THE POWER TO KNOW.

SAS® AppDev Studio™ 4.4 Eclipse Plug-ins

Migration Guide

Third Edition

SAS® Documentation

The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2014. *SAS® AppDev Studio™ 4.4 Eclipse Plug-ins: Migration Guide, Third Edition*. Cary, NC: SAS Institute Inc.

SAS® AppDev Studio™ 4.4 Eclipse Plug-ins: Migration Guide, Third Edition

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513-2414.

December 2014

Contents

About This Book . . . . . *vii*
What's New in SAS AppDev Studio 4.4 . . . . . *ix*
Accessibility Features of SAS AppDev Studio . . . . . *xiii*

Chapter 1 • Migrating, Importing, and Upgrading SAS Projects **1**
Migrating SAS Projects 1
Importing a Project into SAS AppDev Studio 4.4 1
Upgrading SAS Projects 2

Chapter 2 • Updating Template Content from 3.41 . . . . . **5**
Introduction 5
Updating Template Content in SAS Web Application Projects That Use the SAS Web Infrastructure Platform 6
Updating Template Content in SAS Web Application Projects That Use SAS Foundation Services 11
Updating Template Content in SAS Java Application Projects 17

Appendix 1 • Other Updates **23**
Updating User Names and Passwords 23
Migrating the SAS Repository Configuration for SAS Java Application Projects 24
Disable the JavaScript Facet 25
Restoring Back Link Behavior for a Report Viewer Servlet Template Added in AppDev Studio 4.4 25

About This Book

Audience

This book is for users who are migrating AppDev Studio projects from AppDev Studio 3.4 through the first maintenance release of AppDev Studio 4.4 to the second maintenance release for AppDev Studio 4.4.

If you want to migrate a project to the first maintenance release of AppDev Studio 4.4, refer to the second edition of this book.

Because projects created in AppDev Studio use the JAR files installed with SAS software, an AppDev Studio project must match the SAS installation on which it will be run. This means that projects from AppDev Studio 3.4 through the first maintenance release for AppDev Studio 4.4 must be upgraded to work with the second maintenance release for AppDev Studio 4.4 and SAS 9.4. It also means that if you upgrade an installation of the original AppDev Studio 4.4 or the first maintenance release of 4.4, all the projects in those installations that you want to work on must also be upgraded.

Existing SAS BI Server profiles are expected to refer to SAS BI installations that have had been upgraded to the second maintenance release for SAS 9.4.

If you need to work on a project that remains at version 4.4 or the first maintenance for 4.4, keep an installation of AppDev Studio at the same version as the project.

Additional information about SAS AppDev Studio is available at **support.sas.com/rnd/appdev/**.

Requirements

Second Maintenance for SAS 9.4

The second maintenance release for AppDev Studio 4.4 works with the second maintenance release for SAS 9.4.

Projects imported from SAS AppDev Studio 3.4 through the first maintenance release for 4.4 must be upgraded to use the second maintenance for SAS 9.4 JAR files before development of those projects can continue.

Eclipse 4.2–4.3.x and the Eclipse Web Tools Platform 3.4.x

SAS AppDev Studio 4.4 is officially supported on Eclipse 4.2 through 4.3.x and the Eclipse Web Tools Platform 3.4.x. However, the SAS AppDev Studio Eclipse Configuration Tool does not enforce these requirements, and allows you to connect to older and newer versions of Eclipse and the Web Tools Platform.

Tomcat

SAS AppDev Studio 4.4 uses Tomcat 7. Tomcat 7.0.30 is the minimum version.

What's New in SAS AppDev Studio 4.4

Overview

SAS AppDev Studio has a new feature that enables you to configure the Tomcat Server automatically. There are also new versions of the SAS facets, and changes to the templates.

New Command to Configure the Tomcat Server

SAS AppDev Studio 4.4 has a new command for configuring a Tomcat server to use for testing SAS web applications. To use the new command, right-click on a Tomcat 7 server in the Servers view, and select **Configure for SAS AppDev Studio Use**. Then choose the BI Server Profile you want that Tomcat 7 server to run against. All the configuration changes are made automatically. Removing the configuration changes is also supported.

These server configuration changes do not include adjusting the start-up time-out, which must be increased from the default to allow for the extra time that SAS web applications need to start. Port modifications must also be managed manually if there are conflicts with a local BI Server. Both of these manual changes are addressed in the **Configure a Tomcat Server for Testing** cheat sheet, which is a part of the **New Workspace Setup** cheat sheet.

New Versions of the SAS Facets for SAS Projects

Building on the facet support in the Eclipse Web Tools Platform to control the configuration of SAS web application projects, SAS AppDev Studio 4.4 provides new versions of the SAS Java Components and SAS Web Infrastructure Platform facets.

AppDev Studio Version	Facet Version
AppDev Studio 4.4	9.4.0.0000
AppDev Studio 4.4 for the first maintenance release of SAS 9.4	9.4.1.0000

AppDev Studio Version	Facet Version
AppDev Studio 4.4 for the second maintenance release of SAS 9.4	9.4.2.0000

Updated BI Server Profiles for SAS 9.4

In SAS 9.4, the SAS Web Infrastructure Platform no longer uses Remote Services. Because of this change, the BI Server Profiles changed, and the only settings that remain from SAS AppDev Studio Prefer IPv4 Stack.

You can still create BI Server Profiles using Search, but because the acquisition of BI Server information now involves a JDBC SQL query, the Search operation cannot create a cached copy of the BI Server information like the prior version of SAS AppDev Studio. This means that before using a BI installation that is offline, you must have acquired the BI Server information at least once to create the cached information.

The time-out for the BI Server Profile is applied only to the time-out for the SQL query that retrieves the BI Server information. It is not applied to other time-outs in your system. This means that if the SAS Web Infrastructure Platform Data Server is not running, or is not accessible, then your system's connection time-out, not the BI Server Profile's time-out, determines how long before the acquisition fails.

Template Differences

Migration of AppDev Studio 3.4 Projects

You can now migrate projects from AppDev Studio 3.4 to the second maintenance release for AppDev Studio 4.4. For projects migrated from 3.4, the 9.2.x.0000 facets are included in various Project Properties dialog boxes. These additional facet versions are needed so that AppDev Studio 3.4 projects can be opened in the second maintenance release for AppDev Studio 4.4.

Local Foundation Services Now Used

Because Remote Services are no longer used, templates now use Local Foundation Services where needed. This means that you must define a Java System property to identify a JAAS configuration file for the Authentication Service. This requirement is not new for web application templates based on SAS Foundation Services, although the JAAS configuration file is now named **`jaas.config`** instead of **`login.config`**.

The JAAS configuration file and a defining Java System property are new for web application templates based on the SAS Web Infrastructure and for the Java client for executing Stored Processes template. For these templates, a **`jaas.config`** file is added to the project and the Java System property is included in the **`launchParameters.txt`** file of the project. For web application projects, the templates continue to add or update these files for historical reasons, although the new

`Configure for SAS AppDev Studio Use` command handles the JAAS configuration and makes these files unnecessary. For the Java client for executing Stored Processes template, you must address the JAAS configuration when you run the Java application added by the template.

