SAS® AppDev Studio™ 3.3 Eclipse Plug-ins

Migration Guide

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2007. *SAS® AppDev Studio™ 3.3 Eclipse Plug-ins: Migration Guide*. Cary, NC: SAS Institute Inc.

SAS® AppDev Studio™ 3.3 Eclipse Plug-ins: Migration Guide

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st electronic book, December 2007

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at **support.sas.com/pubs** or call 1-800-727-3228.

Table of Contents

Introduction

The purpose of this document is to provide users of previous releases of SAS AppDev Studio software with guidance on upgrading to this new release of the product. The document includes updated system requirements for the product, information on how to migrate existing development projects forward, guidelines for getting the most from the new features, and enumeration of obsolete or significantly changed functionality.

New and Noteworthy Features

SAS AppDev Studio 3.3 now supports Eclipse 3.3 and Eclipse Web Tools Platform 2.0. For more information about the Eclipse platform, see the Eclipse Foundation Web site (http://www.eclipse.org). And see for details about the new features and capabilities provided by Eclipse 3.3 and Eclipse Web Tools Platform 2.0, see the Eclipse IDE for Java EE Developers Web site (http://www.eclipse.org/downloads/moreinfo/jee.php).

SAS AppDev Studio 3.3 contains the following new and noteworthy features:

- New and improved "cheat sheets" that guide users in configuring the SAS AppDev Studio Eclipse Plug-ins, setting up their development workspaces, and defining their Web application servers for use within the development environment.
- Improved application templates for portlet, SAS Management Console Plug-in, and Data Integration Studio Plug-in development.
- New support for importing external Web applications.
- Enhanced support for managing JDBC connections in the DataBean Wizard and the JDBC Web application template.
- Integration with SAS hot fix installers so that the development environment and application templates are automatically updated when new hot fix content is installed.

Changes to the SAS AppDev Studio System Requirements

Supported Versions of SAS Software

SAS AppDev Studio 3.3 supports SAS 9.1.3 Service Pack 4 software.

Java Platform Requirements

SAS AppDev Studio 3.3 requires Java 2 Standard Edition (J2SE) 1.4.2_09 or higher for both application development and execution of applications at run time. For execution of Web applications, SAS AppDev Studio 3.3 supports the Web browsers, application servers, and Java run times as defined on the *Third Party Software for SAS 9.1.3 Foundation with Service Pack 4* page (http://support.sas.com/resources/thirdpartysupport/v913sp4/index.html).

Note: In addition to the system requirements listed above, SAS AppDev Studio 3.3 Eclipse Plug-ins also requires the minimum Java release required by Eclipse as documented on the Java Runtime Environment page (http://www.eclipse.org/downloads/moreinfo/jre.php). Specifically, Eclipse IDE for Java EE Developers version 3.3 requires Java 5.

Eclipse Platform Requirements

The SAS AppDev Studio 3.3 Eclipse Plug-ins requires the Eclipse IDE for Java EE Developers version 3.3. This software must be installed on your system as a pre-requisite to the SAS AppDev Studio 3.3 Eclipse Plug-ins install. See the *SAS AppDev Studio Java Components 3.3 Install Instructions* for further information on the install process. A download link for the supported release of Eclipse can be found on the SAS Third-party Downloads page (http://support.sas.com/resources/thirdpartysupport/v913sp4/index.html#eclipse).

See the *SAS AppDev Studio Java Components 3.3 System Requirements* document for complete system requirements information. A hard copy of the document is provided in the product packaging and it can also be viewed online at the SAS AppDev Studio Developer's Site (http://support.sas.com/rnd/appdev).

Changes to the SAS AppDev Studio Product Bundle

The SAS AppDev Studio Server Side Catalogs are no longer included in the SAS AppDev Studio product bundle, because SAS Release 8.2 is no longer supported. These catalogs, which were provided to support connections to certain SCL based-data objects on the SAS 8.2 server, have been deprecated in favor of using more standard and scalable data connection technologies such as JDBC (for relational data) and the SAS 9 Java OLAP interfaces (for OLAP data).

