Workday Studio

Product Summary

October 29, 2026

workday.

Contents

Setup Considerations: Workday Studio....10

Studio Fundamentals....12
- Download and Install Workday Studio for Windows....12
- Download and Install Workday Studio for macOS....14
- Concept: Workday Studio....15
- Concept: Integration Development Lifecycle....17
- Concept: MediationContext Object....18
- Concept: Integration Systems and Events....19
- Reference: Interface Fundamentals....19

Integration Design....21
- Assemblies....21
  - Create an Assembly Project....21
  - Add Steps to Components....21
  - Add Child Elements....22
  - Connect Assembly Elements....23
  - Configure Assembly Element Properties....23
  - Manage Collections and Projects....24
  - Declare Namespace Prefixes....25
  - Create Path Variables....26
  - Concept: Assembly Palette....26
  - Concept: Components and Steps....28
  - Concept: Assembly Annotations....28
  - Concept: Linked Resource Files for Assembly Projects....29
  - Concept: Quick Outline View....29
  - Reference: Assembly Properties....30
  - Reference: Swimlane Properties....30
  - Reference: Connection Line Properties....31
- Assembly Transports....31
  - Use Enumerations as Launch Parameters in Studio....31
  - Concept: Assembly Transports....32
  - Concept: Enumerations....34
  - Reference: Workday-In Transport Properties....35
  - Reference: Local-In Transport Properties....36
  - Reference: Workday-Out-Rest Transport Properties....38
  - Reference: Workday-Out-Soap Transport Properties....39
  - Reference: Local-Out Transport Properties....40
  - Reference: HTTP-Out Transport Properties....42
  - Reference: FTP-Out Transport Properties....45
  - Reference: FTPS-Out Transport Properties....50
  - Reference: SFTP-Out Transport Properties....54
  - Reference: XMPP-Out Transport Properties....59
  - Reference: Custom-Out Transport Properties....60
  - Reference: Email-Out Transport Properties....62
- HTTP-Out Transport Child Elements....69
  - Concept: HTTP-Out Transport Child Elements....69

©2026 Workday, Inc. All rights reserved

Reference: HTTP-Basic-Auth Properties....69
Reference: HTTP-Custom-Auth Properties....70
Reference: HTTPS Properties....71
FTPS-Out Transport Child Elements....72
Concept: FTPS-Out Transport Child Elements....72
Reference: Proxy-Properties Properties....72
Reference: Firewall-Properties Properties....73
Reference: Client-Key Properties....73
Reference: Server-Key Properties....74
SFTP-Out Child Elements....74
Concept: SFTP-Out Child Elements....74
Reference: Proxy-Properties Properties....74
Reference: Private-Key Properties....75
Components....75
Concept: Components....75
Reference: Async-Mediation Component Properties....76
Reference: Sync-Mediation Component Properties....77
Reference: Custom-Mediation Component Properties....78
Reference: Route Component Properties....78
Reference: Splitter Component Properties....78
Reference: Aggregator Component Properties....79
Route Component Child Elements....79
Concept: Route Component Child Elements....79
Reference: Custom-Strategy Properties....80
Reference: Regex-Strategy Properties....81
Reference: MVEL-Strategy Properties....81
Reference: Round-Robin-Strategy Properties....81
Reference: XPath-Strategy Properties....82
Reference: Failover-Strategy Properties....82
Reference: All-Strategy Properties....83
Reference: Doc-Iterator Properties....83
Reference: Loop Strategy Properties....83
Reference: Choose-Route Properties....84
Reference: Sub-Route Properties....84
Splitter Component Child Elements....84
Concept: Splitter Component Child Elements....84
Reference: Custom-Splitter Properties....85
Reference: Standard-Splitter Properties....86
Reference: XPath-Splitter Properties....86
Reference: XML-Stream-Splitter Properties....86
Reference: JSON-Splitter Properties....87
Reference: Unzip-Splitter Properties....87
Reference: Mtable-Splitter Properties....87
Standard-Splitter Child Elements....88
Concept: Standard-Splitter Child Elements....88
Reference: Header-Ends-With Properties....88
Reference: Header-Fixed-Lines Properties....88
Reference: Content-Fixed-Lines Properties....89
Reference: Content-Starts-Ends Properties....89
Reference: Content-Starts-With Properties....89
Reference: Footer-Fixed-Lines Properties....89
Reference: Footer-Starts-With Properties....90
Aggregator Component Child Elements....90
Concept: Aggregator Component Child Elements....90
Reference: Custom-Batch-Strategy Properties....90
Reference: Size-Batch-Strategy Properties....91

©2026 Workday, Inc. All rights reserved

Reference: Time-Batch-Strategy Properties....91
Reference: Custom-Collater Properties....91
Reference: Message-Content-Collater Properties....92
Reference: XML-Message-Content-Collater Properties....93
Reference: JSON-Aggregator Properties....94
Reference: Zip-File-Collater Properties....94
Reference: Mtable-Collater Properties....95
Common Components....96
Concept: Common Components....96
Reference: FTP Subassembly Properties....99
Reference: GetEventConfigurations Subassembly Properties....102
Reference: GetEventDocuments Subassembly Properties....104
Reference: GetIntegrationEvent Subassembly Properties....105
Reference: GetIntegrationSystems Subassembly Properties....107
Reference: PagedGet Subassembly Properties....109
Reference: PagedGetLocalPaging Subassembly Properties....112
Reference: PdfPrintStep Subassembly Properties....114
Reference: PutIntegrationEvent Subassembly Properties....117
Reference: PutIntegrationMessage Subassembly Properties....119
Reference: SalesforceConnector Subassembly Properties....122
The PrismAnalytics Subassembly....124
Error Handler Components....136
Concept: Error Handler Components....136
Reference: Custom-Error-Handler Component Properties....137
Reference: Log-Error Component Properties....139
Reference: Send-Error Component Properties....140
Transform Steps....141
Convert Text to XML with a Text Schema File....141
Concept: Transform Steps....144
Reference: XSLT Step Properties....146
Reference: XSLT+ Step Properties....149
Reference: FOP Step Properties....151
Reference: Text-Excel Step Properties....155
Reference: TextSchema Step Properties....157
Reference: Wrap-Soap Step Properties....159
Reference: XML-to-Java Step Properties....161
Reference: JSON Transformer Properties....162
Reference: XML-to-JSON Step Properties....164
Reference: JSON-to-XML Step Properties....166
Reference: Java-to-XML Step Properties....168
Reference: XML-to-CSV Step Properties....169
Reference: CSV-to-XML Step Properties....172
Reference: Character-Conversion Step Properties....175
Reference: Mtable-Builder Step Properties....177
Reference: Mtable-Writer Step Properties....178
Reference: ETV Step Properties....179
Reference: XTT Step Properties....180
Mtable-Builder Child Elements....182
Concept: Mtable-Builder Child Elements....182
Reference: CSV-Mtable-Reader Properties....182
Reference: JSON-Mtable-Reader Properties....183
Mtable-Writer Child Elements....183
Concept: Mtable-Writer Child Elements....183
Reference: CSV-Mtable-Writer Properties....184
Reference: JSON-Mtable-Writer Properties....184
Logging and Eventing Steps....185

©2026 Workday, Inc. All rights reserved

Concept: Logging and Eventing Steps....185
Reference: Log Step Properties....185
Reference: Cloud-Log Step Properties....187
Message Manipulation Steps....189
Concept: Message Manipulation Steps....189
Reference: Copy Step Properties....189
Reference: JavaScript Step Properties....191
Reference: Set-Headers Step Properties....193
Reference: Write Step Properties....195
Validation Steps....196
Concept: Validation Steps....196
Reference: Eval Step Properties....197
Reference: Validate Step Properties....198
Reference: Validate-Exp Step Properties....200
Reference: Validate-Xpath Step Properties....201
Encoding Steps....203
Concept: Encoding Steps....203
Concept: PGP Filenames....204
Reference: PGP-Encrypt Step Properties....205
Reference: PGP-Decrypt Step Properties....208
Reference: Base64-Decode Step Properties....210
Reference: Base64-Encode Step Properties....212
Reference: Zip Step Properties....213
Reference: Unzip Step Properties....216
Reference: Compress Step Properties....218
Reference: Decompress Step Properties....219
Other Steps....221
Concept: Other Steps....221
Reference: Custom Step Properties....222
Reference: Set-Dynamic-Endpoint Step Properties....223
Reference: Store Step Properties....224
Reference: Retrieve Step Properties....227
Reference: Xmldiff Step Properties....229
Reference: Enqueue-Message Step Properties....233
Assembly Modules....234
Create Assembly Modules....234
Add Assembly Modules to a Project....236
Concept: Assembly Modules....237
Subassemblies....237
Create Subassemblies....237
Concept: Subassemblies....238
Custom Assembly Components....238
Create Custom Java Classes and Spring Beans....238
Concept: Custom Assembly Elements....240
Concept: Custom Element Packaging....241
Spring Bean Configuration....242
Concept: Spring Bean Configuration....242
Reference: Global Bean Properties....243
Reference: Individual Bean Properties....244
Reference: Alias Properties....245
Reference: Property Properties....245
Reference: Constructor-Arg Properties....246
Reference: Ref Properties....246
Reference: List Properties....246
Reference: Map Properties....246
Message Builder....247

©2026 Workday, Inc. All rights reserved

Create XML Literals Based on Schema or WSDLs ... 247
Create XPath Expressions Based on Schema or WSDLs ... 248
Concept: Workday Studio Message Builder ... 249
Reference: Message Element Properties ... 250
Transport Bindings ... 254
Concept: Transport Bindings ... 254
Reference: WSDL-HTTP-Binding Properties ... 254
Reference: WSDL-SOAP-Binding Properties ... 255
WSDL-SOAP-Binding Child Elements ... 255
Concept: WSDL-SOAP-Binding Child Elements ... 255
Reference: Custom-Policy Properties ... 256
Reference: WS-Rm Properties ... 256
Reference: WS-Security Properties ... 256
Reference: WS-Ut Properties ... 256
Schema Explorer ... 257
Display WSDL and XSD Files ... 257
Display Workday Web Service WSDLs ... 258
Display Custom RaaS Report Schemas ... 258
Concept: Schema Explorer ... 259
MVEL ... 259
Concept: MVEL Fundamentals ... 259
Concept: MVEL and Studio ... 262
Reference: Studio Helper Objects ... 263
Reference: Document Accessor Helper Object ... 267
Reference: Launch Parameters Helper Object ... 271
Reference: Util Helper Object ... 274
Reference: Sample Studio MVEL Expressions ... 276
Reference: Studio MVEL Auto-Conversion Syntax Samples ... 278
Reference: MVEL Methods ... 281
Studio Integration Services ... 282
Create Attribute Map Services ... 282
Create Delivery Services ... 284
Create Listener Services ... 284
Create Retrieval Services ... 285
Create Sequence Generator Services ... 286
Create Transaction Log Services ... 287
Create Report Services ... 287
Create Custom Object Services ... 288
Create Data Initialization Services ... 289
Create Service References ... 290
Add References to Custom Reports on Workday-Out-Rest Transports ... 290
Add References to Custom Objects on Workday-Out-Rest Transports ... 291
CLAR Files ... 292
Generate Manifest Files ... 292
Import CLAR Files ... 293
Export CLAR Files ... 293
Concept: CLAR Files ... 294
JAXB Projects ... 294
Create JAXB Projects ... 294
Add JAXB Projects as Dependencies in Assembly Projects ... 296
Concept: JAXB Support in Studio ... 296
Studio Ivy Plugin ... 297
Configure the Ivy Plugin ... 297
Add Workday Ivy Natures on Workday Studio Projects ... 297
Resolve Dependencies for Projects with Ivy Definitions ... 298
Add Dependent JARs to Projects with Ivy Natures ... 298

©2026 Workday, Inc. All rights reserved

Remove Workday Ivy Natures from Studio Projects....299

Integration Deployment....299
Add a Connection....299
Add a Group of Connections....300
Edit a Connection....301
Edit a Group of Connections....301
Remove Connections....302
Import Connections....302
Export Connections....303
Connect to Workday....303
Deploy Integrations to Workday....304
View Deployed Integrations in Workday....305
Configure Studio to Use a HTTPS Proxy Server....305
Concept: Studio Connections....306
Concept: Cloud Explorer....306
Reference: Connection Errors....306
Multifactor Authentication in Workday Studio....307
Enable Multifactor Authentication in Workday Studio....307

Integration Configuration....309
Assign an Integration System User Account to an Integration....309
Concept: Integration System Security....309

Integration Launching....310
Launch Integrations from Studio....310
Copy Launch URLs....310
Create Launch Configurations....311
Edit Launch Configurations....312

Integration Monitoring and Troubleshooting....312
Process Monitor....312
Concept: Integration Event IDs....312
Reference: Process Monitor Toolbar Buttons....313
Consolidated Report Viewer....313
Load Consolidated Reports for Integrations....313
Save Consolidated Reports to Your Workspace....314
View Workday Integration Event Details Using Consolidated Reports....315
View Request Messages for Integration Events Using Consolidated Reports....315
Monitor Integrations with the Consolidated Report Viewer Events Log....316
Monitor Integrations with the Consolidated Report Viewer Profile Log....316
Replay Assembly Executions for Consolidated Reports....317
Link Consolidated Report Audit Events to Assembly Elements....318
Display Consolidated Report Annotations on Assembly Diagrams....318
Concept: Consolidated Report Viewer....319
Reference: Consolidated Report Error Messages....319
AUnit Assembly Testing....323
Create an AUnit Test Skeleton for an Assembly Project....323
Concept: AUnit Tests....323
Reference: AUnit Test Annotations....325
Reference: AUnit Test Actions....326
Web Service Testing....327

©2026 Workday, Inc. All rights reserved

Test Web Services Using a WSDL File..... 327
Test Web Services Using an XSD File..... 328
Test Individual Web Services from the Schema Explorer..... 329
Test Get Operations and Request Messages Using the Request Builder Wizard..... 330
Configure Web Service Tester Authentication..... 330
Configure Web Service Tester X.509 Authentication..... 331
Configure Web Service Tester Advanced WS-Security..... 332
Concept: Web Service Testing..... 332
Reference: Web Service Tester Toolbar Buttons..... 333
Assembly Debugging..... 334
Debug an Assembly..... 334
Concept: Assembly Debugger..... 335

Studio Management..... 336
Studio Preferences..... 336
Concept: Assembly Editor Preferences..... 336
Concept: Connections Preferences..... 337
Concept: Deployment Preferences..... 337
Concept: HTTPS Preferences..... 337
Concept: Ivy Preferences..... 338
Concept: Linking Preferences..... 338
Concept: MVEL Validation Preferences..... 338
Concept: Packaged Integrations Preferences..... 338
Concept: Runtime Preferences..... 338
Concept: Server Preferences..... 338
Concept: Splash Screen Login Preferences..... 338
Concept: Status Bar Preferences..... 339
Concept: Update Preferences..... 339
Concept: XPath Explorer Preferences..... 339
Reference: Assembly Debug Runtime Preferences..... 339
Reference: Assembly Debug View Preferences..... 340
Reference: Cloud Explorer Preferences..... 340
Reference: Consolidated Report Viewer Preferences..... 340
Reference: Java Project Settings Preferences..... 341
Reference: Notification Preferences..... 341
Reference: Process Monitor Preferences..... 342
Reference: Web Service Tester Preferences..... 342
Studio Updates..... 342
Update Studio..... 342
Workday Assembly Runtime Changes..... 343
Concept: Assembly Versions..... 343
Reference: Version Changes at the Schema Level..... 343
Reference: Version Changes at the XML Level..... 347
Reference: Saxon XSLT Processor Changes..... 349

Sample Assemblies..... 351
Example: Run the Hello Cloud Sample Assembly..... 351
Example: Run the Hello Workday Web Services Sample Assembly..... 352
Example: Run the Common Usage Patterns Sample Assembly..... 353
Example: Run the Debugging Sample Assembly..... 354
Example: Run the Workday-In Features Sample Assembly..... 355
Example: Run the Error Handling Sample Assembly..... 357
Example: Run the AUnit Sample Tests..... 358
Example: Run the FOPStep Sample Assembly..... 359

©2026 Workday, Inc. All rights reserved

Example: Run the Extract Organization Data for a Worker Sample Assembly....360
Example: Run the PrismAnalytics Sample Assembly....361
Third-Party Payroll Interface Sample....363
Steps: Run the Payroll Interface Sample....363
Create the Payroll Interface Security Group....363
Create the PayData Extract Integration System....364
Test the PayData Extract Integration System....365
Deploy the Studio PayData Integration....366
Assign an Integration System User to the PayData Integration....367
Link the PayData Extract and PayData Integrations....368
Launch the Linked Integrations....368

©2026 Workday, Inc. All rights reserved

Setup Considerations: Workday Studio

Note: The solutions described in this section are not part of the Workday Service. See Legal Notice for details.