Create Application Metadata Template Now Uses the SAS Application Metadata Utility

The Create Application Metadata template has been updated to use the SAS Application Metadata Utility to create the application metadata. Because of this, the **`Application.xml`** file that contains the application metadata has a new format.

Because of the SAS Application Metadata Utility, when the application metadata is created an Application object appears in the folder tree for SAS Management Console and in the SAS AppDev Studio SAS Metadata Folders view in the SAS Metadata perspective. The default path of this folder is **`Products/SAS AppDev Studio/<application name>`**. You can modify this path in the **`Application.xml`** file after the template has been added.

Report Viewer Servlet Template Changes

The Report Viewer Servlet templates in both the first and second maintenance releases for AppDev Studio 4.4 include the changes needed to restore the back link behavior.

As of AppDev Studio 4.4, Remote Services have not been used, and the mechanism in the Report Viewer Servlet templates that communicates with both SAS Web Report Viewer and SAS Web Report Studio changed. Although SAS 9.4 incorporated URL parameters to support back link information, AppDev Studio 4.4 provided for opening a selected report, but did not support passing back link information. The first maintenance release for AppDev Studio 4.4 corrects those templates.

If you import an AppDev Studio 4.4 project that contains an uncorrected Report Viewer Servlet template and want to restore the back link behavior, see "Restoring Back Link Behavior for a Report Viewer Servlet Template Added in AppDev Studio 4.4" on page 25.

Accessibility Features of SAS AppDev Studio

Overview

SAS AppDev Studio 4.4 Eclipse Plug-ins has been tested with assistive technology tools. The software includes accessibility and compatibility features that improve the usability of the product for users with disabilities. (Some accessibility issues remain and are noted below.) These features are related to accessibility standards for electronic information technology that were adopted by the U.S. Government under Section 508 of the U.S. Rehabilitation Act of 1973 (2008 draft proposal initiative update). For detailed information about the accessibility of this product, send e-mail to **accessibility@sas.com** or call SAS Technical Support.

Documentation Format

Please contact **accessibility@sas.com** if you need this document in an alternative digital format.

Exceptions to Accessibility Standards

Exceptions to accessibility standards are documented in the following table.

Exceptions to Accessibility Standards

Accessibility Issue	Workaround
In some cases, screen-reading technology is unable to read text, read field labels in the correct order, or read only the text that is currently visible.	No workaround is available.
The color of the text in the Help window of the SAS AppDev Studio Configuration Tool cannot be changed.	No workaround is available.

Chapter 1

Migrating, Importing, and Upgrading SAS Projects

Migrating SAS Projects 1

Importing a Project into SAS AppDev Studio 4.4 1

Upgrading SAS Projects 2
Upgrade a SAS Web Application Project to Use SAS 9.4 2
Upgrade a SAS Java Application or Portlet Project 3

Migrating SAS Projects

The process of migrating SAS projects from SAS AppDev Studio 3.4 and 3.41 to the second maintenance release involves all three of the following steps.

If you are migrating an AppDev Studio project from a version 4.4 workspace or a first maintenance release for 4.4 workspace, then only the first two steps apply.

If you have applied the second maintenance release for 4.4 to an existing installation of AppDev Studio and are upgrading an existing workspace, then only the second step applies.

1. Import the project into the second maintenance release for SAS AppDev Studio 4.4.

   This step is the same for all SAS projects.

2. Upgrade the project to use the second maintenance release for SAS 9.4.

   The upgrade steps vary depending on the type of project.

3. Update any added content to meet the new SAS and Eclipse standards.

   The update procedure varies depending on the type of content that was added to the project. This is covered in Chapter 2, “Updating Template Content from 3.41,” on page 5.

Importing a Project into SAS AppDev Studio 4.4

You can import a project using an archive of the source workspace or access to the source workspace directory. If the source workspace is on a different system, archive the workspace or the individual projects, and then copy them to the system where the second maintenance release for SAS AppDev Studio 4.4 is installed.

To import a SAS project:

1. From Eclipse, select **File** ⇨ **Import**.
2. Expand the **General** folder.
3. Select **Existing Projects into Workspace** and click **Next**.
4. To import from a SAS AppDev Studio source workspace directory, select **Select root directory** and click **Browse**.

   To import from an archive, select **Select archive file**, and click **Browse**.
5. Select one of the following:
   - the project folder under the workspace
   - the workspace directory to import all the enclosed projects
   - the archive file
6. Select **Copy projects into workspace**.
7. If more than one project is available for import, select the check boxes to enable those projects that you want to import and disable those that you do not want to import.
8. Click **Finish**.

Upgrading SAS Projects

The goal of upgrading an imported project is to alter the project content so that it matches the contents of a project of the same type that was created with the second maintenance release for SAS AppDev Studio 4.4. The upgrade process depends on the type of project.

Upgrade a SAS Web Application Project to Use SAS 9.4

To upgrade a SAS web application project:

1. Right-click the project in the Project Explorer and select **Properties**.
2. Select **Project Facets**.
3. Change the version of the SAS Java Components facet to match your development target.

   Use the `9.4.2.0000` facet for projects running against the second maintenance release for SAS 9.4.

   If the SAS Web Infrastructure Platform facet is enabled, also change its version to `9.4.2.0000`.

   Both SAS facets can be enabled if they are the same version. Until you change the version of both facets, an error is displayed.
4. Click **Apply** to upgrade the project facets.

   If you are upgrading from 4.4 or the first maintenance release for 4.4, skip to step 8.
5. Select the **Runtimes** tab.

6. Deselect the **ADS Apache Tomcat v6.0** run time, select the **ADS Apache Tomcat v7.0**, and then click **Apply**.
7. Click **OK**.
8. Select **Project** ⇨ **Clean**.
9. In the Clean dialog box, select **Clean projects selected below** and enable the check box for the project that you just upgraded. Click **OK**.

The project is now upgraded.

Upgrade a SAS Java Application or Portlet Project

If the SAS Repository configuration for the project is the same as the default SAS AppDev Studio configuration, no changes to the project are needed. If manual changes were made to the SAS Repository configuration, see "Migrating the SAS Repository Configuration for SAS Java Application Projects" on page 24.

If the project contains a Portlet template, then the classpath of the project needs to be updated to reference the new Tomcat 7 server run time. If you are upgrading from 4.4 or the first maintenance release for 4.4, this update has already been done.

To update the project classpath:

1. Right-click the project in the Project Explorer and select **Properties**.
2. Select **Java Build Path** and then the **Libraries** tab.
3. In the list of libraries, select the **Server Library**. It is typically named **`Server Library [ADS Apache Tomcat v6.0] (unbound)`**.
4. Click **Edit**.
5. In the dialog box that appears, select **ADS Apache Tomcat v7.0** and click **Finish**.
6. Click **OK**.