This is the last release of the SAS AppDev Studio Eclipse Plug-ins that will provide the **Import WebAF Project** feature. If you have old webAF 3.*x* projects that you would like to import into a SAS AppDev Studio Eclipse Plug-ins development workspace, you must do so with this release of the software if you intend to maintain the project going forward.

Project Conversion Considerations

General Project Conversion Considerations

SAS AppDev Studio 3.3 can be installed side-by-side with the previous release. When using SAS AppDev Studio 3.3 side-by-side with SAS AppDev Studio 3.2, it is strongly recommended that you specify different project workspaces for the two installs. Then, rather than being updated in place, old projects can be copied out of the SAS AppDev Studio 3.2 workspace and into the SAS AppDev Studio 3.3 workspace.

Note: If you open a SAS AppDev Studio 3.2 project directly from SAS AppDev Studio 3.3, the project will be updated in place and will no longer be editable using SAS AppDev Studio 3.2.

Projects created using the webAF software provided with SAS AppDev Studio 3.0 can be imported using the *webAF Project Import* feature of the SAS AppDev Studio Eclipse Plug-ins (by selecting **File > Import > Other > webAF Project Import**). Note that SAS AppDev Studio 3.3 only supports the migration of webAF projects from SAS AppDev Studio 3.0. It will not import projects from earlier versions.

During the import process, the Java source files that were generated by webAF 3.0 are modified slightly to remove artifacts required by the webAF 3.0 source editor. These changes are in code comments only and have no impact on the content of the compiled Java class files.

In webAF 3.0, it was possible for files to be added to projects without webAF actually being notified of their existence (for example, by adding them to the project build script directly rather than by using the *Add File to Project* command). In such cases, the *webAF Project Import* feature in the SAS AppDev Studio 3.3 Eclipse Plug-ins might fail to include these files during the import process. Make sure that these files have

been expressly added to your project in webAF 3.0 using the Add File to Project command before importing the project into the SAS AppDev Studio 3.3 Eclipse Plug-ins.

SAS AppDev Studio 3.0 included the ability to create applets and applications using a visual drag-and-drop builder. This allowed for a WYSIWYG experience when designing your interface. This technology is not yet provided in the SAS AppDev Studio Eclipse Plug-ins. Visual editing of Java user interfaces within Eclipse is provided by the Eclipse Visual Editor Project (http://www.eclipse.org/vep/WebContent/main.php).

See below for details on specific project conversion scenarios.

Application Project Conversion

When application projects are imported into the SAS AppDev Studio 3.3 Eclipse Plug-ins, the source files are modified to remove bookmark artifacts that were required by the source editor in webAF 3.0. In addition, two JAR files (`sas.ads.core.jar` and `sas.ads.misc.jar`) are added to the classpath defined by the SAS Versioned Jar Repository. These JAR files are added because they contain component classes that were used when webAF 3.0 generated the application code. At this time, SAS is discontinuing the use of the classes contained in these two JAR files in favor of newer components. Since these JAR files will not be available in future versions of SAS AppDev Studio, you should migrate away from using them.

When an application project was created in webAF 3.0, two Java classes were created by default -- `<projectName>.java` and `<projectName>Main.java`. The application implementation code resided in `<projectName>Main.java` while the application entry point was implemented in `<projectName>.java`. In order to test your application within the SAS AppDev Studio 3.3 Eclipse Plug-ins, you will need to run `<projectName>.java`.

Web Application Project Conversion

Note: Before importing a webAF Web application project, make sure that you are using a workspace path that does not include spaces. You might encounter problems testing the Web application if the workspace path contains spaces.

To convert a webAF Web application project to the AppDev Studio Eclipse Plug-ins you simply import the project using the webAF Project Import feature (by selecting **File > Import > Other > webAF Project Import**). The import process automatically converts and upgrades the project to be a SAS Web Application Project. This project type is based upon the Dynamic Web Project provided in the Eclipse Web Tools Platform. Because the Eclipse Web Tools Platform defines specific locations where content under development must be stored, the conversion from a webAF 3.0 Web Application project to a SAS Web Application project in SAS AppDev Studio 3.3 involves changes to your Web application project structure.