You can use this topic to help make decisions when planning your configuration and use of Workday Studio. It explains:

- Why to set it up.
- How it fits into the rest of Workday.
- Downstream impacts and cross-product interactions.
- Security requirements and business process configurations.
- Questions and limitations to consider before implementation.

Refer to detailed task instructions for full configuration details.

What It Is

Studio is an Eclipse™-based integrated development environment (IDE) that enables you to:

- Build custom integrations.
- Exchange bulk data with external endpoints.
- Upload integrations to Workday.

Studio is an external software application that you install separately from the rest of Workday.

Business Benefits

Studio improves productivity by enabling you to:

- Create integrations that support greater design flexibility than Enterprise Interface Builder (EIB) or connector templates.
- Debug the XML source code of your custom integrations.
- Export and import large volumes of data through Workday.
- Transform large datasets into a specific format. Example: You can transform CSV datasets into XML format.

Use Cases

You need:

- A custom integration that isn't available through EIB or connector templates. Example: An integration that implements XML-to-text (XTT) transformations on a dataset.
- An inbound integration to import bulk data to Workday from multiple data sources.
- An outbound integration that exports bulk data from Workday in a specific format.
- To conduct unit tests on the XML source code of a custom integration.

Questions to Consider

Question	Considerations
Does your organization have a long-term support agreement with Oracle?	If yes, you can install Studio using Oracle JDK 1.8. If no, you can install Studio using OpenJDK 8.
Which platform are you using to install Studio?	Install the appropriate installation file for your platform:

©2026 Workday, Inc. All rights reserved

Question	Considerations
	• MacOS (64-bit) • Windows (32-bit) • Windows (64-bit)
Have you attended training for Studio?	For learners interested in taking Studio training, Workday offers a Creating Integrations Using Workday Studio course. Learn more in the Training Catalog.

Recommendations

Workday recommends that you deploy and test your custom Studio integrations in your Preview Sandbox environment before transferring them to your Production tenant.

Requirements

Studio integrations require a Java™ Development Kit (JDK) compatible with Java version 8. In addition, the Eclipse IDE requires a JDK compatible with Java version 17. If you:

- Have a license agreement with Oracle, use Oracle JDK 1.8 and Oracle JDK 1.17.
- Don't have an agreement with Oracle, use OpenJDK 8 and OpenJDK 17.

Both JDKs are required.

Limitations

Studio integration output files can be a maximum of 5 GiB (compressed) for an individual file or for the total of all files.

Tenant Setup

You can access the source code of Studio integrations by selecting the **Require Source Code** check box on the **Edit Tenant Setup - Integrations** task in Workday. Accessing the source code of Studio integrations enables you to identify bugs and improve efficiency.

Security

Consider setting up these domains in the Integration functional area.

Domains	Considerations
Integration Build	Deploy integrations from Studio to Workday.
Integration Configure	Edit Studio integrations in Workday.
Integration Debug	Debug Studio integrations in Workday.
Integration Event	Launch Studio integrations in Workday.
Integration Security	Create an integration system user (ISU) account for your Studio integrations in Workday.

Business Processes

For Studio integrations, you can configure the:

- *Integration Process Event* business process in the Integration functional area to control how an integration runs.

©2026 Workday, Inc. All rights reserved

- *Integration* business process step to launch an integration during a business process. Example: You can add an integration step to the *Complete Form I-9* business process. This step launches an integration to send data for the worker in the business process.

Reporting

Workday provides these reports for Studio integrations:

Report	Description
Integration Events	Displays: • A summary of integration events that are in process or complete. • The status of each integration event.
View Integration System	Displays all integration systems available in your Workday tenant.

Connections and Touchpoints

Studio interacts with different parts of Workday depending on the product area, purpose, and data sources of your integration:

- Inbound integrations add and update data in Workday.
- Outbound integrations extract data from Workday, but don't change the data in Workday.

Workday offers a Touchpoints Kit with resources to help you understand configuration relationships across your tenant. Learn more about the Workday Touchpoints Kit on Workday Community.

Studio Fundamentals

Download and Install Workday Studio for Windows

Context

Workday Studio is built on the Eclipse development platform. It comprises more than 150 Eclipse plugins in addition to platform and third-party dependencies. To use it, you need 2 versions of the Java™ Development Kit (JDK):

- Eclipse requires a JDK compatible with Java version 17.
- The Studio runtime requires a JDK compatible with Java version 8.

Install both JDKs and then ensure that:

- Your Java Runtime Environment is set to Java 8.
- Your JDK compiler compliance level is set to Java 1.8.

©2026 Workday, Inc. All rights reserved

Steps

1. Download and install a JDK that's compatible with Java version 17. You can download the free open-source implementation, OpenJDK, from a variety of sources.

   Example:

   - Select the Zulu OpenJDK 17 that’s appropriate for your operating system.

   Note:

   - Users of Apple Silicon must use an Apple Silicon ARM JDK17. An x86_64 JDK will not work.
   - Studio requires a JDK that's compatible with Java 17 specifically, not Java 17 as a minimum.
2. Download and install a JDK that's compatible with Java version 8. Again, you can choose the free open-source implementation, OpenJDK.

   Example:

   - Select the Zulu OpenJDK 8 that’s appropriate for your operating system

   Note:

   - Users of Apple Silicon must use an Apple Silicon ARM JDK8. An x86_64 JDK will not work.
   - Studio requires a JDK that's compatible with Java 8 specifically, not Java 8 as a minimum.
3. Download the Workday Studio for Windows installer.
4. To launch the installation, double-click the installation executable file. To install Studio in the C:\Program Files folder, you must run the executable as an Administrator by right-clicking it and selecting **Run as Administrator**.

   Note:

   You can use the Command Prompt application to run a silent Studio installation. In Command Prompt, navigate to the directory where you saved the installation file and enter:

   ```
   <Installation_Filename>.exe /S /J=<Java_Home> /D=<Installation_Directory>
   ```

   where the argument:

   - `/S` specifies silent installation.
   - `/J` specifies the JDK 17 path.
   - `/RD` installs Workday Report Designer.
   - `/D` specifies the installation folder (this argument must be the last in the command).

   Example: `workday-studio-x86_64.exe /S /J=C:\Program Files\Zulu\zulu17.62.17-ca-jdk17.0.18-win_x64 /D=C:\Studio`
5. Read the license agreement, then select the terms acceptance check box. Click **Next**.
6. Browse to the location of your JDK 17 installation and click **Next**. Example: of JDK installation locations: C:\Program Files\Zulu\zulu17.62.17-ca-jdk17.0.18-win_x64.
7. If the JVM details are correct, click **Confirm**. Otherwise, click **Back**.
8. Select the **Report Designer** pack to install it alongside Workday Studio. You can also install Report Designer later from within Studio.
9. Browse to the directory where you wish to install Studio. Click **Next**.
10. When Studio displays the Installation Complete message, click **Next**, then click **Finish**.

    Note: Eclipse with Java 17 can't use a Workspace created by Eclipse with Java 8. On launch, you can update your existing Workspace to make it compatible with Java 17. Alternatively, you can create an entirely new compatible Workspace. Once you've updated your Workspace, you can't use it with earlier versions of Studio.

©2026 Workday, Inc. All rights reserved
Workday Proprietary and Confidential

11. To ensure that Java 8 is available as a Java Runtime Environment in Studio, navigate to its **Preferences** panel: **Window > Preferences > Java > Installed JREs**

    If Java 8 is missing, click **Add** and select *Standard VM* as the JRE Type. Specify the **Directory** where you installed the JDK 8.
12. Ensure that the check box beside the Java 8 JRE is selected.
13. Check the Execution Environments for a perfect match: **Window > Preferences > Java > Installed JREs > Execution Environments**

    Select **JavaSE-1.8** in the **Execution Environments** panel, then select the environment described as a 'perfect match' in the **Compatible JREs** panel.
14. In Studio's **Preferences** panel, navigate to **Java > Compiler**.
15. Under **JDK Compliance**, from the **Compiler compliance level** drop-down menu, select *1.8*.
16. Click **Apply and Close**.

    Note: Before configuring Studio, refer to this overview of Workday Data Centers. The data center page for your tenants contains information that your IT department can use to configure firewalls and ensure proper Studio access to Workday.

Download and Install Workday Studio for macOS

Context

Workday Studio is built on the Eclipse development platform. It comprises more than 150 Eclipse plugins in addition to platform and third-party dependencies. To use it, you need 2 versions of the Java™ Development Kit (JDK):

- Eclipse requires a JDK compatible with Java version 17.
- The Studio runtime requires a JDK compatible with Java version 8.

Install both JDKs and then ensure that:

- Your Java Runtime Environment is set to Java 8.
- Your JDK compiler compliance level is set to Java 1.8.

Steps

1. Download and install a JDK that's compatible with Java version 17. You can download the free open-source implementation, OpenJDK, from a variety of sources.

   Example:

   - Select the Zulu OpenJDK 17 that’s appropriate for your operating system.

   Note:

   - Users of Apple Silicon must use an Apple Silicon ARM JDK17. An x86_64 JDK will not work.
   - Studio requires a JDK that's compatible with Java 17 specifically, not Java 17 as a minimum.
2. Download and install a JDK that's compatible with Java version 8. Again, you can choose the free open-source implementation, OpenJDK.

   Example:

   - Select the Zulu OpenJDK 8 that’s appropriate for your operating system

   Note:

   - Users of Apple Silicon must use an Apple Silicon ARM JDK8. An x86_64 JDK will not work.
   - Studio requires a JDK that's compatible with Java 8 specifically, not Java 8 as a minimum.

©2026 Workday, Inc. All rights reserved

3. Download the correct macOS Studio installer:
    - Apple Silicon
    - x86, 64 bit
4. To launch the installation, open the installation .jar file.
5. Read the license agreement, then select the terms acceptance check box. Click **Next**.
6. Browse to the location of your JDK 17 installation and click **Next**. Example: /Library/Java/JavaVirtualMachines/zulu-17.jdk/Contents/Home.
7. If the JVM details are correct, click **Confirm**. Otherwise, click **Back**.
8. Select the **Report Designer** pack to install it alongside Workday Studio. You can also install Report Designer later from within Studio.
9. Browse to the directory where you wish to install Studio. Click **Next**.
10. When Studio displays the Installation Complete message, click **Next**, then click **Finish**.

    Note: Eclipse with Java 17 can't use a Workspace created by Eclipse with Java 8. On launch, you can update your existing Workspace to make it compatible with Java 17. Alternatively, you can create an entirely new compatible Workspace. Once you've updated your Workspace, you can't use it with earlier versions of Studio.
11. To ensure that Java 8 is available as a Java Runtime Environment in Studio, navigate to its **Preferences** panel: **Workday Studio > Preferences > Java > Installed JREs**

    If Java 8 is missing, click **Add** and select *Standard VM* as the JRE Type. Specify the **Directory** where you installed the JDK 8.
12. Ensure that the check box beside the Java 8 JRE is selected.
13. Check the Execution Environments for a perfect match: **Workday Studio > Preferences > Java > Installed JREs > Execution Environments**

    Select **JavaSE-1.8** in the **Execution Environments** panel, then select the environment described as a 'perfect match' in the **Compatible JREs** panel.
14. In Studio's **Preferences** panel, navigate to **Java > Compiler**.
15. Under **JDK Compliance**, from the **Compiler compliance level** drop-down menu, select *1.8*.
16. Click **Apply and Close**.

    Note: Before configuring Studio, refer to this overview of Workday Data Centers. The data center page for your tenants contains information that your IT department can use to configure firewalls and ensure proper Studio access to Workday.

Concept: Workday Studio

Workday provides 2 ways to perform integrations:

- Connectors: Packaged integrations that connect to a variety of common third-party services.
- Enterprise Interface Builder (EIB): A simple tool for creating integrations without doing any programming.

Workday Studio is a third option that enables you to build, own, and support much more sophisticated integrations, which Workday hosts and runs on your behalf. Unlike Connectors and EIB, Studio isn't part of Workday. It's an Eclipse-based Integrated Development Environment (IDE) that you download separately. As with any other integrations in your tenant, you can launch, schedule, and audit the integrations you build with Studio. You can launch Studio-created integrations from Studio itself or from the Workday application.

The target user for Studio:

- Has a development skill set.
- Is familiar with integration programming concepts.
- Is comfortable using an IDE.

©2026 Workday, Inc. All rights reserved

The Studio platform has 4 principal components:

- The Studio IDE.
- Integrations, collections, and the Cloud Repository.
- The Workday application.
- Cloud Runtimes.

The Studio IDE

Studio is an Eclipse-based IDE. You use it to construct integration flows that:

- Access information from Workday.
- Manipulate it.
- (Optional) Send it somewhere.

These flows are called assemblies. You create assemblies by adding and connecting graphical elements called components and steps in the Studio Assembly Editor.

Studio includes components and steps for many actions, including splitting, transforming, and routing. Workday-specific components have a built-in description of how Workday represents information.

You can test your integration in Studio before deploying it directly to your Workday tenant.

Studio supports the use of Java extensions in assembly projects. You can supply your own Java code and configure it as a Spring bean. In addition, you can package third-party Java libraries within your integrations.

Integrations, Collections, and the Cloud Repository

The Cloud Repository is Workday's term for its Studio integration storage. Before deployment, Studio assigns each integration to a collection. A collection can contain multiple integrations. An integration can belong to more than 1 collection. This means you can easily reuse assemblies and subassemblies across integrations and tenants.

Only 1 instance of a particular collection can run at any time. If you modify an assembly project within a collection and redeploy the project, it replaces any previously deployed version.

The binary file that Studio deploys to the Workday Cloud Repository is called a CLAR (CLoud ARchive). A CLAR contains:

- All of the artifacts from the included integration projects, such as their assembly.xml, XSLT, and Text Schema files.
- Any compiled Java code or third-party Java libraries that developers have added to the integration projects.

As binary objects, CLARs don't include source-code artifacts like Java source files.

The Cloud Repository stores CLARs in an encrypted format. However, don't store information like passwords directly in your integrations. Configure sensitive details in the Workday application instead so you can manage them dynamically.