Chapter 2

Updating Template Content from 3.41

Introduction **5**

Updating Template Content in SAS Web Application Projects That Use the SAS Web Infrastructure Platform **6**
- Update Content for an Examples Welcome Page Template 6
- Update Content for a SAS Web Infrastructure Platform Application Metadata Creation Template 6
- Update Content for Information Map Servlet (uses SAS WIP) Templates 7
- Update Content for JDBC Servlet (uses SAS WIP) Templates 8
- Update Content for a Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP) Template 9
- Update the Java Files for a SAS Stored Process Servlet (uses SAS WIP) Template 10

Updating Template Content in SAS Web Application Projects That Use SAS Foundation Services **11**
- Update the Context Listener Template Content 11
- Update Content for Information Map Servlet (uses SAS FS) Templates 12
- Update Content for JDBC Servlet (uses SAS FS) Templates 13
- Update Content for a Report Viewer Servlet (uses SAS Web Report Viewer and SAS FS) Template 14
- Update the Java Files for a SAS Stored Process Servlet (uses SAS FS) Template . 16

Updating Template Content in SAS Java Application Projects **17**
- Introduction 17
- Update the Content for a Java Client for Executing a SAS Stored Process Template 17
- Update the Content of a SAS Information Delivery Portal Editable Portlet Template 19
- Update the Content of the SAS Information Delivery Portal Information Map Runtime Portlet Template 19
- Update the Content of the SAS Information Delivery Portal Information Map Fixed Portlet Template 20

Introduction

Updating template content involves changing the parts of a project that were added after the project was created. The changes to the SAS AppDev Studio 4.4 templates are almost exclusively related to switching from using Remote Foundation Services to Local Foundation Services. The following topics address how to make these same changes to a

migrated project that has content that was added with the SAS AppDev Studio templates.

Where SAS AppDev Studio 4.4 and SAS 9.4 are mentioned, the second maintenance release of each is assumed.

Updating Template Content in SAS Web Application Projects That Use the SAS Web Infrastructure Platform

Update Content for an Examples Welcome Page Template

The Examples Welcome Page creates a JSP page that detects the presence of a particular HTTP session attribute to determine whether a Log Off link should be displayed. The HTTP session attribute to test for has changed in SAS 9.4.

To update the HTTP session attribute:

1. Expand the **WebContent** folder.
2. Right-click the **sas_examples.jsp** file or the appropriate JSP file if you chose a different name, and select **Open With** ⇨ **Text Editor**.
3. Change `sas.framework.userid` to `com.sas.services.session.SessionContextInterface`
4. Save and close the file.

Update Content for a SAS Web Infrastructure Platform Application Metadata Creation Template

The following process updates the content from a SAS Web Infrastructure Platform Application Metadata Creation template to work with a new SAS 9.4 Enterprise BI server by re-applying the template to the project and selecting the SAS 9.4 BI Server Profile that you want to use with the project.

To reapply the metadata creation template:

1. Expand the project's **metadata** folder.
2. If manual changes have been made to the Application.xml file in that folder, or if you are unsure if manual modifications have been made, then rename the `Application.xml` file to `ApplicationOld.xml`.
3. From the main menu, select **File** ⇨ **New** ⇨ **Other**.
4. Expand **SAS AppDev Studio**, select **Add Template Content to Project**, and click **Next**.
5. In the **Project** field, select the project to update.
6. In the Templates tree, expand the **SAS Java Web Application** and **SAS Web Infrastructure Platform Support** categories, and then select **SAS Web Infrastructure Application Metadata Creation**.

   Click **Next**.

7. Due to changes for SAS 9.4, the BI Server Profile selection is either blank if no metadata server connection is open, or matches the BI Server Profile of the open connection. Select the BI Server Profile for which you want to initially create the application metadata.
8. Update the **Application Name** and **Description**.
9. Update the **Protocol**, and **Web Server Host** and **port** to match the ADS Apache Tomcat server (or server of your choice).
10. Click **Finish** to apply the template.

When the template wizard finishes reapplying the template, the **`Application.xml`** file is automatically opened in the XML Editor. The file includes a commented-out example of how to define properties in the new format used by the SAS Application Metadata Utility. If properties were added to the **`ApplicationOld.xml`** file, you can copy these properties to the new **`Application.xml`** file. Also, if the **`ApplicationOld.xml`** file contains other edits for settings like the description, then you can copy those changes from the **`ApplicationOld.xml`** file to the Application.xml file.

Since the SAS Web Infrastructure Platform uses Foundation services internally, the switch to Local Foundation Services means that reapplying the template will add the JAAS configuration needed by the web application. When the template is reapplied, a **`jaas.config`** file is added, and the **`launchParameters.txt`** file updated for SAS 9.4.

Run the create metadata script to actually create the server metadata. Because the SAS Application Metadata Utility now creates the application metadata, a Software Component object representing the application metadata will be visible in the metadata folder tree. The default location is **`/Products/SAS AppDev Studio/<`*`name of application`*`>`**. It is specified by the **Folder** attribute of the Application element in the **`Application.xml`** file. The delete metadata script deletes the application metadata, but does not delete any of the folders that SAS Application Metadata Utility creates in the folder tree.

Update Content for Information Map Servlet (uses SAS WIP) Templates

The Information Map TableView Servlet (uses SAS WIP), Information Map OLAPTableView Servlet (uses SAS WIP), and Information Map Default Servlet (uses SAS WIP) templates can be migrated to SAS 9.4 with a simple change to the servlet.

To update the template content:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the Information Map Servlet template was placed. The default folder is **servlets**.
2. Right-click the Information Map servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.
4. In the **Find** field, enter **`REMOTE_SESSION_CONTEXT`** and select **Find**. The file scrolls to **`CommonKeys.REMOTE_SESSION_CONTEXT`**.
5. In the **Replace with** field, enter **`SESSION_CONTEXT`** and select **Replace**.
6. Save and close the servlet source file.

Rebuild the project if it is not building automatically.

The SAS 9.4 version of the ExamplesSessionBindingListener associated with these templates does not contain support for remote SessionContexts and UserContexts, but is otherwise unchanged from SAS AppDev Studio 3.41. Thus, you are not required to make the ExamplesSessionBindingListener in the migrated project match the 9.4 version for the web application to function correctly. This also applies for the other template types in the SAS Web Application Examples category.

Update Content for JDBC Servlet (uses SAS WIP) Templates

The two templates, JDBC TableView Servlet (uses SAS WIP) and JDBC Default Servlet (uses SAS WIP), can be migrated to SAS 9.4 by updating how the JDBC user name and password values are obtained. Obtaining these values from the UserContext of the currently logged in user does not work in SAS 9.4. The SAS 9.4 versions of these templates have been modified to work the same way as the templates based on SAS 9.4 Foundation Services. This means that the JDBC user name and password are hardcoded in the servlet init-parameters.

Necessarily, implementing a way of obtaining user-specific JDBC credentials instead of hardcoded credentials is left to the customer. If the original code to obtain the JDBC user name and password from the UserContext is still in place, then this code can be modified to match the SAS 9.4 template behavior.