First, JAR files included in your Web application are handled as follows:

- Certain SAS common JAR files are not copied from the webAF project. Instead, they are replaced with references to the JAR files in the SAS Versioned Jar Repository by configuring them in the SAS Repository of the Eclipse project.
- All other JAR files are copied to the `/WEB-INF/lib` directory under the specified Web Content Folder of the Eclipse project.

In addition, the conversion process copies Java source files to the new project using the following rules:

- Java source files in the `WEB-INF/classes` directory of the webAF project are copied to the specified Java Source Folder of the Eclipse project. Associated class files are not copied since they will be built by Eclipse and included in the `WEB-INF/classes` of the Web application. During this process, any Foundation Services context listener detected is replaced with an upgraded version that uses newer support classes provided in SAS 9.1.3 SP4. Additionally, any Build Frame servlets that are copied are updated to remove comment blocks required by the webAF 3.0 source code generator. These changes are made in code comments only and have no impact on the content of the compiled Java class files.

- Java source files in the content portion of the Web application in the webAF project are copied to the specified *WebContent* folder within the Eclipse project. Associated class files are not copied. Note that Java files stored in the *WebContent* folder are not visible in Java-aware navigator views, such as the Package Explorer or Project Explorer. They are visible in the simple Navigator view. In addition, corresponding class files are not built and included in the Web application content. This is because the Eclipse Web Tools Platform currently does not support building Java classes (such as applet classes or Java Web Start classes) in the content portion of a Web application.

  Note: A workaround is described in the *Building Java Into the Content Portion of a Web Application Project* page (http://support.sas.com/rnd/appdev/doc/JavaWebContent.html).

- Java files in the webAF project, but outside of the Web application, are copied to the same relative path under the root of the Eclipse project. Associated class files are not copied. Like Java source files under the content portion of the Web application, these files are not in a specified Java source folder, so they are not visible in Java-aware navigator views and are not built into classes. If these Java files are built into a JAR file that is used by the containing Web application project, then it is best to move the classes into a separate Java project. Then, to use the output of this new Java project in your Web application, open the Properties dialog box for the SAS Web Application project and use the J2EE Module Dependencies page to add the separate project as a dependency.

Next, static Web application content (for example, HTML files, images, and so on) will be migrated as follows:

- Static content in the webAF project that is part of the Web Infrastructure Kit is not copied to the Eclipse project. The static content for the SAS 9.1.3 Service Pack 4 version of the Web Infrastructure Kit is copied into to the Eclipse project instead.

- All other Web application content is copied to the *WebContent* folder of the Eclipse project using the same relative path within the Web application.

Finally, the remaining files in the webAF Web application project are copied in the following manner:

- Files that are located under the project folder but outside of the Web application content area are copied to the same relative location under the imported project. Should that location happen to be under `build/classes`, then these files should be moved to a different folder. This folder is the default Java output folder whose contents are deleted if you execute **Project > Clean** on this project.

- External files that have been inserted into the webAF project are copied into the imported project under a folder named `external_files` with subfolders that duplicate the absolute path of the file from the root of the drive.

During the conversion process, a file called `launchParameters.txt` is created in the root folder of the project. This file contains Java start-up arguments needed by the project. If you had defined custom

server start-up arguments in your webAF Web application project, some of those arguments are included for you automatically, as follows:

- Java heap size arguments will be the greater of the SAS AppDev Studio 3.3 Web application template defaults or those specified in the imported project.
- Values for known server-related system properties are imported.

Any additional custom arguments or changes that were made to the startup arguments in your webAF Web application project need to be transferred manually from the imported project. If you later update this new project with content from a Web application template, the `launchParameters.txt` file might get overwritten. If you have custom start-up arguments, you might want to save them in a separate file. For details about how to apply these start-up arguments to your application server, see "Configuring the Web Application Server Launch Settings" in the *SAS Web Application Templates* section of the *SAS AppDev Studio User's Guide* on the SAS AppDev Studio developer's site (http://support.sas.com/rnd/appdev).

As a final step in the conversion process, additional metadata is created for the project to support its use of the SAS Repository and for compatibility with the SAS AppDev Studio 3.3 Web application templates.

