Contents

Chapter 1 • Changes from AppDev Studio 3.4 .................................................. 1
  Changes in AppDev Studio 3.41 .................................................. 1
  Accessibility ................................................................. 4

Chapter 2 • Migrating, Importing, and Upgrading ........................................ 5
  Migrating Projects ........................................................... 5
  Importing a Project into AppDev Studio 3.41 .................................. 5
  Upgrading SAS Projects ....................................................... 6

Chapter 3 • Updating Template Content ..................................................... 9
  Introduction ................................................................. 9
  Updating Template Content in SAS Web Application Projects
    That Use the SAS Web Infrastructure Platform ............................. 10
  Updating Template Content in SAS Web Application Projects
    That Use SAS Foundation Services ...................................... 14
  Updating Template Content in SAS Java Application Projects ............ 18

Appendix 1 • Other Updates ................................................................. 23
  Updating User Names and Passwords ........................................ 23
  Migrating the SAS Repository Configuration for SAS Java Application Projects ... 24
  Workspace Javascript Validation Correction ................................. 25
Chapter 1
Changes from AppDev Studio 3.4

Changes in AppDev Studio 3.41

General Differences

Requires SAS 9.3
Projects created in AppDev Studio 3.41 use the SAS 9.3 jars. Projects imported from AppDev Studio 3.4 must be upgraded to use the SAS 9.3 jars before development of those projects can continue.

Requires Eclipse 3.6.x and the Eclipse Web Tools Platform 3.2.x
AppDev Studio 3.41 runs on Eclipse 3.6.x and the Eclipse Web Tools Platform 3.2.x. The minimum version of the Eclipse Web Tools Platform is 3.2.1.

New Compiler Warning Type
The Eclipse 3.6.x Java compiler has been changed to warn you about the use of raw types even if the code is marked with the @SuppressWarnings("unchecked") statement.

To prevent these warnings, Java files imported from AppDev Studio 3.4 need @SuppressWarnings("unchecked") statements changed to the following:

@SuppressWarnings("rawtypes")

If this code causes the unchecked warnings to reappear, suppress both types of warnings by changing the code to the following:

@SuppressWarnings({"unchecked", "rawtypes"})

Templates in AppDev Studio 3.41 already include these changes where needed.
**BI Server Profiles Now Include Local System Settings**

The BI Server Profile Properties now includes three settings that help AppDev Studio acquire the BI Server information.

**Table 1.1 New Local System Settings**

<table>
<thead>
<tr>
<th>Setting</th>
<th>System Property</th>
<th>Default value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefer IPv4 Stack</td>
<td>java.net.preferIPv4Stack</td>
<td>true</td>
<td>Must be set to <code>true</code> on Windows Vista and Windows 7 to work with an EBI configured to use IPv4. Because the default for EBI installations is IPv4, the setting defaults to <code>true</code>.</td>
</tr>
<tr>
<td>JGroups Bind Address</td>
<td>jgroups.bind_addr</td>
<td>[not set]</td>
<td>When the address is not set, JGroups selects a network interface. On a system with several network interfaces, JGroups might pick a network interface that is unable to connect to the BI Server middle tier. In this situation, the IP address of a network interface that can connect to the BI Server middle tier should be specified in the JGroups Bind Address.</td>
</tr>
<tr>
<td>Multicast TTL</td>
<td>multicast_udp_ip_ttl</td>
<td>32</td>
<td>To limit the multicast broadcasts, reduce this value to reach only the BI Server middle tier. If the BI Server middle-tier is on the same system as AppDev Studio, set the value to 0.</td>
</tr>
</tbody>
</table>

These new parameters are output to the `launchParameters.txt` file that is generated by most of the SAS Web Application templates. However, AppDev Studio cannot use that file to acquire BI Server information directly. Instead, it relies on the Eclipse system property settings in `eclipse.ini`. Because a BI Server acquisition can succeed only when these new properties match between the BI Server Profile and the running Eclipse, be sure to add these settings to `eclipse.ini`.

**Acquisition Timeout Increased**

The Acquisition Timeout for BI Server acquisitions within AppDev Studio has been increased from 10 seconds to 20 seconds to avoid failures caused by the slow loading of plug-ins when they were first used in Eclipse. The timeout used by the BI Server middle tier and test servers in Eclipse remains 10 seconds.