To update the template code:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the JDBC Servlet template was placed. The default folder is **servlets**.
2. Right-click the JDBC servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.
4. In the **Find** field, enter **`JDBC_DATABASE_URL`** and select **Find**. The file scrolls to where the JDBC_DATABASE_URL constant is defined.
5. Change the JDBC_DATABASE_URL value to match the SAS EBI or BI installation that you want to run against.
6. In the **Find** field of the Find/Replace dialog box, enter **`Get credentials`** and click **Find**. The file scrolls to the comment **Get credentials from user context**.
7. Replace that comment and the 5 lines that follow it with the following code:

```
ServletConfig sc = getServletConfig();
String username = sc.getInitParameter("username");
String password = sc.getInitParameter("password");
```

8. (Optional) Delete any unused imports, and then save and close the servlet source file.
9. In the **WEB-INF** folder, right-click the **`web.xml`** file, and open it with a Text Editor or XML Editor.
10. In the **`web.xml`** file, find the servlet declaration that corresponds to the servlet modified in step 5.
11. Append as child elements of that servlet declaration the following init-parameter specifications:

```
<init-param>
    <param-name>username</param-name>
    <param-value>your jdbc username</param-value>
</init-param>
<init-param>
```

```
    <param-name>password</param-name>
    <param-value>your jdbc password</param-value>
</init-param>
```

Ensure that the specified **`username`** has *Log on as a batch job* permission. You can encode the password using the com.sas.util.SasPasswordString class.

12. Save and close the **`web.xml`** file.

Update Content for a Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP) Template

Because Remote Services are no longer used by the SAS Web Infrastructure Platform, the com.sas.webapp.contextsharing.WebappContextParams cannot be used to display a report or information map in SAS Web Report Viewer or SAS Web Report Studio. Instead, URL parameters are now used to communicate the same information.

To manually update a Report Viewer Servlet template to current SAS 9.4 behavior:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the Report Viewer Servlet template was placed. The default folder is **servlets**.
2. Right-click the Report Viewer servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.
4. In the **Find** field, enter **`sendRedirect`** and click **Find**. The file scrolls to where the sendRedirect method is called within a try block.
5. Replace the line with the sendRedirect call with the following code (note the altered sendRedirect call at the end):

```
// Build URL with parameters to open the report.
String url = viewer + "&" + CommonKeys.PFS_REQUEST_PATH_URL
   + "=" + URLEncoder.encode(path,"UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKURL_LIST
   + "=" + URLEncoder.encode(request.getRequestURL().toString(), "UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKLABEL_LIST
   + "=" + URLEncoder.encode(backLabel,"UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKDESCRIPTION_LIST
   + "=" + URLEncoder.encode(backDesc, "UTF-8");
response.sendRedirect(url);
```

Be sure to add an import for java.net.URLEncoder if one is not already present.

6. Delete the lines of code just above the try block:

```
SessionContextInterface remoteSessionContext = [...]
params.setParamRequestPathUrl(path);
```

7. In the **Find** field of the Find/Replace dialog box, enter **`REMOTE_USER_CONTEXT`** and click **Find**. The file scrolls to **`CommonKeys.REMOTE_USER_CONTEXT`**.
8. In the **Replace with** field, enter **`USER_CONTEXT`** and click **Replace**.
9. (Optional) Delete any unused imports.
10. Save and close the servlet source file.

Rebuild the project if it is not building automatically.

Update the Java Files for a SAS Stored Process Servlet (uses SAS WIP) Template

Switching the SAS Stored Process Servlet (uses SAS WIP) template from using Remote Foundation Services to Local Foundation Services resulted in several files changing. One of these files is `StoredProcessConnection.java`. One way to update this file is to add a new stored process template to the project. Another way, documented below, is to add a stored process template to a new temporary project, and copy the file from the new project. The remaining files can be updated manually.

To create the temporary project and add the stored process template:

1. Select **File** ⇨ **New** ⇨ **Other**, expand **SAS AppDev Studio**, and select **SAS Web Application Project**. Then click **Next**.
2. Enter a name for this temporary project and click **Next**.
3. Enable **Add Template Content**, and select the **SAS Stored Process Servlet (uses SAS WIP)** template under the **SAS Web Application Examples** category. Click **Next**.
4. Click **Next** and then select a BI Server Profile. Click **Next** again.

   The BI Server Metadata Server for this profile must be running.
5. If not already connected to a BI Server, click the **Change** button and log on.
6. Click the **Change** button in the Stored Process section, and select a stored process.
7. Click **OK**, and then click **Finish**.

To copy the StoredProcessConnection.java file (and StoredProcessFacade.java file, if needed):

1. In the Project Explorer view for the new temporary project, expand **Java Resources**, **src**, and **support.storedprocess**.
2. Expand the same items in the project being migrated.
3. Select the **StoredProcessConnection.java** file in the source project, and then right-click and select **Copy**.

   When migrating from AppDev Studio 3.4 projects, also copy **StoredProcessFacade.java**.
4. In the migrated project, right-click the **support.storedprocess** package, and select **Paste**.
5. When prompted to confirm overwriting, click **Yes**.
6. Delete the temporary project.

To update the remaining files:

1. In the Project Explorer view for the migrated project, expand the **Java Resources** folder, the **src** folder, and then the folder that represents the package where the servlet for the Stored Process Servlet template was placed. The default folder is **servlets**.
2. Right-click the stored process servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.

4. Use Find/Replace to make the following replacements. Optional replacements make the code read more accurately.

Find	Replace with	Occurrences	Required
`setRemoteInformationService`	`setInformationService`	1	Yes
`setRemoteStoredProcessService`	`setStoredProcessService`	1	Yes
`REMOTE_USER_CONTEXT`	`USER_CONTEXT`	1	Yes
`REMOTE_SESSION_CONTEXT`	`SESSION_CONTEXT`	1	Yes
`remoteInformationService`	`informationService`	5	No
`remoteStoredProcessService`	`storedProcessService`	5	No
`the remote`	`the`	2	No

5. Save and close the servlet source file.
6. In the Project Explorer view for the migrated project, expand **WebContent**, **WEB-INF**, and **spring-config** folders.
7. Right-click the **webapp-config.xml** file and select **Open**.
8. For the bean definition in the file that corresponds to the stored process servlet just updated, replace the two child property elements with the following:

```
<property name="informationService" ref="localInformationService"></property>
<property name="storedProcessService" ref="localStoredProcessService"></property>
```

9. Save and close the file.

Repeat the last set of steps for each additional stored process servlet in the project that you want to update. Rebuild the project if it is not building automatically.

Updating Template Content in SAS Web Application Projects That Use SAS Foundation Services

Update the Context Listener Template Content

If your project contains a FoundationServicesContextListener class, you must apply the Context Listener For Local Services template. Doing so updates the FoundationServicesContextListener class to use only Local Foundation Services, and it updates the JAAS configuration for SAS 9.4. The existing login.config name is preserved when the template updates the project. In new projects, the JAAS configuration file is named **`jaas.config`**.

It is not necessary to create a deployment descriptor for AppDev Studio because the SAS 9.4 Web Infrastructure Platform provides a general purpose deployment descriptor called Platform Local Services. Applying the Context Listener For Local Services template

adds or updates the **sas_metadata_source.properties** file so that the Platform Local Services deployment descriptor is used.