Importing SAS AppDev Studio 3.2 and 3.2.x SAS Web Application Projects

The *Import a SAS AppDev Studio 3.2 or 3.2.x SAS Web Application Project* cheat sheet is provided for importing SAS Web Application projects from SAS AppDev Studio 3.2 and 3.2.x workspaces. The steps in this cheat sheet update the project to work with the newer version of the Eclipse Web Tools Platform, address changes in the SAS AppDev Studio workspace setup, and fix issues with respect to the project's new location. This cheat sheet appears at the end of the *SAS AppDev Studio New Workspace* Setup cheat sheet. To run the cheat sheet separately, follow these steps:

1. Select **Help > Cheat Sheets** to open the Cheat Sheet Selection dialog box.
2. In the tree, expand **SAS App Dev Studio** and select **Import SAS AppDev Studio 3.2 or 3.2.x Web Project**.
3. Click **OK** to open the cheat sheet.

Click the icon at the bottom of the *Introduction* section to start, or restart, the cheat sheet.

Project Deployment Considerations

There are a number of important concepts that should be understood prior to deploying applications created with the SAS AppDev Studio 3.3 Eclipse Plug-ins. These considerations are described below.

SAS Versioned Jar Repository

In past releases, the SAS AppDev Studio installation copied SAS library JAR files to multiple locations, sometimes creating duplicate copies, in order to support different usage scenarios such as application development, Web application development, and JSASNetCopy downloads. The project classpath was configured in the Project Properties dialog box; either the SAS AppDev Studio classloader was turned on or items were added to the classpath manually. In the case of Web application projects, the SAS library JAR files were also copied into the Web application file structure under each project. These limitations made it difficult to develop against different versions of the same JAR file or to know which JAR files were not needed for a given project.

One of the goals of the SAS Versioned Jar Repository is to eliminate the duplication of JAR files. The SAS Java Project type provided by the SAS AppDev Studio 3.3 Eclipse Plug-ins enables a project's classpath to reference SAS Versioned Jar Repository content for easy building and testing of applications which use common libraries from SAS. The SAS Repository Jar Selector provides the ability to specify the project classpath as a set of dependencies on common JAR files stored in the SAS Versioned Jar Repository. The project classpath can be configured as a particular subset of JAR files or to target development against specific versions of JAR files. You can now develop projects that reference a small set of the SAS library JAR files instead of the entire library. The Organize SAS Imports editor command and the SAS Repository Quick Fix automatically manage your project dependencies by adding import statements to your Java source files and updating the project classpath as you use SAS library classes in your code.

When the time comes to deploy your application, you can use the Application Export wizard to copy the specific SAS library JAR files that the project uses from the SAS Versioned Jar Repository to the export area. Or, if the SAS Versioned Jar Repository is already installed on the destination machine (see **Installing the SAS Versioned Jar Repository** below), the export wizard has options for enabling exported applications to reference the SAS Versioned Jar Repository directly.

SAS Launcher

The SAS Launcher is a custom ClassLoader installed as the Java application ClassLoader. Various SAS applications use the launcher (possibly with an .exe wrapper that displays a splash screen) to isolate the application classpath from JAR file conflicts (for example, it limits visibility into the Java Runtime Environment extensions directory). The SAS Launcher also provides a few convenience features like being able to specify that directories of JAR files be added to the application classpath.

In SAS AppDev Studio 3.2 and later, the SAS Launcher has been updated to use the SAS Versioned Jar Repository, which enable applications to use a simple configuration (config) file to add SAS Versioned Jar Repository content to their application classpath. The SAS Launcher is now also available for use by applications developed within SAS AppDev Studio.

SAS Repository Locator

Using the SAS Launcher typically requires adding several command-line parameters to the invocation of a Java program. Java system property definitions are used to get the custom ClassLoader installed as the Java application ClassLoader, and to optionally specify a config file that requests content from the SAS Versioned Jar Repository. In addition, the classpath defined on the Java command line is used for loading the custom ClassLoader class. That classpath must therefore include a reference to the `sas.launcher.jar` file that is installed in the SAS Versioned Jar Repository. Rather than mixing that classpath together with the real application classpath, the classpath to use for the launched application is defined in a separate system property. The SAS Java Project Export Wizard can generate a batch file containing a command line with all of the necessary parameters when exporting an application.