**Metadata Server Connections No Longer Expire**

When logging in to a Metadata server using a Connection Profile, the timeout for the associated SAS SessionContext is now disabled.
SAS Project Differences

New 9.3.0.0000 Versions of the SAS Facets
Building on the facet support in the Eclipse Web Tools Platform to control the configuration of SAS Web application projects, AppDev Studio 3.41 provides new versions of the SAS Java Components and SAS Web Infrastructure Platform facets.

New SAS Tooling Library
All SAS projects now include the SAS Tooling library and the SAS Repository library. The Tooling library contains Java classpath entries needed at build time. These entries are not needed at runtime, which is why they are in a separate library. Currently there is only one entry in the SAS Tooling library. Upgrading the facet of an AppDev Studio 3.4 SAS Web application project automatically adds the SAS Tooling library to the project. To upgrade SAS Java projects, the library must be added manually.

Template Differences

Credential Changes for SAS Stored Process Templates
The Java client for executing a SAS Stored Process template now uses trusted (privileged server account) credentials to deploy SAS Foundation Services. Access to the SAS stored process is controlled with user credentials. Projects must be updated manually to account for these changes.

New Log4j Logging Configuration Template
A new Log4j Logging Configuration template is now available to SAS Web application projects. Because this template requires only the SAS Java Components facet, it can be added to Web application projects that are based on the Foundation Services, and also to SAS Web application projects that use the Web Infrastructure Platform. This logging template replaces the AppDev Studio 3.4 Logging Configuration template, which depended on the now-deprecated Foundation Services Logging Service.

Updated Editable Portlet Template to Use Portal Logger
The SAS Information Delivery Editable Portlet template now uses \texttt{com.sas.portal.Logger} in place of the deprecated \texttt{com.sas.services.logging.LoggerInterface}. The other portlet templates already use the \texttt{com.sas.portal.Logger} to perform logging. Imported AppDev Studio 3.4 editable portlets must be manually updated.

Application ID Field Now Defaults to Blank in the Create Application Metadata Template
The Create Application Metadata template now leaves the Application ID field blank. Imported SAS Web application projects must change the value to a blank string. New validation in SAS 9.3 requires that this value be blank for Web applications developed by customers. This field might be removed in a future version of AppDev Studio.
Deprecated Tools

**Table Server Removed**
The Table Server is no longer available. Projects imported from AppDev Studio 3.4 that use the Table Server must be updated to use the SAS IOM or SAS ShareNet JDBC driver to continue working in SAS 9.3.

**SAS Visual Data Explorer Removed**
The SAS Visual Data Explorer is no longer available. SAS Web Application projects can no longer display Information Map Viewer Servlet content.

In SAS 9.3, Information Maps are displayed by SAS Web Report Viewer or SAS Web Report Studio. To support this shift, the Report Viewer Servlet template can now select Information Maps and Reports. Data Explorations and Information Maps that use advanced features, such as filters that use prompting, must be converted to reports to achieve the same or similar functionality. Portlets based on Information Maps must be altered to remove their dependency on the SAS Visual Data Explorer. The AppDev Studio 3.41 templates that are based on Information Map portlets already incorporate the needed changes.

Accessibility

SAS AppDev Studio 3.41 Eclipse Plug-ins includes accessibility and compatibility features that improve the usability of the product for users with disabilities, with exceptions 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, as amended.

The exceptions are as follows:

- 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.

- The color of the text in the help window of the AppDev Studio Configuration Tool cannot be changed.

If you have questions or concerns about the accessibility of SAS products, send e-mail to accessibility@sas.com.
Chapter 2
Migrating, Importing, and Upgrading

Migrating Projects

The process of migrating SAS projects from AppDev Studio 3.4 to 3.41 involves three steps:

1. Import the project into AppDev Studio 3.41.
   This step is the same for all SAS projects.
2. Upgrade the project to use SAS 9.3.
   The upgrade path varies 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. See Chapter 3, “Updating Template Content,” on page 9.

Importing a Project into AppDev Studio 3.41

You can import a project using an archive of the source workspace or access to the 3.4 workspace directory. If the 3.4 workspace is on a different system, archive the workspace or the individual projects, and then copy them to the system where AppDev Studio 3.41 is installed.