Perform the following steps to update the FoundationServicesContextListener class in the project:

1. In the Navigator view, expand the root folder of the project to update.
2. Right-click the **.SASTemplateResources** file and select **Open With** ⇨ **Text Editor**.
3. Find the line that contains the following:

   ```
   <management function="sas.foundation.services.context.listener"
   ```

4. Change the value of the **level** attribute on this line to **local** if it does not have this value already.
5. Save and close the file.
6. From Eclipse, select **File** ⇨ **New** ⇨ **Other**.
7. Expand the **SAS AppDev Studio** folder, select **Add Template Content to Project**, and then click **Next**.
8. In the **Project** field, select the project to update.
9. In the Templates tree, expand **SAS Java Web Application** and then **SAS Foundation Services Support**.
10. Select **Context Listener For Local Services** and click **Next**.
11. Click **Next** to accept the default Template Configuration Parameters.
12. Due to changes for SAS 9.4, the BI Server Profile selection is blank if no metadata server connection is open, or matches the BI Server Profile of the open connection. Select the BI Server Profile for the Enterprise BI installation that you want the project to run against.
13. If the SAS Metadata Server for the BI installation is running, you can click **Test Configuration** to verify that the service deployment can be read from the metadata server.
14. Click **Finish**.
15. (Optional) Expand the **WebContent**, **WEB-INF**, and **conf** folders in the project and delete the **sas_remote_metadata_source_omr.properties** and **sas_metadata_source_omr.properties** files, if present.

Update Content for Information Map Servlet (uses SAS FS) Templates

The Information Map TableView Servlet (uses SAS FS), Information Map OLAPTableView Servlet (uses SAS FS), and Information Map Default Servlet (uses SAS FS) templates can be migrated to SAS 9.4 by changing the servlet to use local services instead of remote services.

To change from local services to remote services:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the Information Map Servlet template was placed. The default folder is **servlets**.
2. Right-click the Information Map servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.

4. In the **Find** field, enter **`// Get remote`** and click **Find**.
5. Delete the following code that starts with the found comment:

```
// Get remote services attributes. The name "remote.sas.services.attributes"
// should match the REMOTE_SERVICES_ATTRIBUTES string in the Foundation Services
// ServletContextListener
ServiceAttributeInterface [] attributes =
   (ServiceAttributeInterface [])getServletContext().getAttribute("remote.sas.services.attributes");
```

6. After deleting the code, two errors will appear in the file where the **`attributes`** variable is used as a parameter. Delete the **`attributes`** parameter and the comma that precedes it where the two errors occur.
7. Use the Find/Replace dialog box to make the following replacements. Optional replacements make the code more readable.

Find	Replace with	Occurrences	Required
`addRemoteUserContext`	`addUserContext`	1	Yes
`addRemoteSessionContext`	`addSessionContext`	1	Yes
`remoteUserService`	`userService`	2	No
`remoteSessionService`	`sessionService`	2	No
`a remote`	`a`	2	No

8. (Optional) Delete any unused imports.
9. Save and close the servlet source file.
10. In the **WEB-INF** folder, right-click the **`web.xml`** file, and open it with a Text Editor or XML Editor.
11. In **`web.xml`**, find the servlet declaration that corresponds to the servlet and update the init-parameters values for **`metadata-userid`** and **`metadata-password`**, if necessary.

    You can encode the password using the **`com.sas.util.SasPasswordString`** class.
12. Save and close the **`web.xml`** file.

Rebuild the project if it is not building automatically.

Update Content for JDBC Servlet (uses SAS FS) Templates

The only changes needed for the JDBC TableView Servlet (uses SAS FS) and JDBC Default Servlet (uses SAS FS) templates are the JDBC URL and the credentials used for the JDBC connection.

To change the URL and credentials:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the JDBC Servlet template was placed. The default folder is **servlets**.

2. Right-click the JDBC servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.
4. In the **Find** field, enter **`JDBC_DATABASE_URL`** and click **Find**. The file scrolls to where this constant is defined.
5. Update the host and port in the value for the JDBC_DATABASE_URL to values appropriate for the EBI installation that you want the servlet to access.
6. Save and close the servlet source file.
7. In the **WEB-INF** folder, right-click the **`web.xml`** file, and open it with a Text Editor or XML Editor.
8. In the web.xml file, find the servlet declaration that corresponds to the servlet and update the init-parameters values for **`username`** and **`password`**, if necessary. You can encode the password using the com.sas.util.SasPasswordString class.
9. Save and close the **`web.xml`** file.

Rebuild the project if it is not building automatically.

Update Content for a Report Viewer Servlet (uses SAS Web Report Viewer and SAS FS) Template

Because Remote Services is no longer used by the SAS Web Infrastructure Platform, the com.sas.webapp.contextsharing.WebappContextParams class cannot be used to display a report or information map in SAS Web Report Viewer or SAS Web Report Studio. Instead, URL parameters are now used to communicate the same information.

Because the web application in this project is not based on the SAS Web Infrastructure Platform, you must perform a separate login for the SAS Web Report Viewer or Studio.

To manually update a Report Viewer Servlet template to SAS 9.4, follow these steps:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the Report Viewer Servlet template was placed. The default folder is **servlets**.
2. Right-click the Report Viewer servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.
4. In the **Find** field, enter **`sendRedirect`** and click **Find**. The file scrolls to where the sendRedirect method is called within a **`try`** block.
5. Replace the single line with the sendRedirect call with the following code:

```
// Build URL with parameters to open the report.
String url = viewer + "?" + CommonKeys.PFS_REQUEST_PATH_URL
    + "=" + URLEncoder.encode(path,"UTF-8")
    + "&" + CommonKeys.PFS_REQUEST_BACKURL_LIST
    + "=" + URLEncoder.encode(request.getRequestURL().toString(), "UTF-8")
    + "&" + CommonKeys.PFS_REQUEST_BACKLABEL_LIST
    + "=" + URLEncoder.encode(BACK_LABEL,"UTF-8")
    + "&" + CommonKeys.PFS_REQUEST_BACKDESCRIPTION_LIST
    + "=" + URLEncoder.encode(BACK_DESC, "UTF-8");
response.sendRedirect(url);
```

   Add imports for java.net.URLEncoder and com.sas.web.keys.CommonKeys if needed.

6. Find the lines of code just above the **`try`** block:

```
WebappContextParams params = [...]
params.setParamRequestPathUrl(path);
```

Replace them with the following:

```
String viewer = sc.getInitParameter("viewer");
```

7. In the **Find** field, enter **`// Get remote`** and click **Find**. The file scrolls to the comment.
8. Delete the code that starts with the following comment:

```
// Get remote services attributes.  The name "remote.sas.services.attributes"
// should match the REMOTE_SERVICES_ATTRIBUTES string in the Foundation Services
// ServletContextListener
ServiceAttributeInterface [] attributes =
   (ServiceAttributeInterface [])getServletContext().getAttribute("remote.sas.services.attributes");
```

9. After deleting the code, two errors will appear in the file where the attributes variable is used as a parameter. Delete the attributes parameter and the comma that precedes it where the two errors occur.
10. Use the Find/Replace dialog box to make the following replacements. Optional replacements make the code more readable.

Find	Replace with	Occurrences	Required
`addRemoteUserContext`	`addUserContext`	1	Yes
`addRemoteSessionContext`	`addSessionContext`	1	Yes
`SAS_REMOTE_SESSION_CONTEXT`	`SAS_SESSION_CONTEXT`	3	No
`remoteSessionContext`	`sessionContext`	3	No
`remoteUserService`	`userService`	2	No
`remoteSessionService`	`sessionService`	2	No
`a remote`	`a`	2	No

11. Delete any unused imports.
12. Save and close the servlet source file.
13. In the **WEB-INF** folder, right-click the **`web.xml`** file, and open it with a Text Editor or XML Editor.
14. In the **`web.xml`** file, find the servlet declaration that corresponds to the servlet and update the init-parameters values for **`metadata-userid`**, **`metadata-password`**, and **`viewer`**, if necessary.