The Workday Application

Workday uses an abstraction called an integration system to model integrations. In Workday, for each integration you can specify:

- Which users have permission to launch the integration and the rights the integration has while executing.
- How and when to notify different Workday users when the integration executes.

Because you design and configure your Studio integrations separately, you can redeploy an integration without affecting its configuration.

©2026 Workday, Inc. All rights reserved

Workday enables integrations to launch in a number of different scenarios. You can:

- Manually launch integrations.
- Schedule integrations to run automatically as an event.
- Launch an integration automatically as part of a business process.
- Trigger launch remotely using a web service client to invoke a Workday Web Service.

You can use the Workday **Process Monitor** report to monitor the execution of an integration.

Cloud Runtimes

When an integration launches, Workday:

- Generates an integration event.
- Retrieves the integration from the Cloud Repository.
- Assigns a protected set of processing resources called a Cloud Runtime.
- Executes the integration in the Cloud Runtime.

If Workday is unable to execute an integration, it updates the **Process Monitor** with an integration launch status of `Failed`.

If Workday encounters a problem during execution, such as the unavailability of an external resource, it terminates and updates the **Process Monitor** with an integration launch status of `Completed with errors`.

Note: It's the integration developer's responsibility to deal with integrations marked as `Completed with errors`.

Workday terminates integrations that take longer than 2 hours to execute. Developer-supplied Java code can't, in general, read from or write to the file system or Java system properties. Developer-supplied Java code can't spawn new Java threads.

Security and Privacy

Studio supports 2 Cloud environments automatically:

- Sandbox
- Production

You can add connections to other Workday environments. It's your sole responsibility to ensure that you deploy the correct integrations to the appropriate environment. Standard Workday Web Services security policies apply to all data accessed using Studio integrations.

You can create assemblies offline in Studio, but you must have the appropriate security configuration to integrate them with Workday. Once deployed, integrations launch and run with the credentials and privileges of the user who runs them.

Workday has no insight into what your integration looks like or what it's designed to do. It's your responsibility to ensure that it behaves as expected. Your integration's interactions with your Workday tenant happen over the Workday Public APIs, so you inherit all the associated robustness and security guarantees.

Concept: Integration Development Lifecycle

You develop and monitor Studio integrations in these distinct phases, using either Workday Studio or the Workday application as appropriate:

Phase	Description
Integration Design	In Studio, develop the assemblies that form your integration. Optionally, create assembly modules,

©2026 Workday, Inc. All rights reserved

Phase	Description
	which enable you to reuse and share integration patterns and configurations. Each assembly has at least 1 **workday-in** transport, which denotes where execution begins. Each **workday-in** transport can optionally define: • Launch parameters, which enable a client to pass information to the integration as input. Example: A payroll integration might specify that it requires 2 launch parameters: `payPeriod` and `payGroup`. This enables the integration's users to specify the month and employee group to use when calculating payroll. • Workday services that the assembly uses at runtime. Workday provides generic integration services for attribute mapping, sequence generation, transaction logging, document retrieval and delivery. It also provides a listener service, a report service, and a custom object service.
Integration Deployment	In Studio, connect to the Workday Cloud Repository to deploy your integration. For each **workday-in** transport defined in an integration, Workday registers an abstraction called an integration system. An integration system includes any required launch parameters and services.
Integration Configuration	In Workday, you can configure certain aspects of your integration, including permissions and notifications.
Integration Launching	The integration system enables Workday to launch and track your integration. But to make development and testing easier, you can also launch from Studio.
Integration Monitoring and Troubleshooting	Track the progress of your integration in either Studio or Workday. In Studio, you can monitor the execution of the integration on the **Process Monitor**. You can also view a consolidated report of the integration process after it has completed.

Concept: MediationContext Object

During assembly execution, Workday creates a temporary holder object called a `MediationContext`. The `MediationContext` contains 3 main types of information:

- `MediationMessage`: an object containing the text or binary data message that Studio ultimately sends to an external target using an Out-transport. Examples: HTTP-Out, FTP-Out.
- Variables: temporary holders for message data.
- Properties: a map of name-object pairs that assembly elements use for storing transient data outside of a message.

©2026 Workday, Inc. All rights reserved

All processing in assemblies involves either reading from or writing to the `MediationContext`.

Note: When you access the `MediationContext` object using MVEL, the accessor is `context`. Example: `context.getErrorMessage()`. In Java, it's `mediationContext`. Example: `mediationContext.getErrorMessage()`.

Concept: Integration Systems and Events

Integration Systems

In Workday, an integration system is an abstraction of an integration with an external system or a Workday endpoint. Integration systems provide the framework by which Workday launches and tracks integrations. You can use an integration system to configure:

- Security: Specify who can launch the integration and what rights the integration has while executing.
- Notifications: Specify who gets notified as an integration executes.
- Parameters: You can parameterize the invocation of an integration. Example: you can configure the endpoints and user credentials an integration needs to upload data to an external SFTP site.
- Triggers: You can configure the integration to be triggered:
    - Manually.
    - As part of a subscription.
    - Using an API.
    - On a schedule.
    - As part of a Business Process.

Workday represents all integrations, regardless of their origin, as integration systems.

When you develop an integration using Workday Studio and deploy it to Workday, you automatically register a new integration system for every **workday-in** transport in that integration.

Integration Events

When an integration launches, Workday creates an object called an integration event. This object is unique to each invocation and contains an ID called an integration event ID. Workday uses this unique ID to track all executing integrations. You can view integration events using the **Process Monitor** in Workday.

As well as creating an integration event object, Workday creates an integration event request message and sends it to the integration. This request contains information about the integration system and the integration event ID. It also contains values for any parameters supplied at launch time. The integration can then use this information when requesting information from Workday Web Services. The integration can also use this information to send update information back to Workday. Example: a status message that tells the user that it has processed 100 employee records successfully. Workday adds these updates to the integration event object.

Reference: Interface Fundamentals

The Workday Studio interface is composed of views. To display or hide views, select **Window > Show View > Other...**, then expand the **Workday** folder.

The most commonly used views in Studio are:

View	Description
Project Explorer	Studio's project navigator. Displays projects saved in the current workspace. Each folder contains all

©2026 Workday, Inc. All rights reserved

View	Description
	of the relevant files for that project and references the collections to which the project belongs.
Cloud Explorer	Displays the integration cloud servers to which Studio can deploy, as well as the currently deployed collections. You can add connections to multiple environments. If connection, authentication, or authorization errors occur, Studio displays an error or warning decorator on the connection icon.
Palette	Displays all the assembly components that you can add. Features 3 tabbed subviews: • **Design:** displays the assembly diagram. • **Source:** displays the XML source. • **Message Builder:** a graphical view that simplifies defining the message content for assembly elements such as the **write** step.
Assembly Editor	A diagrammatic representation of the assembly. To add new components and steps, drag and drop them from the **Palette**. To connect components, hover over a source component to display the connection handles, then drag the handle to the target component.
Properties	Displays configuration details for the assembly element currently selected in the assembly diagram. Edit properties and parameters using the tabs that display along the left-hand side of the view.
Process Monitor	Displays information about the assembly process when you launch an integration. Select a process to display its details on the **Process Details** view.
Schema Explorer	Enables you to explore the contents of WSDL and XML Schema files and provides several utility features for developing assemblies that interact with web services.
Consolidated Report Viewer	Enables you to analyze assembly processing steps in an integration.

©2026 Workday, Inc. All rights reserved
Workday Proprietary and Confidential

Integration Design

Assemblies

Create an Assembly Project

Context

Assemblies define message-based integration processes consisting of annotations, transports, mediation components, mediation steps, and error handlers.

Steps

1. Select **File > New > Workday Assembly Project**.
2. In the **Project name** field, enter a name for your project.

   Project names must not contain spaces.
3. From the **Target Runtime** list, select *Workday Runtime*.
4. From the **Configurations** list, select 1 of these configuration options:

Option	Description
Assembly	Creates a simple assembly project.
Assembly (with Java support)	Creates a project containing a Java facet.
Custom	Enables you to specify the project facets for your assembly project.

5. Click **Next**. Enter a **Collection name**.

   The **Collection name** field automatically populates with your project name.
6. Click **Finish** to create your project.

   Note: If you create an assembly project outside your workspace, you can give it a new name in Studio without changing the folder name in the original location. This also applies to projects you import from outside your workspace. To do so, select **File > Rename**.

Result

Your new assembly project displays on the **Project Explorer** view. It contains an assembly.xml file, which you can now open in the Assembly Editor.

Next Steps

Using the Assembly Editor, you can define your integration by adding and linking transports, mediation components, and mediation steps.

Add Steps to Components

Prerequisites

Create an assembly project.

©2026 Workday, Inc. All rights reserved

Context

Steps perform individual operations on a message as it passes through an integration. You can add steps to **async-mediation** and **sync-mediation** components only.

Steps

1. Open the assembly in the Assembly Editor.
2. Add an **async-mediation** or **sync-mediation** component to the assembly.
3. Drag a step from the **Palette** into your **async-mediation** or **sync-mediation** component. Place steps in order of execution. As you complete the task, consider:

Palette Category	Description
Transform	Displays mediation steps that enable you to transform messages as they pass through your integration.
Logging & Eventing	Displays mediation steps that enable you to log messages as they pass through your integration.
Message Manipulation	Displays mediation steps that enable you to manipulate messages as they pass through your integration.
Validation	Displays mediation steps that enable you to validate messages as they pass through your integration.
Encoding	Displays mediation steps that enable you to encode messages as they pass through your integration.
Other	Displays miscellaneous mediation steps that enable you to perform individual operations on messages as they pass through your integration.

4. Click **Save**.

Result

Studio displays the new step within the component on the assembly diagram.

Next Steps

Configure the properties of the new step.

Add Child Elements

Prerequisites

Create an assembly project.

Context

Some assembly elements support child elements. Child elements are optional.

©2026 Workday, Inc. All rights reserved

Steps

1. Open the assembly in the Assembly Editor.
2. Add an assembly element that supports child elements to the assembly diagram.
3. Hover your cursor over the title of the assembly element to display its child element options.
4. Click a child element option to add it to your assembly element.

Result

Studio displays the child element on your assembly element.

Next Steps

Configure the properties of the child element.

Connect Assembly Elements

Prerequisites

Create an assembly project.

Context

Connections between assembly elements define the path of a message through your assembly. You can connect assembly elements using the **Outgoing connection** handle (blue square), or the **Incoming connection** handle (orange square) in the Assembly Editor.

Steps

1. Open the assembly in the Assembly Editor.
2. Add 2 assembly elements to the assembly diagram.
3. Place your cursor over the title of the first assembly element to display its connection handles. Drag the **Outgoing connection** handle (blue square) from the first assembly element to the second assembly element.
4. Alternatively, place your cursor over the title of the second assembly element and drag the **Incoming connection** handle (orange square) to the first assembly element.

Result

The message output from the first assembly element becomes the input for the second assembly element.

Configure Assembly Element Properties

Prerequisites

Add assembly elements to an assembly diagram.

Context

Each element on the Workday Studio **Palette** has a **Properties** view where you can configure its functionality.

Steps

1. Right-click an assembly element and select **Show Properties View**.

©2026 Workday, Inc. All rights reserved

2. Required properties have an asterisk in the upper left-hand corner of the icon. As you complete the task, consider:

Property Icon	Property Description
	Values that must be unique within the assembly.
	Input to the assembly element.
	Output from the assembly.
	MVEL expressions.
	MVEL templates.
	Boolean values.
	Options.
	Numeric values.
	Attribute values.
	Java class implementations.
	Spring beans that reference Java class implementations.

Note: When properties are MVEL-capable, the relevant MVEL icon is added to the property's own icon. Example:

3. To compare property views:
   a) Select a component in the assembly diagram.
   b) Hold down the Ctrl key and select a second component.
   c) Right-click either component and select **Compare Properties**. The first component's **Properties** view displays on the left. The second component's **Properties** view displays on the right, and is pinned to that selection. The view on the left changes when you select a different component.
4. In MVEL-capable property fields, press Ctrl+space to access content assist, a set of context-sensitive code completion options.

   Note: Content assist is only available in assembly projects that contain the Java facet.

Manage Collections and Projects

Prerequisites

Create an assembly project.

©2026 Workday, Inc. All rights reserved

Context

In Workday Studio, collections enable you to package and deploy multiple projects. A project always belongs to at least 1 collection, and can be a member of multiple collections simultaneously.

Steps

1. Click **Workday > Manage Collections**.

   If you expand a collection in the left window pane, Studio displays the projects it contains. Projects that aren't a member of that collection display in the right window pane.
2. As you complete the task, consider:

Button	Description
New Collection	Adds a new collection to your workspace.
Rename Collection	Renames the selected collection.
Remove Collection	Removes the selected collection from your workspace.
Add to Collection	Adds the selected project to a collection.
Add All Integrations to the Collection	Adds all projects to the selected collection.
Remove the Integration from Collection	Removes the selected project from a collection.

   Note: Ensure that all projects in a collection have the same assembly version. Otherwise, Studio displays an error marker. You can configure the **Version** property value for the assembly on the **Properties** view in the Assembly Editor.
3. Click **Finish**.

Declare Namespace Prefixes

Prerequisites

Create an assembly project.

Context

In Workday Studio, namespace prefixes are abbreviations with predefined URI values. You can declare namespace prefixes to refer to specific URI values in your assemblies.

Steps

1. Open an assembly in the Assembly Editor.
2. Right-click on the background of the assembly diagram, and select **Show Properties View**.
3. Click the **Namespace Prefixes** tab.
4. Click **Add** to add a new namespace prefix. As you complete the task, consider:

Option	Description
Prefix	Enter a prefix name here.
Namespace	Enter the namespace URL here.

5. To edit your new namespace prefix, click **Edit**.
6. To remove your new namespace prefix, click **Remove**.

©2026 Workday, Inc. All rights reserved

Result

The namespace prefix for the assembly displays in the XML schema.

Create Path Variables

Prerequisites

Create an assembly project.

Context

Path variables specify the location of resource files in your workspace folder. You can use workspace path variables when you link platform resources in your assembly projects.

Steps

1. Select **Windows > Preferences > General > Workspace > Linked Resources**.
2. Click **New**.

   As you complete the task, consider:

Option	Description
Name	Enter the name of your path variable.
Location	Enter the file location of your path variable.

3. Click **Ok**.

Result

Your new path variable now specifies the location of resource files in your workspace folder.

Concept: Assembly Palette

The **Palette** displays all the assembly elements available in Studio. To add an assembly element to an assembly diagram, drag it from the **Palette** onto the Assembly Editor. The **Palette** lists assembly elements under these categories:

Palette Category	Description
Favorites	Displays your favorite assembly elements. You can add assembly elements to this category by clicking the empty star icon beside them. You can remove elements from this category by clicking the highlighted star icon beside them.
Annotation	Displays annotations that enable you to structure, organize, and label your assembly diagram.
Module Library	Displays module assembly elements that enable you to configure Workday Web Service calls and global error handlers for your integration.
In Transports	Displays in-transport assembly elements that receive messages from Workday or a local resource.
Out Transports	Displays out-transport assembly elements that send messages to a resource.

©2026 Workday, Inc. All rights reserved

Palette Category	Description
Components	Displays mediation components that enable you to manipulate and transform messages as they pass through your integration.
Common Components	Displays prepackaged subassemblies that you can use to build your integrations.
Workspace Components	Displays custom subassemblies that you've created and saved to your workspace.
Error Handlers	Displays error handlers that enable you to catch errors in your integrations.
Transform	Displays mediation steps that enable you to transform messages as they pass through your integration.
Logging & Eventing	Displays mediation steps that enable you to log messages as they pass through your integration.
Message Manipulation	Displays mediation steps that enable you to manipulate messages as they pass through your integration.
Validation	Displays mediation steps that enable you to validate messages as they pass through your integration.
Encoding	Displays mediation steps that enable you to encode messages as they pass through your integration.
Other	Displays mediation steps that enable you to perform individual operations on messages as they pass through your integration.
Connections	Displays the **Route To** and **Route Response To** connections that enable you to define the path of a message through your integration.