To import a SAS project, follow these steps:

1. From Eclipse, select File ➔ Import.
2. Expand the General folder.
3. Select Existing Projects into Workspace and click Next.
4. To import from an AppDev Studio 3.4 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 entire workspace directory to import all the projects
   - the archive file

6. Select **Copy projects into workspace**.

7. If more than one project is available for import, use 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 AppDev Studio 3.41. The upgrade process depends on the type of project.

**Upgrade a SAS Web Application Project for SAS 9.3**

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 `9.3.0.0000`.

   If the SAS Web Infrastructure Platform facet is enabled, also change its version to `9.3.0.0000`.

   Both SAS facets can be enabled if they are the same version. Until you change the version of both facets to `9.3.0.0000`, an error is displayed.

4. Click **Apply** to upgrade the project facets.

5. Select **Targeted Runtimes**.

6. Deselect the **ADS Apache Tomcat v6.0** runtime and click **Apply**.

   This removes the old server runtime that references the SAS 9.2 Mid-tier JDK Java runtime.

7. Add an updated server runtime that references the SAS 9.3 Mid-tier JDK Java runtime by re-selecting the **ADS Apache Tomcat v6.0** runtime and clicking **Apply**.

8. Click **OK**.

9. Select **Project ⇒ Clean**.

10. 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

1. Right-click the project in the Project Explorer and select Properties.
2. Select Java Build Path and then the Libraries tab.
3. Click Add Library.
4. Select the SAS Tooling library and click Next.
5. Click Finish to add the library, and then OK.

If the SAS Repository configuration for the project is the same as the default AppDev Studio configuration, the project upgrade is complete. If manual changes were made to the SAS Repository configuration, see “Migrating the SAS Repository Configuration for SAS Java Application Projects” on page 24.
Chapter 3
Updating Template Content

Introduction

Updating template content deals with the parts of a project that were added after the project was created. Many of the changes to the AppDev Studio 3.41 templates address minor warnings generated by the updated Java compiler available in Eclipse 3.6.x. The sections that follow address how to make more extensive changes to a migrated project that has content that was added with the SAS AppDev Studio templates.

Introduction

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

Update Content for a SAS Web Infrastructure Platform

Application Metadata Creation Template

Remove Content for an Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS WIP) Template

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

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

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

Recreate the Local Foundation Services Deployment Descriptor

Update the Context Listener Template Content

Remove the Content from an Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS FS) Template

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

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

Updating Template Content in SAS Java Application Projects

Introduction

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

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

Update the Content of the SAS Information Delivery Portal Information Map Fixed Portlet Template and SAS Information Delivery Portal Information Map Runtime Portlet Template
Updating Template Content in SAS Web Application Projects That Use the SAS Web Infrastructure Platform

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

The following two processes update the content from a SAS Web Infrastructure Platform Application Metadata Creation template to work with a new SAS 9.3 EBI server by re-applying the template to the project and selecting the SAS 9.3 BI Server Profile that you want to use with the project.

The AppDev Studio Template Wizard stores the multicast address and port of the last SAS BI Server Profile that a project used in the project's .SASTemplateResources file. To remove the relevant lines from that file and force the project to reacquire the multicast properties for use in the launchparameters.txt file, follow these steps:

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. Delete these two lines and all intervening lines:
   
   ```xml
   <managed-parameter name="multicast.launch.system.properties">
   [...]  
   </managed-parameter>
   ```
   
4. Save and close the file.

To reapply the metadata creation template, follow these steps:

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, rename the Application.xml file to Application92.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. When the Template Wizard fails to display the SAS BI Server Profile for the old, invalid multicast address and port, a new profile called Auto-created Profile is generated. This generated SAS BI Server Profile contains the stored multicast address and port and the defaults for all remaining settings.

When AppDev Studio attempts to acquire the BI Server information for an Auto-created Profile, click Cancel, or allow the acquisition to fail. Failure takes approximately 20 seconds.
8. Select a valid BI Server Profile.
9. Update the Application Name and Description.
10. Delete the contents of the Application ID field.
11. Update the Protocol, and Web Server Host and port to match the ADS Apache Tomcat server (or server of your choice).
12. Click Finish to apply the template.
13. If you saved a copy of the original Application.xml in step 2, select both Application.xml and Application92.xml and select Compare With Each Other. Port any other changes to the new Application.xml file.
14. Remove the automatically generated profile.
   a. Select Window Preferences.
   c. From the BI Server Profile list, select the Auto-created Profile, and click Remove.
   d. Confirm the deletion, and then click OK.