The SAS Repository Locator provides an alternative to the extended Java command line normally required to use the SAS Launcher. By using the SAS Repository Locator is it possible to preserve `java -jar <application jar>` semantics, which include double-clicking a JAR file in Windows Explorer to launch an application.

The SAS Repository Locator is an optional part of the SAS Java application export process. If the SAS Repository Locator option is selected during export, then the SAS Repository Locator code is added to the application JAR. The JAR manifest is also updated so that the SAS Repository Locator code is invoked when the application JAR is executed. Additionally, any JAR files in the SAS Versioned Jar Repository that the application depends upon are recorded in the application JAR manifest.

As a result, when the application JAR is run with `java -jar` (for example, double-clicking in Windows Explorer), the SAS Repository Locator main method is run. The SAS Repository Locator code determines the location of the SAS Versioned Jar Repository on the machine and of the `sas.launcher.jar` file within it. The SAS Launcher custom ClassLoader from that JAR is created and then used to replace the original Java application ClassLoader. The SAS Launcher code then loads and runs the original application, using any recorded dependencies on SAS Versioned Jar Repository content.

Thus, the SAS Repository Locator enables applications that would normally be able to use the `java -jar` invocation pattern to keep that simplicity, while making use of the SAS Versioned Jar Repository on the installation machine and of the SAS Launcher.

Application Project Export

In the past, SAS AppDev Studio supported exporting of Java applications via the Package Wizard. This wizard enabled you to pick which resources would be included in the exported application and then built the application JAR file. Then it was your responsibility to handle the remainder of the deployment process.

The SAS AppDev Studio 3.3 Eclipse Plug-ins leverage the new SAS Versioned Jar Repository to make the rest of the deployment process easier. Similar to the Package Wizard, you can use the Eclipse Export wizard to export your SAS Java Project. In this wizard, you select which project resources should be included in the application JAR file. You can also choose whether or not to deploy the application against the SAS Versioned Jar Repository. If you choose to deploy against the SAS Versioned Jar Repository (the default), then you can take the application JAR file and copy it to any other machine that has the SAS Versioned Jar Repository installed (see **Installing the SAS Versioned Jar Repository** below). When executed, the application JAR automatically locates the SAS Versioned Jar Repository and gives your program access to the SAS libraries. If you choose not to deploy against the SAS Versioned Jar Repository, then the required library JAR files is copied out to the same location as your application JAR file. All of the resulting JAR files can then be copied to the deployment location for execution.

Installing the SAS Versioned Jar Repository

The SAS Versioned Jar Repository can be installed and used on remote machines. To install the repository for use by the SAS Application Launcher and the SAS Repository Locator, follow these steps:

1. Copy the SAS Versioned Jar Repository from the development machine where SAS AppDev Studio is installed to the target machine. The SAS Versioned Jar Repository can be found in the main SAS AppDev Studio directory. For example, the default location is `C:\Program Files\SAS\SASAppDevStudio\3.3\VersionedJarRepository`.

2. Open a command prompt on the target machine.

3. Navigate to the `eclipse\plugins` directory under the `VersionedJarRepository` root directory.

4. Execute `java -jar sas.launcher_3.3.0.jar -install` to install the SAS Versioned Jar Repository.

Note that you must have administrator or root privileges in order to install the SAS Versioned Jar Repository successfully.

Web Application Project Export

In webAF, the Package Wizard created the WAR file for Web applications. In the SAS AppDev Studio Eclipse Plug-ins, this is handled by the Eclipse Web Tools Platform extension to the Eclipse Export feature. To create the WAR file, follow these steps:

1. Select **File > Export** to open the Export wizard.
2. Select **WAR file** and click **Next**.
3. Select the Web module you want and then specify the **Destination**.
4. (Optionally) Select the **Export source files** and **Overwrite existing file** check boxes.
5. Click **Finish**.