To manually filter the assembly elements displayed on the **Palette**, use the **type filter text** field. Studio also provides these preconfigured category filters directly underneath the **type filter text** field:

Category Filter	Description
Restore	Clears previous filter selections.
Transports	Displays only transport **Palette** categories.
Components	Displays only mediation component **Palette** categories.
Steps	Displays only mediation step **Palette** categories.
Handlers and Connections	Displays only error handler and connection **Palette** categories.
Filter Categories	Displays the **Palette Category Filter** window.
Expand All	Expands all **Palette** categories.

©2026 Workday, Inc. All rights reserved

Category Filter	Description
Collapse All	Collapses all **Palette** categories.

Studio automatically displays the **Palette** beside the Assembly Editor. To display the **Palette** in a separate view, select **Window > Show View > Other > General > Palette** and click **OK**. You can also change the layout of the **Palette** by right-clicking on it and selecting **Layout**.

Related Information
Reference
Workday 31 What's New Post: Workday Studio

Concept: Components and Steps

You can add steps to **async-mediation** and **sync-mediation** components only.

Workday Studio provides steps that support operations such as transformation, validation, and logging. You can also write custom steps using Spring beans.

The **async-mediation** component supports 1 processing chain, handling request messages only. The **sync-mediation** component supports 2 processing chains, handling request messages and request-response messages.

Steps have **Input** and **Output** properties that specify where they obtain data as input and where they direct the output. There are 5 possible values:

Option	Description
message	The entire message, including rootpart and attachments. When you specify this option, Studio creates a new message, effectively removing any attachments.
soapbody	The contents of the SOAP body from the rootpart of the message.
attachment	A MIME attachment to the message. Specify the index value of the attachment you wish to access. You can also add a new attachment to the message by specifying an index value of -1.
rootpart	The content of the rootpart of a MIME message, or if the message isn't MIME, the message itself. When specified for the output attribute, this value preserves any attachments on the original message.
variable	A variable from the `MediationContext`. Specify the name of the variable to use.

Concept: Assembly Annotations

Annotations enable you to structure, organize, and label your assembly diagrams. Studio provides these options under the **Annotations** section of the **Palette**:

- **Rectangle**
- **Ellipse**
- **Note**
- **Text**
- **Swimlane**

©2026 Workday, Inc. All rights reserved

You can:

- Use the **Rectangle**, **Ellipse**, **Note**, and **Text** annotations to add general explanatory text to your assembly.
- Use the **Show/Hide annotations** button to display or hide the **Rectangle**, **Ellipse**, **Note**, and **Text** annotations. You can only add new annotations to assembly diagrams when **Show annotations** is active.
- Use the pin icon on **Rectangle**, **Ellipse**, **Note**, and **Text** annotations to pin them to your assembly. Pinned annotations always display on your diagrams, even when **Hide annotations** is active.

Swimlanes

Swimlanes enable you to organize assembly components into groups.

You can:

- Display or hide swimlane descriptions by clicking the **Show/Hide swimlane descriptions** button.
- Expand and collapse all swimlanes by clicking the **Expand all swimlanes** and **Collapse all swimlanes** buttons.
- Expand and collapse individual swimlanes by clicking the plus and minus icons in their headers. Swimlane status is persistent. When you open an assembly, swimlanes display in the collapsed or visible state they were in when the assembly was saved.

Swimlanes are nestable.

Related Information
Reference
Workday 31 What's New Post: Workday Studio

Concept: Linked Resource Files for Assembly Projects

Linked resources are files stored on your workspace that you can link to in multiple assembly projects.

To view a list of linked resources for a project, right-click on the project in the **Project Explorer** view. Select **Properties > Workday > Linked Resources** to display the **Linked Resources** window.

To remove a linked resource file from a project, right-click on the resource file in the **Project Explorer** view and select **Remove linked resource**.

To copy a linked resource to the project, right-click on the resource file in the **Project Explorer** view and select **Convert linked resource to copy**.

Concept: Quick Outline View

The **Quick Outline** view provides an easy way to locate various elements of your assembly. To access the **Quick Outline** view, press Ctrl+O when the **Assembly Editor** has focus.

Note: You can't access the **Quick Outline** view from the **Window > Show View** menu.

The **Quick Outline** view enables you to search your assembly by:

Search Type	Description
MVEL Expression	Search MVEL expression and template properties. Use this view to search for occurrences of text values.
ID	Search for assembly component IDs.
Variable	Search for variable names. Studio categorizes variables into 2 groups: those defined for

©2026 Workday, Inc. All rights reserved

Search Type	Description
	accessing steps, and those defined for **write** steps.

To select a search type, click the relevant button under the search field at the top of the view.

To search, enter text in the search field. When you select an MVEL expression, component ID, or variable in the list of search results, Studio jumps to the relevant component in the assembly diagram.

Reference: Assembly Properties

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the assembly.
Version	`version`	The assembly version number.
Audit Log	`audit-log`	The message log settings for the assembly. The options are: • **True**: Enables and stores all logs. • **False**: Disables all logs. • **On Error**: Enables all logs and stores only error logs.
Callback Handler Bean	`callback-handler-bean`	The Spring bean that handles asynchronous communication for the assembly.
Beans	`beans`	The Spring beans in the assembly.
Namespace Prefixes	`xmlns`	The namespace prefixes in the assembly.

Reference: Swimlane Properties

Common Tab

Property	Description
Name	The name of the swimlane. Displays in the header of the swimlane.
Description	A description of the swimlane. Displays when you select **Show swimlane descriptions**.
Orientation	The orientation of the swimlane. The options are: • **HORIZONTAL** • **VERTICAL**
Alignment	The alignment of components in the swimlane. The options are: • **BEGINNING** • **MIDDLE** • **END**

©2026 Workday, Inc. All rights reserved

Property	Description
Label Alignment	The alignment of the swimlane label. The options are: • **LEFT** • **CENTER** • **RIGHT**
Top border color	The color of the top border on the swimlane.
Line color	The color of the side and bottom borders of the swimlane.
Fill color	The background color of the swimlane.
Font	The font of the swimlane label.
Line style	The style of the side and bottom borders of the swimlane. The options are: • **SOLID** • **DASH** • **DOT** • **DASHDOT** • **DASHDOTDOT**

Reference: Connection Line Properties

Common Tab

Property	Description
Line Type	The type of line used to display **Route To** and **Route Response To** connections on an assembly diagram. The options are: • **straight** • **curved**

Assembly Transports

Use Enumerations as Launch Parameters in Studio

Prerequisites

Create an assembly project containing a **workday-in** transport.

Context

In Workday Studio, you can use enumerations as integration launch parameters. This restricts the values you can select in the **Integration Criteria** tab when you launch the integration. You can define the enumerations in Studio or refer to existing enumeration definitions in your tenant.

Steps

1. Open the assembly in the Assembly Editor.

©2026 Workday, Inc. All rights reserved

2. On the **Launch Parameters** tab of the **Properties** view for the **workday-in** transport, click **Add Enumeration Type Parameter**.
3. Click **Edit Type**.
4. As you configure the enumeration, consider:

Option	Description
Enumeration Reference	Enter the **Type Name** of an existing enumeration definition in your tenant.
Enumeration Definition	Enter a unique **Type Name** for the enumeration definition. Note: When you enter the **Type Name** of an existing enumeration definition, Workday overwrites the definition. Add the values that the enumeration allows.

Next Steps
Deploy your integration.
Related Information
Tasks
Deploy Integrations to Workday

Concept: Assembly Transports

Studio provides 2 categories of assembly transports on the **Palette**:

- **In Transports**
- **Out Transports**

In-transports receive messages from Workday or a local resource, and forward them for processing to other elements defined within the assembly. In-transports always indicate the start of a processing chain.

Out-transports exchange messages with a resource such as an HTTP endpoint, Workday application, or FTP site. Assemblies containing a **local-out** transport can communicate with subassemblies containing a **local-in** transport.

Studio supports these in-transport assembly elements on the **Palette**:

Name	Function	Notes
workday-in	Defines a starting point for executing an assembly.	• Listens for Workday application request messages. • Supports launch parameters and service definitions for your integration. • Synchronous (request-response).
local-in	Defines an entry point to an assembly or a subassembly.	• Exchanges messages with assemblies containing a **local-out** transport. • Supports parameters and out-parameters.

©2026 Workday, Inc. All rights reserved

Name	Function	Notes
		• Synchronous (request-response).

Studio supports these out-transport assembly elements on the **Palette**:

Name	Function	Notes
workday-out-rest	Sends Workday application REST request messages to an HTTP URL.	• Synchronous (request-response).
workday-out-soap	Sends Workday application SOAP request messages to an HTTP URL.	• Synchronous (request-response).
local-out	Exchanges messages with assemblies and subassemblies containing a **local-in** transport.	• Synchronous (request-response).

http-out	Sends HTTP request messages to an HTTP URL.	• Supports these HTTP child elements:
		  • **http-basic-auth**
		  • **http-custom-auth**
		  • **https-properties**
		• Synchronous (request-response).

ftp-out	Sends messages to an FTP server.	• Asynchronous (request only).
ftps-out	Sends messages securely to an FTP server using the FTP over SSL (FTPS) protocol.	• Asynchronous (request only).
sftp-out	Sends messages securely to an FTP server using the FTP over SSH (SFTP) protocol.	• Asynchronous (request only).
xmpp-out	Writes text messages to a Jabber server. Example: a Google Gmail server.	• Asynchronous (request only).
custom-out	Creates a customizable out-transport.	• Synchronicity defined by the Spring bean implementation.

©2026 Workday, Inc. All rights reserved

Name	Function	Notes

email-out	Writes e-mails to an SMTP (mail) server.	• Studio checks these properties for preceding and trailing whitespace:
		  • **To**
		  • **Subject**
		  • **From**
		  • **Bcc**
		  • **Cc**
		  • **Host**
		  • **Port**
		  • **User**
		  • **Password**
		  • **Reply To**
		• Asynchronous (request only).

Concept: Enumerations

Enumerations restrict an attribute value to a set of literal values, such as cost center codes. You can use enumerations to define a set of constants to use as literal values when you design integrations. This allows you to restrict the range of data your integration uses.

You can create and maintain enumeration definitions in Workday using these tasks:

- **Create Integration Enumeration Definition**
- **Edit Integration Enumeration Definition**
- **Delete Integration Enumeration Definition**
- **View Integration Enumeration Definition**

You can use enumerations in these elements:

- Launch parameters.
- Integration attributes.
- Integration maps.

You can assign enumerations:

- In Studio.
- Using the **Create Integration Generic Service** task in Workday.

You can assign enumerations in Studio integrations using 1 of these methods:

- Create an enumeration definition in Studio and deploy it to your tenant when you launch the Studio integration.
- Create an enumeration reference to an existing enumeration definition in your tenant.

You can use the same enumeration definition across multiple integrations by selecting 1 integration to provide the definition, and referring to it in the others.

Related Information
Tasks
Create Attribute Map Services

©2026 Workday, Inc. All rights reserved

Reference: Workday-In Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **workday-in** transport within the assembly.
Routes To	`routes-to`	The assembly element towards which the **workday-in** transport directs the message.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Integration System Name	`integration-system-name`	The name of your integration system.

Launch Parameters Tab

Launch Parameter	Description
Simple Type Parameter	Launch parameters that use string literal values.
Reference Type Parameter	Launch parameters that use references to Workday CRFs and web services.
Enumeration Type Parameter	Launch parameters that use Workday CRFs and enumeration definitions.

Services Tab

Integration Service	Description
Attribute Map Service	The integration attributes and integration maps used in your integration.
Delivery Service	Delivers documents to a target endpoint from your Workday tenant.

©2026 Workday, Inc. All rights reserved

Integration Service	Description
Listener Service	Enables your integration to receive launch messages from third-party clients through your Workday tenant.
Retrieval Service	Retrieves documents from remote clients and stores them in your Workday tenant.
Sequence Generator Service	Generates unique, sequenced filenames for documents each time you run your integration.
Transaction Log Service	Displays the logs for transaction events that your integration system has subscribed to in your Workday tenant.
Report Service	Specifies aliases for any custom reports that you refer to in your integration.
Custom Object Service	Specifies aliases for any custom objects that you refer to in your integration.
Service Reference	References any integration service configured on another **workday-in** transport within the same collection.

Reference: Local-In Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **local-in** transport within the assembly.
Routes To	`routes-to`	The assembly element towards which the **local-in** transport directs the message.
Access	`access`	Specifies whether your subassembly is **public** or **private**. The **Palette** displays only public subassemblies in the **Workspace Components** category.
Use Global Error Handlers	`use-global-error-handlers`	Determines whether errors that occur within the subassembly are handled at the subassembly level. The default is `false`, meaning that errors are handled by the assembly that invoked the subassembly.
Icon	`icon`	Specifies a custom icon for the subassembly. Supports 24x24 pixel PNG or GIF icons.
Tooltip	`tooltip`	A tooltip for your subassembly.

©2026 Workday, Inc. All rights reserved

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	**Store Message** doesn't apply to **local-in** transports. Studio ignores this setting for **local-in** transports.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.

Parameters Tab

The **local-in** transport's parameters specify any `MediationContext` properties you can set.

Property	XML Attribute Name	Description
Name	`name`	The name of your **local-in** parameter.
Documentation	`documentation`	A text description of your **local-in** parameter.
Required	`required`	Determines whether your **local-in** transport requires your parameter.
Type	`type`	Specifies your parameter type. The options are: • *string* • *integer* • *long* • *double* • *float* • *boolean*
Default	`default`	The default value of your parameter.
Validation	`validation`	A Boolean MVEL expression that must evaluate as `true`, otherwise Studio throws an error. Enables subassembly developers to impose constraints on parameter values.

Out Parameters Tab

The **local-in** transport's out-parameters specify any `MediationContext` properties that the subassembly retrieves.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Name	`name`	The name of your out-parameter.
Documentation	`documentation`	A text description of your out-parameter.

Reference: Workday-Out-Rest Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **workday-out-rest** transport within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **workday-out-rest** transport.
Execute When	`execute-when`	A condition that determines whether the **workday-out-rest** transport executes.
Extra Path	`extra-path`	Specifies the path in your Workday tenant through which Studio sends the REST request. Studio automatically sets up and appends the Workday service endpoint for this path. Example: the endpoint in the path below invokes a Workday *Business Site Directory* report, from a tenant named ACME, requested in JSON format: `systemreport2/ACME/Business_Site_Directory?format=json`

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis. You can globally change the implementation classes by

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		editing the WEB-INF/classes/spring/assembly-bean.xml file.
Method	`method`	Enables you to add a REST method to your **workday-out-rest** transport. Example: `put`, `get`, `post`.
Failure Message	`failure-message`	A custom message used to replace any exception message returned when you invoke a **workday-out-rest** endpoint.
Response Timeout	`response-timeout`	Specifies, in milliseconds, a timeout period for awaiting a response from the server. The default value is 21600000 ms (6 hours).