    You can encode the password using the **`com.sas.util.SasPasswordString`** class.
15. Save and close the **`web.xml`** file.

Rebuild the project if it is not building automatically.

Update the Java Files for a SAS Stored Process Servlet (uses SAS FS) Template

Switching the SAS Stored Process Servlet (uses SAS FS) template from using Remote Foundation Services to Local Foundation Services requires only a couple of simple changes to switch from using remote services to local services.

If you are migrating from SAS AppDev Studio 3.4, you must also update the `StoredProcessFacade.java` and `StoredProcessFacade.java` files. One way to update these two files is to add a new stored process template to the project. Another way, documented below, is to add a stored process template to a new temporary project, and copy the files from the new project.

To create the temporary project and add the stored process templates:

1. Select **File** ⇨ **New** ⇨ **Other**, expand **SAS AppDev Studio**, and select **SAS Web Application Project**. Then click **Next**.
2. Enter a name for this temporary project and click **Next**.
3. Enable **Add Template Content**, and select the **SAS Stored Process Servlet (uses SAS FS)** template under the **SAS Web Application Examples** category. Click **Next**.
4. Click **Next** and then select a BI Server Profile. Click **Next** again.

   The BI Server Metadata Server for this profile must be running.
5. If not already connected to a BI Server, click the **Change** button and log on.
6. Click the **Change** button in the Stored Process section, and select a stored process.
7. Click **OK**, and then click **Finish**.

To copy the `StoredProcessConnection.java` and `StoredProcessFacade.java` files into the temporary project:

1. In the Project Explorer view for the new temporary project, expand **Java Resources**, **src**, and **support.storedprocess**.
2. Expand the same items in the project being migrated.
3. Select the **StoredProcessConnection.java** and **StoredProcessFacade.java**files in the source project, and then right-click and select **Copy**.
4. In the migrated project, right-click the **support.storedprocess** package, and select **Paste**.
5. When prompted to confirm overwriting, click **Yes**.
6. Delete the temporary project.

Even if you are not migrating an AppDev Studio 3.4 project, you must still update the template.

To update a Stored Process Servlet template:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the Stored Process Servlet template was placed. The default folder is **servlets**.
2. Right-click the Stored Process servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.
4. In the **Find** field, enter `// Get remote` and click **Find**.

5. Delete the following code:

```
// Get remote services attributes.  The name "remote.sas.services.attributes"
// should match the REMOTE_SERVICES_ATTRIBUTES string in the Foundation Services
// ServletContextListener
final ServiceAttributeInterface [] attributes =
   (ServiceAttributeInterface [])getServletContext().getAttribute(
       FoundationServicesContextListener.REMOTE_SERVICES_ATTRIBUTES);
```

6. After deleting the code, an error will appear in the file where the attributes variable is used as a parameter. Replace the attributes parameter with null.
7. (Optional) Delete any unused imports.
8. Save and close the servlet source file.
9. In the **WEB-INF** folder, right-click the **`web.xml`** file, and open it with a Text Editor or XML Editor.
10. In the **`web.xml`** file, find the servlet declaration that corresponds to the servlet and update the init-parameters values for **`metadata-userid`** and **`metadata-password`**, if necessary.

    You can encode the password using the com.sas.util.SasPasswordString class.
11. Save and close the **`web.xml`** file.

Rebuild the project if it is not building automatically.

Updating Template Content in SAS Java Application Projects

Introduction

The switch from Remote Foundation Services to Local Foundation services affects the following templates for SAS Java projects:

- Java client for executing a SAS Stored Process template
- SAS Information Delivery Portal Information Map Runtime Portlet

Update the Content for a Java Client for Executing a SAS Stored Process Template

Switching the Java Client for Executing a SAS Stored Process template from using Remote Foundation Services to Local Foundation Services resulted in a number of files changing. In addition, a JAAS configuration file needs to be added to the project so that the Java application that executes the stored process can be run successfully. The preferred way to update a Java client in SAS Java Applications projects is to re-apply the stored process template and create a new Java client. You can either create a new Java client in a new package, or refactor the original Java client package and then reuse it for the new Java client.

The classes found in the support.storedprocess package are automatically updated when a new Java client is created, and do not need updating. These classes are used by the Stored Process Java clients found in the project.

For each Stored Process Java client in the project, follow these steps to create a new Java client file and then copy your customized code to it:

1. In the Project Explorer, right-click the existing package (typically **`console.app`**) and select **Refactor** ⇨ **Rename**.

   The package typically contains the Java and properties files for the client (**`StoredProcessDriver.java`** and **`StoredProcessDriver.properties`**).

2. In the **New name** field, append .old to the existing package. Click **OK** and then click **Continue**.

3. In the *old* package, open the **`StoredProcessDriver.java`** class, find the STORED_PROCESS_PATH_URL static String, and note the name of the stored process.

   These instructions assume that you want to run the same stored process as in the original client.

4. From Eclipse, select **File** ⇨ **New** ⇨ **Other**.

5. Expand the **SAS AppDev Studio** folder.

6. Select **Add Template Content to Project**, and then click **Next**.

7. In the **Project** field, ensure that the project containing the Java client to update is selected.

8. In the Templates tree, expand **SAS Stored Process**, select the **Java client for executing a SAS Stored Process** template, and click **Next**.

9. Select the BI Server Profile for the SAS 9.3 BI installation that you want the Java client to run against, and click **Next**.

10. Ensure that you are logged in with the correct Connection Profile.

11. Update the **Package name** and **Class name**.

12. For the Stored Process, click the **Change** button, select the stored process to execute, and then click **OK**.

13. Update any other options and click **Finish**.

14. In the new Java file that is automatically opened, update the stored process prompt values. Missing values are marked as errors.

15. Copy your customizations in the old Java file to the new one.

    To compare files, select both the old and new versions of the Java file in the navigator view, and then right-click on either file and select **Compare With** ⇨ **Each Other**.

16. (Optional) Delete the *old*, renamed package.

Adding this template creates the **`jaas.config`** and **`launchParameters.txt`** files in the root of the project. The **`jaas.config`** file contains the needed JAAS configuration, and the **`launchParameters.txt`** file contains an example of the Java System property that must be defined for the Java application to execute.

If you are working in AppDev Studio 4.4 with maintenance for SAS 9.4, add the JAAS configuration to the run configuration VM arguments.

1. Right-click **`StoredProcessDriver.java`** in the new **`console.app`** package.

2. Select **Run As** ⇨ **Run Configurations**.

3. Select **Arguments**.

4. Add the Java VM argument to the VM arguments section and click **Apply**.

Update the Content of a SAS Information Delivery Portal Editable Portlet Template

If you are migrating from SAS AppDev Studio 3.4, the content for the SAS Information Delivery Portal Editable Portlet template was updated to use the com.sas.portal.Logger class instead of the now-deprecated SAS Foundation Services LoggerInterface. Although the LoggerInterface will work in SAS 9.4, Eclipse will display warnings and you might need to remove the LoggerInterface in the future.