Be sure to run the script to actually create the server metadata.

**Remove Content for an Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS WIP) Template**

Because the SAS Visual Data Explorer is no longer available in SAS 9.3, content from the Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS WIP) template must be removed from your project. To prevent these templates from being displayed by the Examples Welcome File page, 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 servlets for the Information Map Viewer Servlet template were placed. The default folder is servlets.
2. In the WEB-INF folder, right-click the sas-examples.xml file and open it with a Text Editor or XML Editor.
3. In the editor, delete the <type> element (and its children) whose name attribute is Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS WIP). Save and close the file.

To find the servlets to delete, continue with these steps:

1. In the WEB-INF folder, right-click the web.xml file, and open it with a Text Editor or XML Editor.
2. In the spring-config folder, right-click the webapp-config.xml file, and open it with a Text Editor or XML Editor.
3. From the main menu, select Search File.
4. In the Search dialog box, enter VisualDataExplorerServlet in the Contains text field.
5. Enter *.java in the File name patterns field.
6. Select Enclosing projects for the Scope, and click OK to perform the search.
All occurrences of the VisualDataExplorerServlet string are displayed. The string will appear twice in any servlet produced by the Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS WIP) template.

For each servlet Java file that appears in the Search view, perform the following steps:

a. In the web.xml file, delete the <servlet> element whose <servlet-name> element contains the servlet base name. Also delete all of its children.
b. Also in the web.xml file, delete the <servlet-mapping> element whose <servlet-name> element contains the servlet base name. Also delete all of its children.
c. In the webapp-config.xml file, delete the <bean> element whose id attribute contains the name of the Java file without the .java extension.
d. In the WebContent folder, right-click the JSP file that corresponds to the servlet Java file and select Delete. The JSP filename follows the pattern <servletBaseName>Viewer.jsp.

e. Right-click the servlet Java file and select Delete.

Remove any other servlets not identified in the search. Save and close both the web.xml file and the webapp-config.xml file.

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

Because the SAS Visual Data Explorer and the Information Map Viewer Servlet template are no longer available, the ability to display a SAS Information Map is now provided by the Report Viewer Servlet (uses SAS Web Report Viewer and SAS WIP) template.

Both SAS Web Report Viewer and SAS Web Report Studio can display reports that are generated from an Information Map. However, these generated reports do not support all the features of the now-deprecated SAS Visual Data Explorer, nor do SAS Web Report Viewer and SAS Web Report Studio always deal with Information Map features in the same way. For example, if you generate a report from an Information Map with a filter that uses a prompt, SAS Web Report Viewer and Web Report Studio handle the prompt differently. To achieve equivalent functionality, a new report must be created based on the Information Map. Similarly, because Data Explorations cannot be displayed by SAS Web Report Viewer or SAS Web Report Viewer Studio, a new report is required.

To manually update all the Report Viewer Servlet templates in your project to display Information Maps without reapplying the template, follow these steps:

1. Select the project to update.
2. From the main menu, select Search ➔ File.
3. In the Contains text field, enter:
   ```java
   filterList.add("Report");
   ```
4. In the File name patterns field, enter * . java.
5. For the Scope select Enclosing projects, and click OK to perform the search.
6. For each occurrence of filterList.add("Report"); in the Search view, perform the following:
a. From the main menu, select Navigate ➔ Next Match. This will open the Java file in an editor and take you to the line where the text was found.

b. Above the filterList.add("Report"); line add a new line:
   
   filterList.add("InformationMap");

c. Save and close the file.

Rebuild the project if it is not building automatically.

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

This process updates the StoredProcessConnection.java and StoredProcessFacade.java files for the SAS Stored Process Servlet template that uses the Web Infrastructure Platform. If you are using Foundation Services, see “Update the Java Files for a SAS Stored Process Servlet (uses SAS FS) Template” on page 17.

There are two ways to update these Java files, which are produced by the SAS Stored Process Servlet (uses SAS WIP) template. The first way is to add a new stored process template to the project. The addition of the template updates the Java files automatically.

The second way, documented here, is to copy the Java files from a new, temporary project that uses the AppDev Studio 3.41 SAS Web Infrastructure Platform and then insert them into the migrated project. Follow these steps 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 Remote Services 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. Click OK, and then Finish.

To copy the StoredProcessConnection.java and StoredProcessFacade.java files, follow these steps:

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 both 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 on the support.storedprocess package, and select Paste.