Reference: Workday-Out-Soap Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **workday-out-soap** transport within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **workday-out-soap** transport.
Execute When	`execute-when`	A condition that determines whether the **workday-out-soap** transport executes.
Application	`application`	The name of the Workday application to which the transport sends content. Studio uses the name that you enter in this field to create a URI that specifies the location of the Workday application.
Version	`version`	The web service version that the SOAP request calls. Note: When you specify a Web services version in an XSLT request, that version takes precedence.

©2026 Workday, Inc. All rights reserved

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Failure Message	`failure-message`	A custom message used to replace any exception message returned when you invoke a **workday-out-soap** endpoint.
Replace with Soap Fault	`replace-with-soap-fault`	Replaces a failed **workday-out-soap** message with a SOAP fault. The SOAP fault contains extracted `faultcode` and `faultstring` property values.
Response Timeout	`response-timeout`	Specifies, in milliseconds, a timeout period for awaiting a response from the server. The default value is 21600000 ms (6 hours).

Reference: Local-Out Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **local-out** transport within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **local-out** transport.
Execute When	`execute-when`	A condition that determines whether the **local-out** transport executes.
Endpoint	`endpoint`	The target endpoint address towards which the **local-out** transport directs the message. Endpoint addresses follow the format `application-name/component-ID`, where:

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• `application-name` is the name of the project containing the target assembly. • `component-ID` is the ID of the target component. Example: `<cc:local-out id="local" endpoint="myassembly/mediation1"/>`

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to create a copy of the request message processed by the **local-out** transport, so that any subassembly modifying the message doesn't affect the others.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Property	XML Attribute Name	Description
Name	`name`	The name of the parameter child element.
Value	`value`	The value of the parameter child element.

Reference: HTTP-Out Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **http-out** transport within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **http-out** transport. Note: The **http-out** transport populates the `MediationContext` property `http.response.status` with the standard HTTP response codes returned by the remote server. Examples: 202 (Accepted), 403 (Forbidden). The assembly can use the value in this property to determine handling for different responses.
Execute When	`execute-when`	A condition that determines whether the **http-out** transport executes.
Endpoint	`endpoint`	The target HTTP URL address towards which the **http-out** transport directs the request message. Example: `http://myhost.example.org/MyService`
Http Method	`http-method`	The HTTP request method supported by the **http-out** transport. Example: `DELETE`, `GET`, `POST`, `PUT`, and `PATCH`.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Note: To use the PATCH method, you must set the `wd.http.client` property to `apache`. If you don't want downstream transports to also use the Apache HTTP client, reset the `wd.http.client` property to `null`. If you don't specify a method for this property, Studio uses the POST method.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Connect Timeout	`connect-timeout`	Specifies, in milliseconds, a timeout period for establishing a connection with an HTTP server. If you don't specify a value, the session times out after 300000 ms (5 minutes).
Response Timeout	`response-timeout`	Specifies, in milliseconds, a timeout period for awaiting a response from an HTTP server. If you don't specify a value, the session time depends on your assembly version:

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• 600000ms (10 minutes) if the assembly is versioned 2025.37. • 21600000 ms (6 hours) if the assembly is versioned below 2025.37. If the HTTP connection remains unresponsive for longer than this period, the **http-out** transport fails.
Error As Response	`error-as-response`	Enables you to copy an error response message from a server to the message on the **http-out** transport.
Close Connection	`close-connection`	Adds a `Connection: close` header to an outgoing **http-out** transport message.
Streaming	`streaming`	Enables HTTP streaming using `PUT` and `POST` HTTP methods.
Auto Inproc	`auto-inproc`	Specifies whether to use in-process transport communication for **http-out** request messages. Enables you to send messages directly to assembly endpoints located within the same server without using the network I/O capabilities of the local host.
Retries	`retries`	Specifies the number of retry attempts that the **http-out** transport makes if a transport exception occurs.
Retry Delay	`retry-delay`	Specifies in milliseconds a delay period between each retry attempt. Note: Workday complies with the `Retry-After` or `RateLimit-Reset` header value in an external endpoint's 429 Too Many Requests response provided it's shorter than the default maximum of 5 minutes or the `retry-delay` value you specify here. If the response header value is missing, Workday uses the `retry-delay` value. If both the `retry-delay` and header values are missing,

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Workday uses the 5 minute default. An integration will be unable to determine a retry delay and will fail if the header value is larger than both the `retry-delay` value and the default maximum.
Log Retries	`log-retries`	Enables you to log HTTP retry attempts to the server log file.
Accept Gzip	`accept-gzip`	Specifies whether to include an HTTP `accept-encoding=gzip` field in the **http-out** request header, which indicates whether to accept gzip content in the response.
Gzip Content	`gzip-content`	Specifies whether to include an HTTP `content-encoding=gzip` field in the **http-out** request header, which indicates whether to gzip the message content in the outgoing request.
Accept	`accept`	Specifies the `accept` header value to send with an HTTP request, which defines 1 or more content types that are acceptable in the response from an HTTP server.

Note: For Amazon S3 transports, add a tag to your IAM user with a key of *workday-type* and a value of *integration*. This tag is case sensitive and doesn't ignore whitespace.

Reference: FTP-Out Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **ftp-out** component within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **ftp-out** transport.
Execute When	`execute-when`	A condition that determines whether the transport executes.
Endpoint	`endpoint`	Specifies the host (and optionally the port) and FTP directory to write files to.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Follows the format `ftp://FTP-directory`, where `FTP-directory` is the target location. Example: `ftp://ftpserver.example.org/outputDir` Note: The directory location you specify must already exist on the FTP server.
Binary File	`binary-file`	Specifies whether the output files are transferred as binary.
Input File Pattern	`input-file-pattern`	A filename pattern for matching files of a particular type. Example: specify `*.xls` to pick up Excel files only.
Input Content Type	`input-content-type`	The media type that the runtime includes for the `content-type` value in the message header. A media type comprises at least 2 parts: a type, a subtype, and 1 or more optional parameters. The `text` subtype has an optional `charset` parameter, which specifies the character encoding. Example: `text/html; charset=UTF-8` This property is particularly useful in the case of non-XML text file formats where the media type and charset aren't explicitly stated in the file.
Output File Pattern	`output-file-pattern`	Specifies how the names of output files are generated. The pattern can contain literal strings as well as one or more of these tokens: • `${INFILE}`, which is replaced with the name of the input file. • `${UUID}`, which is replaced with a universally unique identifier. • `${EXT}`, placed at the end of the pattern, which is replaced with a filename extension based

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		on the content type of the data that's written to the assembly component's output.
Method	`method`	Specifies the transfer method: • `put`: writes files to the FTP endpoint that contains the remote directory. • `list`: lists all files and directories at the FTP endpoint. Use the **Input File Pattern** property with `list` to apply a pattern to the listing being returned. You can access the results of the `list` method using the MVEL `ftp` object. Example: `ftp.list()` returns a list of strings that contains the last listing retrieved. • `get`: retrieves the first file found to match the **Input File Pattern** property value. • `delete`: deletes the first file found to match the **Input File Pattern** property value. Note: A file isn't automatically deleted from the FTP site after a `get` has been performed. If you wish to delete a file, you must invoke a separate method. Example: you can include a separate `ftp-out` transport specifically to delete the file.
Password	`password`	The password associated with the username used to access the FTP server.
Username	`username`	The username used to access the FTP server.

©2026 Workday, Inc. All rights reserved

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Passive Mode	`passive-mode`	Specifies whether the transport supports FTP in passive mode. In passive mode, the FTP client initiates both connections required for the FTP transfer. The client contacts the server on the command port and issues the PASV command. The server then replies, indicating which port it's listening to for the data connection. The client then initiates the data connection from its data port to the specified server data port. Finally, the server sends back an ACK to the client's data port. In active mode, the client initiates the command connection and the server initiates the data connection.
Timeout	`timeout`	Specifies in milliseconds a timeout period for FTP sessions between clients and the FTP server on which the transport is listening.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		If you don't specify a value, the session times out after 300000 ms (5 minutes).
Mime Types Map File	`mime-types-map-file`	The location of a map file for mapping between MIME types and file extensions. Use this property to: • Map MIME types to file extensions when the **Method** property is set to `put`. • Map file extensions to MIME types when the **Method** property is set to `get`. The file for this mapping is `ccx/conf/mime.types`. You can provide your own version of this file. This transport can generate filenames for its output files using a `${EXT}` token placed at the end of the `output-file-pattern` value.
Temp File Name	`temp-file-name`	Specifies a temporary filename. When the **Method** property is set to `put`, the file uploads using this temporary filename. After the FTP completes, the file is renamed to the name specified by the **Output File Pattern** property.
Close Connection	`close-connection`	Specifies whether disconnection occurs when the FTP action is complete.
Pool Size	`pool-size`	The maximum number of simultaneous connections and sessions that can be supported on a per-host basis for this component. If more than the specified number of threads attempt to connect to the same host, some are forced to wait for access to the pool. To make the pool size unlimited, specify a negative number.
Debug	`debug`	Enables JSCAPE debug mode in the transport.

©2026 Workday, Inc. All rights reserved

Reference: FTPS-Out Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **ftps-out** component within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **ftps-out** transport.
Execute When	`execute-when`	A condition that determines whether the transport executes.
Endpoint	`endpoint`	Specifies the host (and optionally the port) and FTP directory to write files to. Follows the format `ftps://FTPS-directory`, where `FTPS-directory` is the target location. Example: `ftps://ftpsserver.example.org/outputDir` Note: The directory location you specify must already exist on the server.
Binary File	`binary-file`	Specifies whether the output files are transferred as binary objects.
Explicit SSL	`explicit-ssl`	Specifies connection type: explicit SSL (AUTH TLS) or implicit SSL (AUTH SSL). Set to `true` to use explicit SLL. FTP clients can choose to continue the communication unencrypted as a normal FTP session or switch to FTPS mode by issuing the AUTH command.
Input File Pattern	`input-file-pattern`	A filename pattern for matching files of a particular type. Example: specify `*.xls` to pick up Excel files only.
Input Content Type	`input-content-type`	The media type that the runtime includes for the `content-type` value in the message header. A media type comprises at least 2 parts: a type, a subtype, and 1 or more optional parameters. The `text` subtype has an

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		optional `charset` parameter, which specifies the character encoding. Example: `text/html; charset=UTF-8` This property is particularly useful in the case of non-XML text file formats where the media type and charset aren't explicitly stated in the file.
Output File Pattern	`output-file-pattern`	Specifies how the names of output files are generated. The pattern can contain literal strings as well as one or more of these tokens: • `${INFILE}`, which is replaced with the name of the input file. • `${UUID}`, which is replaced with a universally unique identifier. • `${EXT}`, placed at the end of the pattern, which is replaced with a filename extension based on the content type of the data that's written to the assembly component's output.
Method	`method`	Specifies the transfer method: • `put`: writes files to the FTP endpoint that contains the remote directory. • `list`: lists all files and directories at the FTP endpoint. Use the **Input File Pattern** property with `list` to apply a pattern to the listing being returned. You can access the results of the `list` method using the MVEL `ftp` object. Example: `ftp.list()` returns a list of strings that contains the last listing retrieved. • `get`: retrieves the first file found to match the **Input**

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		File Pattern property value. • `delete`: deletes the first file found to match the **Input File Pattern** property value. Note: A file isn't automatically deleted from the FTP site after a `get` has been performed. If you wish to delete a file, you must invoke a separate method. Example: you can include a separate `ftp-out` transport specifically to delete the file.
Password	`password`	The password associated with the username used to access the FTP server.
Username	`username`	The username used to access the FTP server.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Passive Mode	`passive-mode`	Specifies whether the transport supports FTP in passive mode.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		In passive mode, the FTP client initiates both connections required for the FTP transfer. The client contacts the server on the command port and issues the PASV command. The server then replies, indicating which port it's listening to for the data connection. The client then initiates the data connection from its data port to the specified server data port. Finally, the server sends back an ACK to the client's data port. In active mode, the client initiates the command connection and the server initiates the data connection.
Timeout	`timeout`	Specifies in milliseconds a timeout period for FTP sessions between clients and the FTP server on which the transport is listening. If you don't specify a value, the session times out after 300000 ms (5 minutes).
Mime Types Map File	`mime-types-map-file`	Specifies the location of a map file for mapping between MIME types and file extensions. Use this property to: • Map MIME types to file extensions when the **Method** property is set to `put`. • Map file extensions to MIME types when the **Method** property is set to `get`. The file for this mapping is `ccx/conf/mime.types`. You can provide your own version of this file. This transport can generate filenames for its output files using a `${EXT}` token placed at the end of the `output-file-pattern` value.
Temp File Name	`temp-file-name`	Specifies a temporary filename. When the **Method** property is set to `put`, the file uploads using this temporary filename. After

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		the FTP completes, the file is renamed to the name specified by the **Output File Pattern** property.
Close Connection	`close-connection`	Specifies whether disconnection occurs when the FTP action is complete.
Pool Size	`pool-size`	The maximum number of simultaneous connections and sessions that can be supported on a per-host basis for this component. If more than the specified number of threads attempt to connect to the same host, some are forced to wait for access to the pool. To make the pool size unlimited, specify a negative number.
Debug	`debug`	Enables JSCAPE debug mode in the transport.

Reference: SFTP-Out Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **sftp-out** component within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **sftp-out** transport.
Execute When	`execute-when`	Specifies a condition that determines whether the transport executes.
Endpoint	`endpoint`	Specifies the host (and optionally the port) and FTP directory to write files to. Follows the format `sftp://SFTP-directory`, where `SFTP-directory` is the target location. Example: `sftp://sftpserver.example.org/outputDir` Note: The directory location you specify must already exist on the server.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Dual Authentication	`dual-authentication`	Specifies whether to enforce dual authentication with both a username/password combination and a private key. The default value is `false`, which means that a user can authenticate with either a private key or a username/password combination, but not both. If the property is set to `false` and a user enters a private key (by configuring the private key properties of the **sftp-out** component), then the authentication automatically uses that key value and the username, ignoring the password field. If the property is set to `false` and no private key is configured then the authentication uses the username/password combination. If the property is set to `true` then the authentication uses both the private key value and username/password combination.
Host Key Fingerprint	`host-key-fingerprint`	The encryption key that the FTP server will use for SSH communications.
Username	`username`	The username used to access the FTP server.
Password	`password`	The password associated with the username used to access the FTP server.
Binary File	`binary-file`	Specifies whether the output files are transferred as binary.
Explicit SSL	`explicit-ssl`	Connection type: explicit SSL (AUTH TLS) or implicit SSL (AUTH SSL). Set to `true` to use explicit SSL.
Input File Pattern	`input-file-pattern`	A filename pattern for matching files of a particular type. Example: specify `*.xls` to pick up Excel files only.
Input Content Type	`input-content-type`	The media type that the runtime includes for the `content-type` value in the message header. A

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		media type comprises at least 2 parts: a type, a subtype, and 1 or more optional parameters. The `text` subtype has an optional `charset` parameter, which specifies the character encoding. Example: `text/html; charset=UTF-8` This property is particularly useful in the case of non-XML text file formats where the media type and charset aren't explicitly stated in the file.
Output File Pattern	`output-file-pattern`	Specifies how the names of output files are generated. The pattern can contain literal strings as well as one or more of these tokens: • `${INFILE}`, which is replaced with the name of the input file. • `${UUID}`, which is replaced with a universally unique identifier. • `${EXT}`, placed at the end of the pattern, which is replaced with a filename extension based on the content type of the data that's written to the assembly component's output.
Method	`method`	Specifies the transfer method: • `put`: writes files to the FTP endpoint that contains the remote directory. • `list`: lists all files at the FTP endpoint. Use the **Input File Pattern** property with `list` to apply a pattern to the listing being returned. You can access the results of the `list` method using the MVEL `ftp` object. Example: `ftp.list()` returns a list of strings

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		that contains the last listing retrieved. • `get`: retrieves the first file found to match the **Input File Pattern** property value. • `delete`: deletes the first file found to match the **Input File Pattern** property value. Note: A file isn't automatically deleted from the FTP site after a `get` has been performed. If you wish to delete a file, you must invoke a separate method. Example: you can include a separate `ftp-out` transport specifically to delete the file.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Timeout	`timeout`	Specifies in milliseconds a timeout period for FTP sessions between clients and the FTP

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		server on which the transport is listening. If you don't specify a value, the session times out after 300000 ms (5 minutes).
Mime Types Map File	`mime-types-map-file`	Specifies the location of a map file for mapping between MIME types and file extensions. Use this property to: • Map MIME types to file extensions when the **Method** property is set to `put`. • Map file extensions to MIME types when the **Method** property is set to `get`. The file for this mapping is `ccx/conf/mime.types`. You can provide your own version of this file. This transport can generate filenames for its output files using a `${EXT}` token placed at the end of the `output-file-pattern` value.
Temp File Name	`temp-file-name`	Specifies a temporary filename. When the **Method** property is set to `put`, the file uploads using this temporary filename. After the FTP completes, the file is renamed to the name specified by the **Output File Pattern** property.
Close Connection	`close-connection`	Specifies whether disconnection occurs when the FTP action is complete.
Pool Size	`pool-size`	The maximum number of simultaneous connections and sessions that can be supported on a per-host basis for this component. If more than the specified number of threads attempt to connect to the same host, some are forced to wait for access to the pool. To make the pool size unlimited, specify a negative number.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Debug	`debug`	Enables JSCAPE debug mode in the transport.