To avoid the warnings and the use of deprecated code, re-create the project using the AppDev Studio 4.4 template. To re-create the Editable Portlet, follow these steps:

1. If you want to reuse the current Editable Portlet project name for the new project, rename the current project by right-clicking the project in the Project Explorer and selecting **Refactor** ⇨ **Rename**.
2. From Eclipse, select **File** ⇨ **New** ⇨ **Other**
3. Expand the **SAS AppDev Studio** folder.
4. Select **SAS Java Project**, and then click **Next**.
5. Enter the name for the project and click **Next**.
6. Enable **Add Template Content**, expand the **SAS Information Delivery Portal Portlet** category, and select **SAS Information Delivery Portal Editable Portlet**. Click **Next**.
7. Update the default values to match the portlet that you are trying to re-create and click **Finish**.
8. Copy your customizations in the old Java file to the new one.

To compare files, select both the old and new versions of the Java file in a navigator view, and then right-click on either file and select **Compare With** ⇨ **Each Other**.

Update the Content of the SAS Information Delivery Portal Information Map Runtime Portlet Template

Remove the SAS Visual Data Explorer

Migrated AppDev Studio 3.4 projects might contain portlets that were created from templates that use the Information Map content. These projects must be updated to remove the use of the SAS Visual Data Explorer. Projects migrated from AppDev Studio 3.41 would already have this fixed. Removing the Visual Data Explorer can be accomplished with a simple code replacement. For each Java Project containing a portlet based on Information Maps, follow these steps:

1. Select the Java Project containing the Information Map-based portlet in the Project Explorer, Package Explorer, or Navigator view.
2. Select **Search** ⇨ **File** in the main menu.
3. In the Search dialog box, enter `VisualDataExplorer` in the **Contains** text field.
4. Enter **.java* in the **File name patterns** field.
5. Select **Enclosing projects** for the **Scope**, and click **OK** to perform the search.

6. The string **VisualDataExplorer** will be found in two locations in **`Display.java`**. Double-click the second search result to take you to that location in the file.
7. In **`Display.java`**, replace the following lines:

```
VisualDataExplorer vde = new VisualDataExplorer(sessionContext, mapName);
DataSelection dataSelection = (DataSelection) vde.getDataModel();

if (!vde.isOLAP()) {
```

with the lines

```
IntelligentQueryMetadataService queryService =
    (IntelligentQueryMetadataService)IntelligentQueryMetadataServiceFactory.newService();
InformationMap informationMap =
    queryService.getInformationMap(sessionContext, new PathUrl(mapName));
DataSelection dataSelection =
    DataSelectionFactory.newSampleDataSelection(informationMap, null);
if (!informationMap.getStructure().isOLAP()) {
```

8. Right-click in the **`Display.java`** file and select **Source** ⇨ **Organize Imports** to remove the import for the VisualDataExplorer and add imports needed by the new code.
9. Save and close the file.

Replace Remote Foundation Services with Local Foundation Services

Switching the SAS Information Delivery Portal Information Map Runtime Portlet template from using Remote Foundation Services to Local Foundation Services requires one change. If you are migrating from SAS AppDev Studio 3.4, you must also update the template content to remove the use of the SAS Visual Data Explorer. This can be accomplished with a simple code replacement.

To update the template:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the package that contains the Initializer.java source file. The default package is infomapruntimeportet.infomap.
2. Right-click the **`Initializer.java`** source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.
4. In the **Find** field, enter **`REMOTE_USER_CONTEXT`**. In the **Replace with** field, enter **`USER_CONTEXT`**. Then click **Replace All** to replace the one occurrence in the file.
5. Save and close the file.

Update the Content of the SAS Information Delivery Portal Information Map Fixed Portlet Template

If migrating from SAS AppDev Studio 3.4, portlets created from the SAS Information Delivery Portal Information Map Fixed Portlet template must be updated to remove the use of the SAS Visual Data Explorer. This can be accomplished with a simple code replacement. For each Java Project containing a portlet that is based on the SAS Information Delivery Portal Information Map Fixed Portlet Template, follow these steps:

1. Select the Java Project containing the Information Map-based portlet in the Project Explorer, Package Explorer, or Navigator view.

2. Select **Search** ⇨ **File** in the main menu.
3. In the Search dialog box, enter **`VisualDataExplorer`** in the **Contains text** field.
4. Enter **`*.java`** in the **File name patterns** field.
5. Select **Enclosing projects** for the **Scope**, and click **OK** to perform the search.
6. The string **VisualDataExplorer** will be found in two locations in **`Display.java`**. Double-click the second search result to take you to that location in the file.
7. In **`Display.java`**, replace the following lines:

```
VisualDataExplorer vde = new VisualDataExplorer(sessionContext, mapName);
DataSelection dataSelection = (DataSelection) vde.getDataModel();

if (!vde.isOLAP()) {
```

with the lines

```
IntelligentQueryMetadataService queryService =
(IntelligentQueryMetadataService)IntelligentQueryMetadataServiceFactory.newService();
InformationMap informationMap =
    queryService.getInformationMap(sessionContext, new PathUrl(mapName));
DataSelection dataSelection =
    DataSelectionFactory.newSampleDataSelection(informationMap, null);

if (!informationMap.getStructure().isOLAP()) {
```

8. Right-click in the **`Display.java`** file and select **Source** ⇨ **Organize Imports** to remove the import for the VisualDataExplorer and add imports needed by the new code.
9. Save and close the file.

Appendix 1

Other Updates

Updating User Names and Passwords 23

Migrating the SAS Repository Configuration for SAS Java Application Projects 24

Disable the JavaScript Facet 25

Restoring Back Link Behavior for a Report Viewer Servlet Template Added in AppDev Studio 4.4 25

Updating User Names and Passwords

Authentication credentials are hardcoded into the content that is generated by the SAS Java application SAS JDBC DataBean Class template and all SAS web application templates that use the SAS Foundation Services. If you used these templates in the migrated project, you need to examine and update the credentials to whatever is appropriate for the SAS 9.4 BI installation that you will access with the web application. The following instructions assume that the generated code for handling authorization is in use and has not been modified.

For SAS web application templates that use the SAS Foundation Services, the credentials can be found in the project's web.xml file as initialization parameters for the servlet declaration for the controller servlet. The initialization parameters are named **`metadata-domain`**, **`metadata-userid`**, and **`metadata-password`**.

For example:

```
<servlet>
  <servlet-name>StoredProcessDriverServlet</servlet-name>
  <servlet-class>servlets.StoredProcessDriverServlet</servlet-class>
  <init-param>
    <param-name>metadata-domain</param-name>
    <param-value>some domain (typically DefaultAuth)</param-value>
  </init-param>
  <init-param>
    <param-name>metadata-userid</param-name>
    <param-value>some userid</param-value>
  </init-param>
  <init-param>
    <param-name>metadata-password</param-name>
    <param-value>some password</param-value>
</init-param>
```

```
</servlet>
```

For the SAS JDBC DataBean Class template, the initialization parameters for the credentials are named **username** and **password**.