5. When prompted to confirm overwriting, click Yes to All.
If either of the files in the migrated project are open in editors, switch to those editors and select File ⇒ Save.

The Java files are now updated.

6. Delete the temporary project.

Stored process servlets in the project being migrated will contain new generics warnings. These warnings can be suppressed by changing each of the @SuppressWarnings found in the Java file to @SuppressWarnings({"unchecked", "rawtypes"}). You can also review how the warnings were addressed in the stored process servlet in a new project and make the same change in the migrated project.

---

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

**Recreate the Local Foundation Services Deployment Descriptor**

If your project uses Local Foundation Services, you must recreate the ADS Local Services deployment descriptor in your SAS 9.3 metadata. Follow these steps:

1. Open SAS Management Console and use sasadm@saspw to connect to the SAS Metadata Server for the SAS BI installation that you are developing against.

2. Select the Plug-ins tab.

3. In the Environment Management folder, expand Foundation Services Manager.

4. Right-click SASWIPServices9.3 Local Services, and then select Duplicate Service Deployment.

5. Enter a name for the service deployment that is appropriate for your Web application (for example, SASAppDevStudio3.4 Local Services).

6. Click OK to create the service deployment and then exit the SAS Management Console.

**Update the Context Listener Template Content**

Regardless of the type of SAS Foundation Service used, update the FoundationServicesContextListener class in the project by re-applying the appropriate SAS Foundation Services Support template.

Before you start the update, you need to know whether the listener deploys Foundation Services that are local, remote, or both. You can determine the type of deployment by examining the web.xml file in the project. If a `<context-param>` named local.sas.foundation.services exists, then Local Foundation Services are being deployed. If a `<context-param>` named remote.sas.foundation.services exists, the Remote Foundation Services are being deployed.

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

1. From Eclipse, select File ⇒ New ⇒ Other.
2. Expand the SAS AppDev Studio folder, select Add Template Content to Project, and then click Next.

3. In the Project field, select the project to update.

4. In the Templates tree, expand SAS Java Web Application and then SAS Foundation Services Support.

5. If the Web application that you are migrating deploys only local SAS Foundation Services, select Context Listener For Local Services.
   
   If the Web application deploys only SAS Remote Services, select Context Listener for Remote Services.
   
   If the Web application deploys both local and remote services, select Context Listener for Remote and Local Services.

6. Click Next.

7. Click Next to accept the default Template Configuration Parameters.

8. When the Template Wizard fails to display the SAS BI Server Profile for the old, invalid multicast address and port, a new profile called Auto-created Profile is generated. This generated SAS BI Server Profile contains the stored multicast address and port and the defaults for all remaining settings.

   When AppDev Studio attempts to acquire the BI Server information for an Auto-created Profile, click Cancel, or allow the acquisition to fail. Failure takes approximately 20 seconds.

9. Select a valid BI Server Profile.

10. If an ADS Local Services deployment descriptor was created but was not named SASAppDevStudio3.4 Local Services, click Advanced. In the Local Services Deployment section, select Other, click Browse, select the service deployment descriptor that you created earlier, and then click OK.

11. If the SAS Metadata Server for the BI installation is running, you can click Test Configuration to verify that the service deployment or deployments can be read from the metadata server.

12. Click Finish.

In addition to updating the FoundationServicesContextListener.java file, you must also update the launchParameters.txt file to match the launch parameters of your test Server. For example, the ADS Apache Tomcat v6.0 at localhost server created in the New Workspace Setup cheatsheet.

---

Remove the Content from an Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS FS) Template

Because the SAS Visual Data Explorer is no longer available in SAS 9.3, content from the Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS WIP) template must be removed from your project. Follow these steps:

1. In the Project Explorer, expand the Java Resources folder, the src folder, and then the folder(s) that represent the package(s) where the servlets for the Information Map Viewer Servlet template were placed. The default folder is servlets.

2. In the WEB-INF folder, right-click the web.xml file, and open it with a Text Editor or XML Editor.

3. Select Search ➔ File in the main menu.
4. In the Search dialog box, enter `VisualDataExplorerServlet` in the **Contains text** field.

5. Enter `*.java` in the **File name patterns** field.

6. Select **Enclosing projects** for the Scope and click OK to perform the search.

   All occurrences of the `VisualDataExplorerServlet` string are displayed. The string will appear twice in any servlet produced by the **Information Map Viewer Servlet (uses SAS Visual Data Explorer and SAS FS)** template.

