Map	
Process Oriented Decomp.	Process Decomposition
Relationship Map	
Logical Organizational Chart	Organization Chart

SA/Catalyst Definition Name Changes

Old Definition Name	**New Definition Name**
Decision	
Sequence	
Flow Break	
Result	
Iteration	
Catalyst Elem. Business Proc.	Elementary Business Process
Catalyst Application	Application
Catalyst Location Type	Location
Catalyst Role	Role
Catalyst Technology Capability	Technology
Catalyst Derived Logical Proc.	Derived Logical Process
Catalyst Developer	Developer
Catalyst Package External	Package External
Catalyst Competency	Competency
Catalyst Team	Team
Catalyst Subsystem	Subsystem
Catalyst Test Category	Test Category
Catalyst Type	Type
Organizational Unit	
Application Area	
Application Interface Defn.	
Balanced Scorecard	
Cost Driver	
Processing Node	
Process Object	
Conceptual Node	
Activity-Based Cost	

Old Definition Name	**New Definition Name**
EBP Measure and Goal	Business Process Measure and Goal
Supplier Input	
Benchmark Company	
Performance Measure	
Customer Requirement	
Customer Needs Summary	
Competitors Profile	
Process Thread	
Business Process Direction	
Business Profile	
Business Segment	
Products/Services Profile	
DFXCATEGORY = 342 Category	
Customers Profile	
Suppliers Profile	
Business Forces Summary	
Business Enterprise Direction	
Critical Business Issue	
EBP/ENTITY	
EBP/ROLE	
USER MATRIX	
REQUIREMENT/PROCESS	
DIMENSION/PRODUCT	
Product Dimension	
Product	
Catalyst Behavior	Behavior
BEHAVIOR/STAKEHOLDER	
Cultural Characteristic	
CULTURE/STAKEHOLDER	
RECEIVER/SENDER	
WORKGROUP/ROLE	
Catalyst Message Category	Message Category

Old Definition Name	New Definition Name
Communication Vehicle	
Organization Objective	
BUSINESS/ORGANIZATION	
ORGUNIT/ORGUNIT	
Location Model View	
LOCATION/APPLICATION	
LOCATION/ENTITY	
Action State	
Signal Transition	
ORACLE View	
Catalyst Object	Simulation Object

10

IBM Support

Contacting IBM Rational Software Support

Support and information for Telelogic products is currently being transitioned from the Telelogic Support site to the IBM Rational Software Support site. During this transition phase, your product support location depends on your customer history.

Product Support

- If you are a heritage customer, meaning you were a Telelogic customer prior to November 1, 2008, please visit the System Architect Support Web site.

  Telelogic customers will be redirected automatically to the IBM Rational Software Support site after the product information has been migrated.

- If you are a new Rational customer, meaning you did not have Telelogic-licensed products prior to November 1, 2008, please visit the IBM Rational Software Support site.

Before you contact Support, gather the background information that you will need to describe your problem. When describing a problem to an IBM software support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, or messages that are related to the problem?
- Can you reproduce the problem? If so, what steps do you take to reproduce it?
- Is there a workaround for the problem? If so, be prepared to describe the workaround.

Other information

For Rational software product news, events, and other information, visit the IBM Rational Software Web site.

11

Appendix

Introduction

This Appendix chapter contains Notices and Trademarks.
.

Notices

Trademarks