Reference: XMPP-Out Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **xmpp-out** component within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **ftp-out** transport.
Execute When	`execute-when`	A condition that determines whether the transport executes.
Endpoint	`endpoint`	Specifies the recipient of the Instant Message. Example: `xmpp:logan.mcneil@gmail.com`
Domain	`domain`	The domain where the username is registered. Example: `gmail.com`
Server	`server`	The host or IP address of the XMPP/Jabber server.
Port	`port`	The port number at which the XMPP transport listens for messages. The port used by most XMPP servers is 5222.
Username	`username`	The username used to access the FTP server.
Password	`password`	The password associated with the username used to access the FTP server.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.

Reference: Custom-Out Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **custom-out** component within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **custom-out** transport.
Execute When	`execute-when`	A condition that determines whether the transport executes.
Endpoint	`endpoint`	The endpoint to which the **custom-out** transport will send messages. Used by the **custom-out** Spring bean implementation.
Input	`input`	Specifies where the assembly component obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the assembly component directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Spring Bean	`spring-bean`	Specifies the Spring bean that implements the transport's interface class.
Method Name	`method-name`	The name of the Java method in the custom Java class.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.

Reference: Email-Out Transport Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **email-out** component within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **email-out** transport.
Execute When	`execute-when`	A condition that determines whether the transport executes.
To	`to`	The primary email address of the intended recipient. Follows the format `mailto:[emailaddress]`. If you want to use only BCC addresses, specify an empty `mailto:` value.
Subject	`subject`	The subject line of the email.
From	`from`	The email address that should display in the **From** field of the delivered email.
Ssl	`ssl`	Specifies whether the connection is SSL.
Starttls	`starttls`	Specifies whether to activate an immediate SSL negotiation

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		with the remote SMTP server by issuing a STARTTLS command before starting the SMTP session. The SMTP server responds with one of these reply codes: • 220 Ready to start. • TLS 501 Syntax error (no parameters allowed). • 454 TLS not available due to a temporary reason. Note: The **Ssl** and **starttls** properties are mutually exclusive.
Host	`host`	The name or IP address of the SMTP mail server that the transport uses to deliver the message. If you're using OAuth, set this value to `localhost`.
Port	`port`	The port number at which the mail server is running. The default is 25, the standard SMTP port. Exceptions: • If the **ssl** property is set to `true` and no port is specified, the default is 465. • If the **Starttls** property is set to `true` and no port is specified, the default is 587. Note:
User		The user name for accessing the email server.
Password	`password`	The password for the username specified in the **User** property.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Bcc	`bcc`	Specifies blind carbon-copy email recipients, using a semicolon separator. Example: `logan.mcneil@workday.com; jake.lee@workday.com`.
Cc	`cc`	Specifies carbon-copy email recipients, using a semicolon separator. Example: `logan.mcneil@workday.com; jake.lee@workday.com`.
Reply To	`reply-to`	The email address that's automatically inserted into the **To** field when a user replies to an email message. For most email messages, the address in the **From** field is sufficient, so there's no need to define the **Reply To** property. Specify multiple emails addresses using a semicolon separator. Example: `logan.mcneil@workday.com; jake.lee@workday.com`.
Timeout	`timeout`	Specifies, in milliseconds, a timeout period for FTP sessions between clients and the FTP server on which the transport is listening. If you don't specify a value, the session times out after 300000 ms (5 minutes).
Transient Exceptions	`transient-exceptions`	Transient exceptions are exceptions that could succeed without any modifications on retry. Nontransient exceptions

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		are those that will continue to fail on retry until the underlying cause of the problem is corrected. By default, Workday treats only `java.net.ConnectException` and `java.net.SocketException` errors as transient exceptions. Specify additional transient exceptions as a comma, space, or comma and space-separated enumeration of class names. Examples: • `java.net.UknownHostException java.net.ConnectException java.net.SocketException` • `java.net.UknownHostException,j` • `java.net.UknownHostException, java.net.ConnectException, java.net.SocketException` • `@{new java.net.UnknownHostException(` Note: If you specify additional transient exceptions, you must also include the default exceptions `java.net.ConnectException` and `java.net.SocketException` in the list. If you don't, Workday overwrites them.

SMTP Headers Tab

Property	XML Attribute Name	Description
Name	`name`	The name of a custom SMTP header. To add a new custom header below any existing rows, click **Add**, then supply a **Name** and **Value**. To add a new header above an existing row, place the cursor in that row, then click **Insert** on the **Add** drop-down menu. Use the buttons to the right of the **SMTP Headers** tab to delete rows or reorder them.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Value	`value`	The value of a custom SMTP header.

OAuth Tab

To use Microsoft Outlook OAuth authentication with Studio's Email-Out component, register Studio as an app with Microsoft Azure, provide domain-wide delegation, grant administrator consent for `Mail.Send` permissions, and obtain various values. For details, see https://learn.microsoft.com/en-gb/azure/active-directory/develop/v2-oauth2-client-creds-grant-flow.

To use Google Gmail OAuth authentication with Studio's Email-Out component, register a Service Account with Google Cloud, provide domain-wide delegation, and obtain a JSON description of various values. For details, see https://support.google.com/cloud/answer/6158849?hl=en.

Property	XML Attribute Name	Description
Provider	`provider`	The name of the email provider, either `MICROSOFT_OUTLOOK` or `GOOGLE_GMAIL`.
Name	`name`	The Workday-mandated names you associate with various values obtained from the Microsoft Azure Portal or Google Cloud Console. For Microsoft Outlook, enter these names: • outlook.client.id • outlook.client.secret • outlook.tenant.id • outlook.user.id For Google Gmail, enter: • gmail.service.account.key.json • gmail.delegated.user
Value	`value`	The values obtained from Microsoft or Google that you associate with the names above. For Microsoft Outlook, create these name/value pairs: Name: outlook.client.id — Value: From a registered app's **Overview** page on the Microsoft Azure Portal: Application (client) ID.

©2026 Workday, Inc. All rights reserved

<table>
<tr><th>Property</th><th>XML Attribute Name</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Value</th></tr>
<tr><td>outlook.client.secret</td><td>From a registered app's Certificates & secrets page on the Microsoft Azure Portal:

Client Secret ID.</td></tr>
<tr><td>outlook.tenant.id</td><td>From a registered app's Overview page on the Microsoft Azure Portal:

Directory (tenant) ID.</td></tr>
<tr><td>outlook.user.id</td><td>The Outlook address of the account from which the email will be sent. Also entered in the From field on the Common tab.</td></tr>
</table>

For Google Gmail, create these name/value pairs:
<table>
<tr><th>Name</th><th>Value</th></tr>
<tr><td>gmail.service.account.key.json</td><td>The JSON text from the file downloaded from the KEYS section of the Service accounts page on the Google Cloud Console.</td></tr>
</table>
</td></tr>
</table>

©2026 Workday, Inc. All rights reserved

<table>
<tr><th>Property</th><th>XML Attribute Name</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Name</th><th>Value</th></tr>
<tr><td>gmail.delegated.user</td><td>The Gmail address from which the email will be sent. Must be a Principal associated with the Google Service Account. Also entered in the From field on the Common tab.</td></tr>
</table>
</td></tr>
</table>

Gmail over SSL or with STARTTLS

When configuring the **email-out** transport to use Gmail over SSL or STARTTLS, ensure that:

- **user** and **password** contain valid Gmail credentials.
- **endpoint** is `mailto:[name]@gmail.com`.
- **host** is `smtp.gmail.com`.
- **ssl** or **starttls** is set to `true`.
- **port** is either:
  - 465, if **ssl** is set to `true`.
  - 587, if **starttls** is set to `true`.
- **timeout** is `30000` (30 seconds).

Microsoft Office 365 over SSL or with STARTTLS

When configuring the **email-out** transport to use Microsoft Office 365 over SSL or STARTTLS, ensure that:

- **user** and **password** contain valid Office 365 user credentials.
- **endpoint** is `mailto:[name]@[domain-name.top-level-domain]`.
- **host** is `smtp.office365.com`.
- **ssl** or **starttls** is set to `true`.
- **port** either:
  - 465, if **ssl** is set to `true`.
  - 587, if **starttls** is set to **true**.
- **timeout** is `30000` (30 seconds).

Configuring the Message Body and Attachment for Nonbinary Formats

When receiving incoming messages, many email applications match the first text part in the MIME message and use it as the message text. To set the MIME type of the root message part to `text/plain`, set the message **Input** and **Output** properties to *message*, then set the **Output** MIME Type to *text/plain*.

Use the **Message Builder** to define the message content for the email message body using a **text** element.

©2026 Workday, Inc. All rights reserved

For the attachment, set the **Output** property to *attachment*, the **Output** attachment index to `0`, and the **Output** MIME Type to *text/plain*.

Again, use the **Message Builder** to define the message content for the attachment using a **text** element.

To prevent some email applications displaying text attachments in the message body, set the attachment MIME type to *application/octet-stream*, even though it's actually text. To do so, use an MVEL expression within an **eval** step. Example: `parts[1].setMimeType('application/octet-stream')`

To set the attachment file name, use an MVEL expression to set its `Content-Disposition` header. Example: `parts[1].setHeader('Content-Disposition', 'attachment; filename=payload.txt')`.

Finally, configure the **email-out** step.

Configuring the Message Body and Attachment for Binary Formats
To inject a binary file into the **MediationContext**, define a Retrieval service for the **workday-in** that retrieves the file from an FTP or SFTP transport.

Create the message body, then use an **eval** step to copy the retrieved file to a variable. You can use the `da` document accessor variable and the `toVar` method to achieve this. Example: `da.toVar(Rising_2011_Logo.jpg', 'wd.retrieve.variable')`.

Use a **copy** step to copy the file from the `wd.retrieve.variable` to an attachment. To do so, set the **Input** property to *variable* and specify `wd.retrieve.variable` as the input variable. Set the **Output** property to *attachment*, specify an **Output** attachment index of `0`, and an **Output** MIME Type of *image/jpeg*.

Finally, configure the **email-out** step.

HTTP-Out Transport Child Elements

Concept: HTTP-Out Transport Child Elements

The **http-out** transport supports these child elements:

Child Element	Description
http-basic-auth	Supports basic authentication on your **http-out** transport.
http-custom-auth	Supports custom authentication on your **http-out** transport.
https-properties	Supports HTTPS properties on your **http-out** transport.

Reference: HTTP-Basic-Auth Properties

Common Tab

Property	XML Attribute Name	Description
Password	`password`	A password for authentication.
Username	`username`	A username for authentication.
Realm	`realm`	The realm name used by a callback handler bean.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Callback-Handler Bean	`callback-handler-bean`	A bean that implements `javax.security.auth.callback.Callba` and handles the callback type `com.capeclear.transport.impl.http.a` to supply username and password details for authentication. If you don't specify password details for an **http-basic-auth** child element, you'll need to use a callback handler bean instead. Example: `<http-basic-auth username="me" callback-handler-bean="myCallBackHandlerBean"/>`. If you don't specify username and password details for an **http-basic-auth** child element, you'll need to use a callback handler bean instead. Example: `<http-basic-auth callback-handler-bean="myCallBackHandlerBean"/>`. If you specify username and password details for an **http-basic-auth** child element, you don't require a callback handler bean. Example: `<http-basic-auth username="me" password="mypassword"/>`.

Reference: HTTP-Custom-Auth Properties

Common Tab

Property	XML Attribute Name	Description
Password	`password`	A password for authentication.
Username	`username`	A username for authentication.
Realm	`realm`	The realm name used by a callback handler bean.
Callback-Handler Bean	`callback-handler-bean`	A bean that implements `javax.security.auth.callback.Callba` and handles the callback type `com.capeclear.transport.impl.http.a` to supply username and password details for authentication.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		If you don't specify password details for an **http-custom-auth** child element, you'll need to use a callback handler bean instead. Example: `<http-custom-auth username="me" callback-handler-bean="myCallBackHandlerBean"/>`. If you don't specify username and password details for an **http-custom-auth** child element, you'll need to use a callback handler bean instead. Example: `<http-custom-auth callback-handler-bean="myCallBackHandlerBean"/>`. If you specify username and password details for an **http-custom-auth** child element, you don't require a callback handler bean. Example: `<http-custom-auth username="me" password="mypassword"/>`.
Authenticator Bean	`authenticator-bean`	The spring bean that performs a custom authentication.

Reference: HTTPS Properties

Common Tab

Property	XML Attribute Name	Description
Accept Hosts	`accept-hosts`	The Java regular expression string to use for verifying host names specified in SSL certificates.
Key Password	`key-password`	The password to use when accessing the appropriate key within the keystore.
Keystore File	`key-file`	The location of the keystore file to use when connecting to an HTTPS endpoint.
Keystore Password	`keystore-password`	The password to use when accessing the keystore.
Proxy Host	`proxy-host`	The host name of a proxy server that you use to proxy HTTPS requests for this transport.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Proxy Port	`proxy-port`	The port of a proxy server that you use to proxy HTTPS requests for this transport.
Truststore File	`truststore-file`	The location of the truststore file to use when connecting to an HTTPS endpoint.
Truststore Password	`truststore-password`	The password to use when accessing the truststore.

FTPS-Out Transport Child Elements

Concept: FTPS-Out Transport Child Elements

The **ftps-out** transport supports these child elements:

Child Element	Description	Notes
proxy-properties	Proxy authentication properties.	
firewall-properties	Firewall connection properties.	• The **ftps-out** transport is firewall-friendly by default because the connection is passive. However, some firewalls may request that you route all connections through them.
client-key-properties	Client-key authentication properties.	
server-key-properties	Server-key authentication properties.	