   For each servlet Java file that appears in the Search view, perform the following steps:

   a. In the `web.xml` file, delete the `<servlet>` element whose `<servlet-name>` is the `servletBaseName`. Also delete all of its children.

   b. Also in the `web.xml` file, delete the `<servlet-mapping>` element whose `<servlet-name>` is the `servletBaseName`. Also delete all of its children.

   c. In the `WebContent` folder, delete both the servlet JSP file (`<servletBaseName>Viewer.jsp`), and the servlet Java file (`<servletBaseName>ControllerServlet.java`).

   Remove any other servlets not identified in the search. Save and close the `web.xml` file.

---

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

Because the SAS Visual Data Explorer and the Information Map Viewer Servlet template are no longer available, the ability to display a selected SAS Information Map is now provided by the **Report Viewer Servlet (uses SAS Web Report Viewer and SAS FS)** template.

Both SAS Web Report Viewer and SAS Web Report Studio can display reports that are generated from an Information Map. However, these generated reports do not support all the features of the now-deprecated SAS Visual Data Explorer, nor do SAS Web Report Viewer and SAS Web Report Studio always deal with Information Map features in the same way. For example, if you generate a report from an Information Map with a filter that uses a prompt, SAS Web Report Viewer and Web Report Studio handle the prompt differently. To achieve equivalent functionality, a new report must be created based on the Information Map. Similarly, because Data Explorations cannot be displayed by SAS Web Report Viewer or SAS Web Report Viewer Studio, a new report is required.

To manually update all the Report Viewer Servlet templates in your project to display Information Maps without reapplying a template, follow these steps:

1. In the Project Explorer, expand the project to update, and then the **WebContent** and **WEB-INF** folders.

2. In the **WEB-INF** folder, right-click the `web.xml` file, and open it with a Text Editor or XML Editor.

3. Select **Search ➔ File** in the main menu.

4. In the **Contains text** field, enter:

   ```
   filterList.add("Report");
   ```

5. In the **File name patterns** field, enter `*.java`.

---
6. For the **Scope**, select **Enclosing projects** and then click **OK** to perform the search.

   For each occurrence of `filterList.add("Report");` in the Search view, perform the following steps:
   
a. From the main menu, select **Navigate ➔ Next Match**. This will open the Java file in an editor and take you to the line where the text was found.
   
b. Above the `filterList.add("Report");` line add a new line:
      ```java
      filterList.add("InformationMap");
      ```
   
c. Save and close the file.
   
d. In the `web.xml` file, find the `<servlet>` declaration whose name matches the Java file that you just edited.
   
e. Update each `<init-param>` value to what is appropriate for the target SAS EBI installation. The `viewer` parameter should be in the form of `http://<ebi_host>:<ebi_port>/<context>/logonAndRender.do`

      The `<context>` is either SASWebReportStudio or SASWebReportViewer, depending on which is in the SAS EBI installation.

   Rebuild the project if it is not building automatically.

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

This process updates the `StoredProcessConnection.java` and `StoredProcessFacade.java` files for the SAS Stored Process Servlet template that uses Foundation Services. If you are using the Web Infrastructure Platform, see “Update the Java Files for a SAS Stored Process Servlet (uses SAS WIP) Template” on page 13.

There are two ways to update these Java files, which are produced by the SAS Stored Process Servlet (uses SAS FS) template. The first way is to add a new stored process template to the project. The addition of the template updates the Java files automatically.

The second way, documented here, is to copy the Java files from a new, temporary project that uses SAS Foundation Services and then insert them into the migrated project. Follow these steps to create the temporary project and add the stored process template:

1. Select **File ➔ New ➔ Other**, expand **Web**, and select **Dynamic Web Project**. Then click **Next**.

2. Enter a name for this temporary project.

3. Change the Dynamic web module version to **2.4**, and the Configuration to **SAS 9.3 Java Components Configuration**. Click **Finish**.

4. Select **File ➔ New ➔ Other**, expand **SAS AppDev Studio**, and select **Add Template Content to Project**. Click **Next**.

5. For **Project**, ensure that the Dynamic Web Project you just created is selected.

6. For the **Template**, expand **SAS Java Web Application** and **SAS Web Application Examples**, and then select **SAS Stored Process Servlet (uses SAS FS)**. Click **Next**.