This WAR file can be deployed to any application server, servlet container, or both that provides full support for the Servlet 2.3/JSP 1.2 standard using JDK 1.4.0 or higher. However, only certain combinations of system hardware, JDK, and server have been tested by SAS and are considered supported for Web applications from SAS Web Application Projects. If you are deploying to a production environment, you should consult the *Third Party Software for SAS 9.1.3 Foundation with Service Pack 4* page (http://support.sas.com/resources/thirdpartysupport/v913sp4/index.html), specifically the JDK and Application Servers sections, for details about the supported combinations and the server security fixes that are expected to be installed. Additional Web application deployment information is available on the SAS AppDev Studio Developer's Site (http://support.sas.com/rnd/appdev).

Portlet Project Export

In webAF, the project build script contained a "package-par" rule that built the Portlet ARchive (PAR) file. In the SAS AppDev Studio Eclipse Plug-ins, the export process is now handled by an Ant build file in the project. This build file is created for the user either by the new content template wizard when a new portlet is added to a project or by the webAF import wizard when an existing webAF portlet project is imported.

To create the PAR file, follow these steps:

1. Open the Navigator view.
2. Select the `{portlet-name}ParBuild.xml` file in top level of the project.
3. Select a file entry, then open the pop-up menu and select **Run As > Ant Build**.

This Ant build script packages any generated classes and static content files into an archive file in the root of project.

Note: You might need to refresh the navigator view to get Eclipse to recognize the new file.

If you need to add any additional content to the archive, edit the Ant XML document and specify that the other content to be included.

To deploy the completed archive file, copy the output file to your Portal deployed portlets directory.

General Usage Notes

Support for SAS 9.1.3 Hot Fixes

SAS AppDev Studio 3.3 includes new hot fixes in addition to those contained in the 322APPDEV01 hot fix. The new hot fixes include a fix for the issue covered in Problem Note 20591 (http://support.sas.com/kb/20/591.html). Since this is a security issue, it is recommended that you upgrade all SAS Web Application projects with the latest hot fixes.

To apply the latest hot fixes to any SAS Web Application Projects that you have already created, follow these steps:

1. Right-click on the project in the Project Explorer, Package Explorer, or Navigator view and select **Properties**.
2. Select the **Project Facets** properties page.
3. Click **Modify Project**.
4. Click on the Version for the "SAS Web Module with WIK" facet to make its combo box appear.
5. In the combo box, select **9.1.3.019002**.
6. Click **Finish** and then click **OK**.

Using Old JAR Files in New SAS Projects

All new SAS Web Application Projects, as well as any SAS Java Projects, will automatically pick up the latest version of the JAR files. If you want to use the latest JAR file, you do not need to make any changes.

If you want to continue using the old JAR files in a new SAS Web Application Project, follow the steps in the “Support for SAS 9.1.3 Hot Fixes” section above to upgrade a SAS Web Application Project, using **9.1.3.019000** or **9.1.3.019001** as the version in step 5 above. Version 9.1.3.019000 gives you the original SAS AppDev Studio 3.2 JAR files and static WIK content. Version 9.1.3.019001 gives you the original JAR files and static WIK content, plus the hot fixes in the 322APPDEV01 hot fix.
If you want to continue using the old JAR files in a new SAS Java Project, you will have to edit the SAS Repository settings using the following steps:

1. Open the project in Eclipse.
2. Expand the project in the package explorer, right-click on **SAS Repository**, and select **Configure**.
3. Select **Add Dependencies**.
4. In the list box, scroll down and select **sas.iquery.metadata**.
5. In the **Match Rule** list box, select **Exact**.
6. In the **Version** window, select **9.1.3.019000**.
7. Repeat steps 4 through 6 for the following entries, if present:

   sas.iquery.dataservices
   sas.swing
   sas.swing.remote
   sas.dataexplorer
   sas.portal
   sas.servlet
   sas.storage

sas.web.framework
sas.ads.iqueryutil

8. Click **Finish**.

To use the 322APPDEV01 hot fix JAR files, select **9.1.3.019001** instead of 9.1.3.019000 for the following entries:

sas.iquery.metadata
sas.iquery.dataservices
sas.swing
sas.dataexplorer
sas.servlet
sas.ads.iqueryutil

Your Turn

We want your feedback.

- If you have comments about this book, please send them to **yourturn@sas.com**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **suggest@sas.com**.