Reference: Proxy-Properties Properties

Common Tab

Property	XML Attribute Name	Description
Proxy Host	`proxy-host`	The host name of the proxy server to use when making a connection to the secure FTP site.
Proxy Password	`proxy-password`	The password for the proxy server to use when making a connection to the secure FTP site.
Proxy Port	`proxy-port`	The proxy server port number to use when making a connection to the secure FTP site.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Proxy Type	`proxy-type`	The proxy type of the proxy server to use when making a connection to the secure FTP site. The default type is `http`.
Proxy Username	`proxy-username`	The proxy server username to use when making a connection to the secure FTP site.

Reference: Firewall-Properties Properties

Common Tab

Property	XML Attribute Name	Description
Firewall Host	`firewall-host`	The host name or IP address of the firewall to use when making a connection to the secure FTP site.
Firewall Port	`firewall-port`	The firewall's port number.

Reference: Client-Key Properties

Common Tab

Property	XML Attribute Name	Description
Passphrase	`passphrase`	The private key passphrase to use when authenticating with the secure FTP site.
Private Key	`private-key`	The private key associated with the username for SFTP authentication. You can specify it in 2 ways: • As a URI using the attachment protocol and the Workday ID. Example, `attachment:privatekey/@{WID}`, where `WID` is a 32-character Workday ID. • As a relative path location. Example: `private_key.txt`, where private_key.txt is a file located in the ws/WSAR-INF folder.

©2026 Workday, Inc. All rights reserved

Reference: Server-Key Properties

Common Tab

Property	XML Attribute Name	Description
Passphrase	`passphrase`	The private key passphrase to use when authenticating with the secure FTP site.
Private Key	`private-key`	The private key associated with the username for SFTP authentication. You can specify it in 2 ways: • As a URI using the attachment protocol and the Workday ID. Example, `attachment:privatekey/@{WID}`, where `WID` is a 32-character Workday ID. • As a relative path location. Example: `private_key.txt`, where private_key.txt is a file located in the ws/WSAR-INF folder.

SFTP-Out Child Elements

Concept: SFTP-Out Child Elements

The **sftp-out** transport supports these child elements:

Child Element	Description	Notes
proxy-properties	Proxy authentication properties.	
private-key-properties	Private key authentication properties.	

Reference: Proxy-Properties Properties

Common Tab

Property	XML Attribute Name	Description
Proxy Host	`proxy-host`	The host name of the proxy server to use when making a connection to the secure FTP site.
Proxy Password	`proxy-password`	The password for the proxy server to use when making a connection to the secure FTP site.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Proxy Port	`proxy-port`	The proxy server port number to use when making a connection to the secure FTP site.
Proxy Type	`proxy-type`	The proxy type of the proxy server to use when making a connection to the secure FTP site. The default type is `http`.
Proxy Username	`proxy-username`	The proxy server username to use when making a connection to the secure FTP site.

Reference: Private-Key Properties

Common Tab

Property	XML Attribute Name	Description
Passphrase	`passphrase`	The private key passphrase to use when authenticating with the secure FTP site.
Private Key	`private-key`	The private key associated with the username for SFTP authentication. You can specify it as a relative path location. Example: `private_key.txt`, where the private key file is located in the `ws/WSAR-INF` folder.

Components

Concept: Components

Components enable you to manipulate and transform messages as they pass through your integration. Workday Studio provides support for these components on the **Components** section of the Assembly **Palette**:

Mediation Component	Description	Notes
async-mediation	Supports 1 processing chain, which you can populate with steps to define a request message in your assembly.	
sync-mediation	Supports 2 processing chains, which you can populate with steps to define request messages and request-response messages in your assembly.	

©2026 Workday, Inc. All rights reserved

Mediation Component	Description	Notes
custom-mediation	Creates a customizable component that supports both request messages and request-response messages.	
route	Implements routing strategies and subroutes to direct the flow of messages through your assembly.	
splitter	Implements splitter strategies and subroutes to direct the flow of messages through your assembly.	Signals to downstream **aggregator** components.
aggregator	Concatenates messages into batches that you can direct towards a configured destination.	Detects upstream **splitter** components.

Reference: Async-Mediation Component Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **async-mediation** component within the assembly.
Routes To	`routes-to`	The assembly element towards which the **async-mediation** directs the message.
Execute When	`execute-when`	A condition that determines if the **async-mediation** component executes.

Advanced Tab

Property	XML Attribute Name	Description
Continue After Error	`continue-after-error`	Indicates how Workday behaves when an error handler clears an error. The default value is **rewind**, which means that processing continues from the previous component in the assembly. The **recover** setting specifies that Workday skips to the next step in the mediation when an error occurs.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Handle Downstream Errors	`handle-downstream-errors`	Indicates whether to handle downstream errors. If set to **false**, local error handlers only address errors that occur within the **async-mediation** component. Global error handlers will address downstream errors instead. If set to **true**, local error handlers for the **async-mediation** component can handle downstream errors.

Reference: Sync-Mediation Component Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **sync-mediation** component within the assembly.
Routes To	`routes-to`	The assembly element towards which the **sync-mediation** directs the message.
Routes Response To	`routes-response-to`	The destination of the response message for the **sync-mediation** component.
Execute When	`execute-when`	A condition that determines whether the **sync-mediation** component executes.

Advanced Tab

Property	XML Attribute Name	Description
Continue After Error	`continue-after-error`	Indicates how Workday behaves when an error handler clears an error. The default value is **rewind**, which means that processing continues from the previous component in the assembly. The **recover** setting specifies that Workday skips to the next step in the mediation when an error occurs.
Handle Downstream Errors	`handle-downstream-errors`	Indicates whether to handle downstream errors.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		If set to **false**, local error handlers only address errors that occur within the **sync-mediation** component. Global error handlers will address downstream errors instead. If set to **true**, local error handlers for the **sync-mediation** component can handle downstream errors.

Reference: Custom-Mediation Component Properties

Common Tab

Property	XML Attribute Name	Description
Id		The unique ID of the **custom-mediation** component within the assembly.
Routes To	`routes-to`	The assembly element towards which the **custom-mediation** directs the message.
Routes Response To	`routes-response-to`	The destination of the response message for the **custom-mediation** component.
Execute When	`execute-when`	A condition that determines whether the **custom-mediation** component executes.
Spring Bean	`ref`	The custom Spring bean Java class implemented by the **custom-mediation**.
Request Method	`request-method`	The name of the method used for request processing.
Response Method	`response-method`	The name of the method used for response processing.

Reference: Route Component Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **route** component within the assembly.

Reference: Splitter Component Properties

Common Tab

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **splitter** component within the assembly.
No Split Message Error	`no-split-error-message`	Specifies whether Workday raises an error when the **splitter** component doesn't generate a split message for a bulk input message.
Split Until	`split-until`	Specifies when the splitter component ceases processing a bulk input message.

Reference: Aggregator Component Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **aggregator** component within the assembly.
Routes To	`routes-to`	The assembly element towards which the **aggregator** directs the message.
Collate When	`collate-when`	An MVEL expression that determines whether incoming messages get collated.
Force Batch On Last Message	`force-batch-on-last-message`	Forces batching once the **splitter** component receives the last message.
Force Batch When	`force-batch-when`	An MVEL expression that determines whether incoming messages get batched.

Advanced Tab

Property	XML Attribute Name	Description
Allow External Control	`allow-external-control`	Enables upstream components to control when the **aggregator** component collates, batches, or both.

Route Component Child Elements

Concept: Route Component Child Elements

The **route** component supports these child elements.

©2026 Workday, Inc. All rights reserved

Child Element	Description	Notes
custom-strategy	A customizable routing strategy that you configure as a Spring bean.	
regex-strategy	Implements regular expressions that direct messages towards subroutes.	Implements **choose-route** child elements to indicate your target **sub-route**.
mvel-strategy	Implements MVEL expressions that direct messages towards subroutes.	Implements **choose-route** child elements to indicate your target **sub-route**.
round-robin-strategy	Directs messages towards subroutes in cyclical order.	
xpath-strategy	Implements XPath expressions that direct messages towards subroutes.	Implements **choose-route** child elements to indicate your target **sub-route**.
failover-strategy	Implements a failover mechanism that directs messages towards subroutes. If a **sub-route** child element fails, the next **sub-route** replaces it.	
all-strategy	Directs messages to each subroute in turn.	
doc-iterator	Iterates over documents that your assembly retrieves from Workday.	
loop-strategy	Implements a looping mechanism that direct messages towards subroutes.	
choose-route	Indicates the target **sub-route** child element towards which your strategy directs the message.	Supports XPath, MVEL, or regular expressions.
sub-route	Receives messages from your routing strategy, and directs them towards other components and transports in your assembly.	

Reference: Custom-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Spring Bean	`ref`	The Spring bean that implements your **custom-strategy** interface class.
Repeatable	`repeatable`	Enables looping.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		If set to **true**, your assembly implements a loop that repeatedly calls your **custom-strategy**. If set to **false**, your assembly doesn't implement a loop for your **custom-strategy**.
Repeat Limit	`repeat-limit`	Limits the number of times a loop calls your **custom-strategy**.
Handle Error	`handle-error`	Indicates whether your parent **route** component handles exceptions for your **sub-route**.

Reference: Regex-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Filter	`filter`	Indicates whether Studio displays an error message when you implement invalid regular expressions on a **choose-route** child element.

Reference: MVEL-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Filter	`filter`	Indicates whether Studio displays an error message when you implement invalid MVEL expressions on a **choose-route** child element.

Reference: Round-Robin-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Clone Request	`clone-request`	Indicates whether the **round-robin-strategy** sends a copy of the original mediation message to each of your **sub-route** child elements.
Failover	`failover`	Indicates whether the **round-robin-strategy** loops through

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		your **sub-route** child elements when it encounters a failed **sub-route**.
Timeout	`timeout`	A millisecond value indicating how long a failed subroute is excluded from routing.

Reference: XPath-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Filter	`filter`	Indicates whether Studio displays an error message when you implement invalid XPath expressions on a **choose-route** child element.
Namespaces	`namespaces`	The namespaces used by your XPath expression in the format `prefix namespace prefix2 namespace2`. Overrides namespace mappings in the assembly.xml file.
XPath Version	`xpath-version`	Specifies whether to use XPath version 1.0, 2.0, or 3.0. Enter the value 1 for assembly versions through Workday 11. Enter 2 or 3 for assembly versions Workday 12 onwards. You can enable Workday to set the version dynamically using an MVEL template that inserts a system property value. Example: `@{props.get('the_version')}`.

Reference: Failover-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Clear Fault	`clear-fault`	Indicates whether the **failover-strategy** clears exceptions generated by your **sub-route** child elements.
Clone Request	`clone-request`	Indicates whether the **failover-strategy** sends a copy of the original mediation message to

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		each of your **sub-route** child elements.

Reference: All-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Clone Request	`clone-request`	Indicates whether the **all-strategy** sends a copy of the original mediation message to each of your **sub-route** child elements.
Ignore Errors	`ignore-errors`	Indicates whether the **all-strategy** ignores errors generated by your **sub-route** child elements.

Reference: Doc-Iterator Properties

Common Tab

Property	XML Attribute Name	Description
Variable Name	`variable-name`	The name of the variable in which the **doc-iterator** stores your retrieved document. The default value is `wd.retrieve.variable`.
Labels	`labels`	Specifies 1 or more labels attached to your retrieved document.
Sort By	`sort-by`	The sort order for retrieving and presenting documents. Options are: • FILENAME_ASCENDING • FILENAME_DESCENDING • NONE

Reference: Loop Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Condition	`condition`	An MVEL expression that generates a Boolean value. The **loop-strategy** runs until the expression evaluates as false.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Increment	`increment`	An increment value for every loop iteration.
Init	`init`	The starting value of your loop.
Repeat Limit	`repeat-limit`	The maximum number of loops implemented by the **loop-strategy**. A route strategy can repeat up to 200,000 times in total. If the same strategy is called multiple times, the loop counter doesn't reset. It continues from the previous value. You can use a custom Java bean or a custom splitter to avoid this limit.

Reference: Choose-Route Properties

Common Tab

Property	XML Attribute Name	Description
Expression	`expression`	An expression used to match incoming message values.
Route	`route`	A target **sub-route** child element for matched expression values.

Reference: Sub-Route Properties

Common Tab

Property	XML Attribute Name	Description
Name	`name`	The unique name of the **sub-route** within the assembly.
Routes To	`routes-to`	The assembly element towards which the **sub-route** directs the message.

Splitter Component Child Elements

Concept: Splitter Component Child Elements

The **splitter** component supports these child elements.

©2026 Workday, Inc. All rights reserved

Child Element	Description	Notes
custom-splitter	A customizable splitter strategy that you configure as a Spring bean.	
standard-splitter	Extracts data from bulk messages, splits the data into tokens, and directs the tokenized data towards your target subroutes.	Supports these child elements: • **header-ends-with** • **header-fixed-lines** • **content-fixed-lines** • **content-starts-ends** • **content-starts-with** • **footer-fixed-lines** • **footer-starts-with**
xpath-splitter	Implements an XPath expression to extract data from incoming bulk messages.	
xml-stream-splitter	Implements an XPath expression to extract data from incoming bulk messages. Supports XML streaming.	
json-splitter	Extracts JSON from bulk input messages.	
unzip-splitter	Extracts data from ZIP and TAR files, splits the data into individual messages, and directs the messages towards their target subroutes.	
mtable-splitter	Extracts data from an `mtable` object, and splits the data into individual rows.	
sub-route	Receives messages from your splitter strategy, and directs them towards other components and transports in your assembly.	

Reference: Custom-Splitter Properties

Common Tab

Property	XML Attribute Name	Description
Spring Bean	`ref`	The Spring bean that implements your **custom-splitter** interface class.

©2026 Workday, Inc. All rights reserved

Reference: Standard-Splitter Properties

Common Tab

Property	XML Attribute Name	Description
Output Mimetype	`output-mimetype`	The MIME type of your tokenized output data.
Line Token Delimiter Regexp	`line-token-delimiter-regexp`	The delimiting regular expression that extracts tokenized data from bulk messages.
Multiple Lines Separator	`multiple-lines-separator`	A string that concatenates tokens spanning multiple lines.

Reference: XPath-Splitter Properties

Common Tab

Property	XML Attribute Name	Description
Namespaces	`namespaces`	The namespaces used by your XPath expression.
XPath	`xpath`	The XPath expression that determines where the input messages are split.
XPath Version	`xpath-version`	Specifies whether to use XPath version 1.0, 2.0, or 3.0. Enter the value 1 for assembly versions through Workday 11. Enter 2 or 3 for assembly versions Workday 12 onwards. You can enable Workday to set the version dynamically using an MVEL template that inserts a property value. Example: `@{props.get('the_version')}`.

Reference: XML-Stream-Splitter Properties

Common Tab

Property	XML Attribute Name	Description
Namespaces	`namespaces`	The namespaces used by your XPath expression.
XPath	`xpath`	The XPath expression that extracts XML from bulk input messages.
XPath Version	`xpath-version`	Specifies whether to use XPath version 1.0, 2.0, or 3.0.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Enter the value 1 for assembly versions through Workday 11. Enter 2 or 3 for assembly versions Workday 12 onwards. You can enable Workday to set the version dynamically using an MVEL template that inserts a system property value. Example: `@{props.get('the_version')}`.

Reference: JSON-Splitter Properties

Common Tab

Property	XML Attribute Name	Description
Json Path	`json-path`	The expression that extracts JSON from bulk input messages.