7. Click **Next**. Select a BI Server Profile. Click **Next** again.

   The BI Server Remote Services for this profile must be running.

8. If not already connected to a BI Server, click the **Change** button and log on.
9. Click the Change button in the Stored Process section, and select a stored process. Click OK, and then Finish.

To copy the StoredProcessConnection.java and StoredProcessFacade.java files, follow these steps:

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 both 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 on the support.storedprocess package, and select Paste.
5. When prompted to confirm overwriting, click Yes to All.

   If either of the files in the migrated project are open in editors, switch to those editors and select File ➤ Save.

   The Java files are now updated.
6. Delete the temporary project.

   Stored process servlets in the project being migrated will contain new generics warnings. These warnings can be suppressed by changing each of the @SuppressWarnings found in the Java file to @SuppressWarnings({"unchecked", "rawtypes"}).

   You can also review how the warnings were addressed in the stored process servlet in a new project and make the same change in the migrated project.

---

### Updating Template Content in SAS Java Application Projects

#### Introduction

Most of the template content found in migrated SAS Java projects will have new warnings because of the updated compiler in Eclipse 3.6. These warnings can be fixed manually for the following templates:

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

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

With the release of the version 4.3 SAS middle tier applications, the Java client for executing a SAS Stored Process template now requires separate credentials for deploying SAS Foundation Services and running the stored process. 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 **Continue**.

3. In the **old** package, open `StoredProcessDriver.java`, 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 will be 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, then right-click on either file and select **Compare With ➔ Each Other**.

16. (Optional) Delete the `old`, renamed package.
Update the Content of a SAS Information Delivery Portal Editable Portlet Template

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 9.3, 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, recreate the project using the AppDev Studio 3.41 template. To recreate 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`.
7. Update the default values to match the portlet that you are trying to recreate 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, then right-click on either file and select `Compare With` → `Each Other`.

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

Portlets created from templates that use the Information Map content 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 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.
Appendix 1
Other Updates

Updating User Names and Passwords .............................................. 23
Migrating the SAS Repository Configuration for SAS Java Application Projects .............................................. 24
Workspace Javascript Validation Correction ................................. 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 9.3 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:

```xml
<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:
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 jars in their SAS Repository. The dependencies for these jars are resolved automatically. Although these default jars and their dependencies are sufficient for many projects, some projects require additional jars. For example, the AppDev Studio templates add jars that are not included by default. Because the SAS Repository of a migrated project might have been customized, there are three issues that should be addressed when migrating from AppDev Studio 3.4 to 3.41.

First, if the SAS Repository includes the `sas.oda.tkts` jar, remove it. The jar will be present if the SAS JDBC DataBean Class template was added to the project in AppDev Studio 3.4. This class supported the SAS Table Server, which is not supported in SAS 9.3.

Second, if the SAS Repository includes `spring.jar`, remove it. For SAS 9.3, this jar has been reorganized into a number of separate jars distinguished by technology. The new jars start with the name `org.springframework`, followed by an abbreviated name to identify the technology (for example, `org.springframework.aop`).

Finally, if the SAS Repository configuration includes jars with Match Rules other than `Latest`, update the rule for those jars to the version that is available in the SAS 9.3 Version Jar Repository. All SAS jars and many third-party jars will have version numbers that differ from the SAS 9.2 version.

For details about configuring the SAS Repository in a project, see Chapter 8, Managing the Jars in a Project, in the SAS AppDev Studio 3.4 Eclipse Plug-ins User’s Guide in the SAS AppDev Studio Developer’s Site.
Workspace Javascript Validation Correction

After the New Workspace Setup cheat sheet is run, some folders are excluded from client-side Javascript validation. However, this exclusion is not immediately honored. To correct this, follow these steps for each new workspace:

1. Select **Window ⇒ Preferences** in the main menu, and then select **Validation**.

2. For **Client-side JavaScript**, enable the check boxes for both the **Manual** and **Build** columns, and click **Apply**. Validation is applied only to JavaScript outside of the excluded folders **WebContent/scripts** and **WebContent/templates**.

3. If you want to completely disable JavaScript validation, deselect the check box for **Build** and **Manual**, and click **Apply**.

4. Click **OK** to exit the Preferences dialog box.

If you have existing projects that display Javascript validation errors from SAS Web Application projects, right-click the project and select **Validate**.