For example:

```
<servlet>
  <servlet-name>JDBCTableViewExampleControllerServlet</servlet-name>
  <servlet-class>servlets.JDBCTableViewExampleControllerServlet</servlet-class>
  <init-param>
    <param-name>username</param-name>
    <param-value>some username</param-value>
  </init-param>
  <init-param>
    <param-name>password</param-name>
    <param-value>some password</param-value>
  </init-param>
</servlet>
```

The SAS JDBC DataBean Class template creates a class that requires that you supply a JDBC connection. Example code for creating this connection is provided inside a comment near the beginning of this class. The code shows that the **user** and **password** keys and values are placed in a java.util.Properties object. If you have implemented any connection code, the code needed to actually implement and create the JDBC connection will differ from the example code.

Migrating the SAS Repository Configuration for SAS Java Application Projects

SAS Java Projects are created with a default set of JAR files in their SAS Repository. The dependencies for these JAR files are resolved automatically. Although these default JAR files and their dependencies are sufficient for many projects, some projects require additional JAR files. For example, the SAS AppDev Studio templates add JAR files that are not included by default. Because the SAS Repository of a migrated project might have been customized, there might be issues that should be addressed when migrating from SAS AppDev Studio 3.41 to 4.4.

Although all the JAR files in the SAS Versioned JAR Repository have new dates for SAS 9.4, the base name of the JAR files has remained largely unchanged. If, in the SAS Repository configuration, all the JAR files are set with Latest as their Match Rule, it is likely that no action needs to be taken. For projects that do need updating, it is not practical to provide instructions about what is needed without knowing the specifics of the project.

Use the Report and JAR Relationships tools in the SAS Repository Configuration dialog box to see whether there are problems that need to be addressed, especially if the **Discover dependencies automatically** option is disabled. Internal dependencies between existing JAR files and the inclusion of new JAR files might require the configuration be updated.

For details about configuring the SAS Repository in a project, see Chapter 8, Managing the JAR Files in a Project in the *SAS AppDev Studio 4.4 Eclipse Plug-ins: User's Guide*, which is available at **support.sas.com/rnd/appdev/**.

Disable the JavaScript Facet

The JavaScript facet is automatically added to SAS web application projects by the Eclipse Web Tools Platform when the project is created. A side effect of this JavaScript support is that it forces JavaScript validation even when such validation is disabled in the Validation preferences. The only way to avoid the numerous JavaScript errors and warnings that come from the JavaScript in the SAS Java Components facet is to disable this JavaScript support by turning off the JavaScript facet.

To turn off the JavaScript facet:

1. Right-click the project in the Project Explorer and select **Properties**.
2. Select **Project Facets**.
3. Right-click the **JavaScript** facet and select **Unlock**.
4. Deselect the check box for the **JavaScript** facet and select **Apply**.
5. Click **OK**.
6. Open or switch to the Markers view.
7. Right-click **JavaScript Problems** in the Description column and click **Delete**.

   Disabling the JavaScript facet does not automatically delete the problem markers that the facet created. Future builds will restore the problem markers for other projects that were included in the deletion.

Restoring Back Link Behavior for a Report Viewer Servlet Template Added in AppDev Studio 4.4

Note: The Report Viewer template in the first maintenance release of AppDev Studio 4.4 already includes the following changes.

The Report Viewer Servlet templates for the SAS Web Infrastructure Platform and SAS Foundation Services do not preserve the back link behavior provided by these templates in the prior version of SAS AppDev Studio. It is possible to restore this behavior using appropriate URL parameters. This has been incorporated into the instructions for updating the content for these templates when migrating projects. The following instructions restore the code as it originally existed.

To restore the back link behavior to a new Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP) template:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the Report Viewer Servlet template was placed. The default folder is **servlets**.
2. Right-click the Report Viewer servlet source file and select **Open**.
3. From the main menu, select **Edit** ⇨ **Find/Replace**.
4. In the **Find** field, enter **`INFORMATION_SELECTOR`** and click **Find**. The file scrolls to where the INFORMATION_SELECTOR constant is found.

5. After this line, insert the following code:

```
private static final String DEFAULT_BACK_LABEL = "Back to RFS";
private static final String DEFAULT_BACK_DESC =
   "Return to remote file selector in the ReportViewer Example";
```

6. Locate the comment **`// Configurable values`** in the file. After this comment, insert the following code:

```
private String backLabel;
private String backDesc;
```

7. Locate the comment **`// Setters for use with Spring configuration`** in the file. After this comment, insert the following code:

```
public void setBackLabel(String backLabel) {
    this.backLabel = backLabel;
}

public void setBackDescription(String backDesc) {
    this.backDesc = backDesc;
}
```

8. Locate the comment **`// Ensure required fields are initialized`** in the file. After this comment, insert the following code:

```
if (backLabel == null) {
    backLabel = DEFAULT_BACK_LABEL;
}
if (backDesc == null) {
    backDesc = DEFAULT_BACK_DESC;
}
```

9. In the **Find** field, enter **`Build URL`** and click **Find**. The file scrolls to where a **`try`** block contains a comment with this text.

10. Replace the comment and the statement that follows it with the following code:

```
// Build URL with parameters to open the report.
String url = viewer + "&" + CommonKeys.PFS_REQUEST_PATH_URL + "="
   + URLEncoder.encode(path,"UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKURL_LIST + "="
   + URLEncoder.encode(request.getRequestURL().toString(), "UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKLABEL_LIST + "="
   + URLEncoder.encode(backLabel,"UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKDESCRIPTION_LIST + "="
   + URLEncoder.encode(backDesc, "UTF-8");
```

11. Save and close the servlet source file.

To restore the back link behavior to a new Report Viewer Servlet (uses SAS Web Report Viewer and SAS FS) template example:

1. In the Project Explorer, expand the **Java Resources** folder, the **src** folder, and then the folder or folders that represent the packages where the servlet for the Report Viewer Servlet template was placed. The default folder is **servlets**.

2. Right-click the Report Viewer servlet source file and select **Open**.

3. From the main menu, select **Edit** ⇨ **Find/Replace**.

4. In the **Find** field, enter **`INFORMATION_SELECTOR`** and click **Find**. The file scrolls to where the INFORMATION_SELECTOR constant is found.

5. After this line, insert the following code:

```
private static final String BACK_LABEL = "Back to RFS";
private static final String BACK_DESC =
   "Return to remote file selector in the ReportViewer Example";
```

6. In the **Find** field, enter **`sendRedirect`** and click **Find**. The file scrolls to where a **`try`** block that contains a comment with this text.

7. Replace the comment and the statement that follows it with the following code:

```
// Build URL with parameters to open the report.
String url = viewer + "?" + com.sas.web.keys.CommonKeys.PFS_REQUEST_PATH_URL
   + "=" + java.net.URLEncoder.encode(path,"UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKURL_LIST + "="
   + URLEncoder.encode(request.getRequestURL().toString(), "UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKLABEL_LIST + "="
   + URLEncoder.encode(BACK_LABEL,"UTF-8")
   + "&" + CommonKeys.PFS_REQUEST_BACKDESCRIPTION_LIST + "="
   + URLEncoder.encode(BACK_DESC, "UTF-8");
```

8. Save and close the servlet source file.