Reference: Unzip-Splitter Properties

Common Tab

Property	XML Attribute Name	Description
File Pattern	`file-pattern`	The regular expression that extracts data from ZIP and TAR input files.
Format	`format`	The format of your input files. The options are: • `zip` • `tar`

Reference: Mtable-Splitter Properties

Common Tab

Property	XML Attribute Name	Description
Mtable Name	`mtable-name`	The name of the `MediationContext` property that contains the `mtable`.
Row Name	`row-name`	The name of the `MediationContext` property that stores the iterated row.
Row Format	`row-format`	Determines the class of the iterated row. Supported values are:

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• `array` • `map`

Standard-Splitter Child Elements

Concept: Standard-Splitter Child Elements

The **standard-splitter** component supports these child elements.

Child Element	Description
header-ends-with	Indicates the end of the header section in your document.
header-fixed-lines	Indicates the number of lines occupied by the header section in your document.
content-fixed-lines	Indicates the number of lines occupied by the data records content section in your document.
content-starts-ends	Indicates the start point and end point of the data records content section in your document.
content-starts-with	Indicates the start point of the data records content section in your document.
footer-fixed-lines	Indicates the number of lines occupied by the footer section in your document.
footer-starts-with	Indicates the start point of the footer section in your document.

Reference: Header-Ends-With Properties

Common Tab

Property	XML Attribute Name	Description
Include In Header	`include-in-header`	Determines whether the matching line of your regular expression is part of the header.
Regexp	`regexp`	A regular expression that identifies the end of the header.

Reference: Header-Fixed-Lines Properties

Common Tab

Property	XML Attribute Name	Description
Lines Count	`lines-count`	The number of lines occupied by the header.

©2026 Workday, Inc. All rights reserved

Reference: Content-Fixed-Lines Properties

Common Tab

Property	XML Attribute Name	Description
Ignore Lines Regexp	`ignore-lines-regexp`	A regular expression that excludes matching lines from the data records content section.
Lines Count	`lines-count`	The number of lines occupied by the data records content section.

Reference: Content-Starts-Ends Properties

Common Tab

Property	XML Attribute Name	Description
End Regexp	`end-regexp`	A regular expression that identifies the end of the data records content section.
Start Regexp	`start-regexp`	A regular expression that identifies the start of the data records content section.

Reference: Content-Starts-With Properties

Common Tab

Property	XML Attribute Name	Description
Regexp	`regexp`	A regular expression that identifies the start of the data records content section.

Reference: Footer-Fixed-Lines Properties

Common Tab

Property	XML Attribute Name	Description
Ignore Lines Regexp	`ignore-lines-regexp`	A regular expression that excludes matching lines from the footer section.
Lines Count	`lines-count`	The number of lines occupied by the footer section.
Trim Eof Ignore Lines	`trim-eof-ignore-lines`	Determines whether the matching lines of your regular expression are trimmed from the footer.

©2026 Workday, Inc. All rights reserved

Reference: Footer-Starts-With Properties

Common Tab

Property	XML Attribute Name	Description
Regexp	`regexp`	A regular expression that identifies the start of the data records content section.

Aggregator Component Child Elements

Concept: Aggregator Component Child Elements

The **aggregator** component supports these child elements.

Child Element	Description
custom-batch-strategy	A customizable batching strategy that you configure as a Spring bean.
size-batch-strategy	Batches incoming messages based on a specified batch size value.
time-batch-strategy	Batches incoming messages based on a specified time period value.
custom-collater	A customizable collating strategy that you configure as a Spring bean.
message-content-collater	Aggregates the message content of the `MediationMessage`.
xml-message-content-collater	Aggregates the XML document content of an XML message stream.
json-collater	Aggregates JSON documents into a single output.
zip-file-collater	Aggregates multiple files into a single ZIP file.
mtable-collater	Aggregates `mtable` rows into a new `mtable` instance.

Reference: Custom-Batch-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Spring Bean	`ref`	The Spring bean that implements your **custom-batch-strategy** interface class.

©2026 Workday, Inc. All rights reserved

Reference: Size-Batch-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Batch Size	`batch-size`	The maximum number of messages that a batch concatenates.

Reference: Time-Batch-Strategy Properties

Common Tab

Property	XML Attribute Name	Description
Time Period	`time-period`	The duration of batch aggregation.
Time Unit	`time-unit`	The unit of time implemented by the **Time Period** property. The default value is **seconds**.

Reference: Custom-Collater Properties

Common Tab

Property	XML Attribute Name	Description
Output	`output`	Specifies where the **custom-collater** child element directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Spring Bean	`ref`	The Spring bean that implements your **custom-collater** interface class.

Reference: Message-Content-Collater Properties

Common Tab

Property	XML Attribute Name	Description
Output	`output`	Specifies where the **message-content-collater** child element directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Header Text	`header-text`	Adds a header to your output message.
Footer Text		Adds a footer to your output message.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Separator	`separator`	A string used when concatenating messages in a batch.

Reference: XML-Message-Content-Collater Properties

Common Tab

Property	XML Attribute Name	Description
Output	`output`	Specifies where the **xml-message-content-collater** child element directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Header Text	`header-text`	Adds a header to your output message.
Footer Text		Adds a footer to your output message.
Separator	`separator`	A string used when concatenating messages in a batch.
Namespaces	`namespaces`	The namespaces used by your XPath expression.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
XPath	`xpath`	An XPath expression that aggregates matching XML messages. If not specified, the **xml-message-content-collater** aggregates all XML message content into 1 output file.

Reference: JSON-Aggregator Properties

Common Tab

Property	XML Attribute Name	Description
Output	`output`	Specifies where the **json-collater** child element directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Reference: Zip-File-Collater Properties

Common Tab

Property	XML Attribute Name	Description
Output	`output`	Specifies where the **zip-file-collater** child element directs

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Format	`format`	The format of your output files. The options are: • `zip` • `tar`
Message Entity Name	`message-entity-name`	The name of each message added to your output file.

Reference: Mtable-Collater Properties

Common Tab

Property	XML Attribute Name	Description
Output	`output`	Workday ignores this setting for **mtable-collater** child elements.
Mtable Name	`mtable-name`	The name of the `MediationContext` property that stores your output `mtable`.
Row Name	`row-name`	The name of the `MediationContext` property

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		that stores your aggregated `mtable` row.
Column Names	`column-names`	A comma-separated list of columns that defines your output `mtable`.
Index Column	`index-column`	The index column name of your output `mtable`.
Allow Null Row	`allow-null-row`	Determines whether your output `mtable` supports null row values.

Common Components

Concept: Common Components

Workday Studio provides a range of prepackaged subassemblies that you can use to build your integrations. These subassemblies are available from the **Common Components** tab of the **Palette**. When you add one to your assembly diagram, Studio inserts a `local-out` element in the XML source with the appropriate endpoint in Workday.

It also displays the parameters you can use with that subassembly, with required parameters already selected. You can adjust parameters at any time in the subassembly's **Properties** view.

Studio includes these **Common Components** on the **Palette**:

Name	Function	Notes
Ftp	Routes to an **ftp-out** transport.	• Provides a single entry point for all FTP transports. • Uses the `wd.ftp.endpoint` parameter value to determine the type of FTP transport implemented. Examples: `ftp://`, `sftp://`, or `ftps://`. • Exposes the transport settings that are commonly used in Workday, such as username and password authentication. Hides more advanced public key authentication. Note: The FTP Subassembly is no longer available from the Studio Palette. In new integrations, you should use the **ftp-out** or **sftp-out** components to send documents to an FTP server. These

©2026 Workday, Inc. All rights reserved

Name	Function	Notes
		components offer improved security, better performance, and enhanced reliability. They're also compatible with later versions of JSCAPE. The FTP Subassembly remains functional in existing integrations, but Workday recommends that, where practical, you replace it.
GetEventConfigurations	Retrieves the service configurations attached to a particular integration event.	
GetEventDocuments	Retrieves the documents attached to a particular integration event.	
GetIntegrationEvent	Retrieves an integration event based on a Workday reference ID.	
GetIntegrationSystems	Retrieves the integration system's configuration from Workday and parses it.	
PagedGet	Implements the paging logic required by any `paged-get` Web service operation, enabling page-by-page data retrieval.	• Use the `is.paged.get.process.endpoint` parameter to send each page of response data to another subassembly for processing. • Use the `is.paged.get.aggregate.header`, `is.paged.get.aggregate.footer`, and `is.paged.get.aggregate.xpath` parameters to aggregate all pages of response data. • Aggregated XML data is available as the message in the `MediationContext` when the subassembly returns. • If a web service is expected to return a large number of pages, use the per-page processing approach.
PagedGetLocalPaging	Implements the paging logic required by any `paged-get` local Web service operation, enabling page-by-page data retrieval.	• Enables the paging process to be performed locally at the ESB server, rather than remotely by Workday.

©2026 Workday, Inc. All rights reserved

Name	Function	Notes
		• Uses a single query to return all Workday IDs. Subsequent calls retrieve the data for each Workday ID.
PdfPrintStep	Converts custom report data from a `workday-out-rest` request to PDF format using a BIRT report design.	• Specify a literal string value for each of the MVEL-capable parameters by enclosing it in single quotes.
PutIntegrationEvent	Updates an integration system's integration event status in Workday.	• Uses the `Put_Integration_Event` web service operation. • Puts the event response Workday ID returned from the Web service operation in the `MediationContext` property `integrations.event.response.wi`
PutIntegrationMessage	Sends update integration messages to Workday.	• Can also add documents in the form of references to Workday document storage. • Uses the `Put_Integration_Message` web service operation. • Workday limits the number of calls a **PutIntegrationMessage** subassembly can make in a single integration to 500. Subsequent integration messages are logged to a log file and can't be viewed in the integration event. (Redirect doesn't apply to messages that have attachments or change the integration event status.)
SalesforceConnector	Sends authenticated messages to salesforce.com.	
PrismConnector	Sends CSV data to Prism Analytics.	

©2026 Workday, Inc. All rights reserved

Reference: FTP Subassembly Properties

Common Tab

Note: The FTP Subassembly is no longer available from the Studio Palette. In new integrations, you should use the **ftp-out** or **sftp-out** components to send documents to an FTP server. These components offer improved security, better performance, and enhanced reliability. They're also compatible with later versions of JSCAPE. The FTP Subassembly remains functional in existing integrations, but Workday recommends that, where practical, you replace it.

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **Ftp** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **Ftp** subassembly.
Execute When	`execute-when`	A condition that determines whether the **Ftp** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **Ftp** component supports a `vm://` endpoint address. Example: `vm://wcc/Ftp`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`wd.ftp.endpoint`	The endpoint scheme used to determine which **ftp-out** transport to use. The endpoint must start with `ftp://`, `sftp://`, or `ftps://`.
`wd.ftp.username`	The username used to access the FTP server.
`wd.ftp.password`	The password associated with the username used to access the FTP server.
`wd.ftp.method`	The method or message action that the FTP subassembly implements when connecting with the endpoint. Enclose each action in single quotes. Valid options include: • `'put'`: The FTP component sends the message to the FTP server and the message remains unchanged. • `'get'`: The assembly replaces the root part of the message with the data that it receives from the FTP server. • `'list'`: The message remains unchanged. Workday adds a context property called `wd.ftp.files.list`, which contains a `List<String>` object instance. Note: Using the `'list'` method with a file pattern of `*.txt` or `*.xml` results in an exception.

©2026 Workday, Inc. All rights reserved

Parameter	Description
	• `'delete'`: Workday requests a file deletion from the FTP server. The message remains unchanged. • `'mget'` (multiget): Workday downloads all files that match the input pattern and replaces the root part of the message with a ZIP file containing those files. • `'mdelete'` (multidelete): Workday deletes all files that match the input pattern. The message remains unchanged.
`wd.ftp.passive.mode`	Specifies whether passive mode is used during the FTP connection.
`wd.ftp.file.pattern`	Specifies the file pattern used when performing the FTP operation. Examples: • `*.xml` • `*.txt` For most methods other than `list`, Workday supports wildcards and performs filtering on the client side. For the `list` method, Workday uses server-side filtering, based on regular-expression syntax. Examples: • `wd.ftp.method` is `'get'` and `wd.ftp.file.pattern` is `*.xml`: Workday gets the first .xml file found on the FTP server in the specified directory. • `wd.ftp.method` is `'mget'` and `wd.ftp.file.pattern` is `*.xml`: Workday gets all the .xml files found on the FTP server in the specified directory and returns them in ZIP format. • `wd.ftp.method` is `'list'` and `wd.ftp.file.pattern` is `^.*\.(xml)$`: Workday lists all the .xml files found on the FTP server in the specified directory.
`wd.ftp.temp.file.name`	Specifies whether a temporary file should be used when performing an FTP `put` message action.
`wd.ftp.timeout`	A timeout for the message action in milliseconds.
`wd.ftp.directory`	The relative or fully qualified directory path.
`wd.ftp.private.key`	The private key associated with the username for SFTP authentication. Formatted as a URI using the attachment protocol and a Workday ID. Example: `attachment:privatekey/@{WID}`, where WID corresponds to a 32-character Workday ID.
`wd.ftp.max.attempts`	The maximum number of attempts to access the FTP server.

©2026 Workday, Inc. All rights reserved

Parameter	Description
`wd.ftp.debug`	Specifies whether Workday will add debug logging to the logs.
`wd.ftp.proxy.host`	The hostname of the FTP proxy server.
`wd.ftp.proxy.port`	The FTP proxy server port.
`wd.ftp.proxy.username`	The username for accessing the FTP proxy server.
`wd.ftp.proxy.password`	The password of the FTP proxy server.
`wd.ftp.proxy.type`	The type of the proxy server.
`wd.ftp.input.content.type`	Specifies the content type of files retrieved using FTP `get`. Example: `text/plain; charset=Big5`.

Reference: GetEventConfigurations Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **GetEventConfigurations** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **GetEventConfigurations** subassembly.
Execute When	`execute-when`	A condition that determines whether the **GetEventConfigurations** subassembly executes.
Endpoint	`endpoint`	Specifies the target endpoint address for the subassembly. The **GetEventConfigurations** component supports a `vm://` endpoint address. Example: `vm://wcc/GetEventConfigurations`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`ie.event.wid`	The Workday ID for the integration event whose service configurations you want to retrieve. Automatically populates with `lp.isSet() ? lp.getIntegrationEventWID() : null`.
`is.event.disable.msg.storage`	Switches off message storage when set to `true`.
`ie.event.wws.version`	The integration event Web Service version.

©2026 Workday, Inc. All rights reserved

Reference: GetEventDocuments Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **GetEventDocuments** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **GetEventDocuments** subassembly.
Execute When	`execute-when`	A condition that determines whether the **GetEventDocuments** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **GetEventDocuments** component supports a `vm://` endpoint address. Example: `vm://wcc/GetEventDocuments`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`ie.event.wid`	The Workday ID for the integration event whose service configurations you want to retrieve. Automatically populates with `lp.isSet() ? lp.getIntegrationEventWID() : null`. This means that the parameter's value defaults to the integration event WID if the launch document has been specified in the mediation context and to null if it has not.

Reference: GetIntegrationEvent Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **GetIntegrationEvent** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **GetIntegrationEvent** subassembly.
Execute When	`execute-when`	A condition that determines whether the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		GetIntegrationEvent subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **GetIntegrationEvent** component supports a `vm://` endpoint address. Example: `vm://wcc/GetIntegrationEvent`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`ie.event.wid`	The Workday ID for the integration event whose service configurations you want to retrieve.

Reference: GetIntegrationSystems Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **GetIntegrationSystems** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **GetIntegrationSystems** subassembly.
Execute When	`execute-when`	A condition that determines whether the **GetIntegrationSystems** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **GetIntegrationSystems** component supports a `vm://` endpoint address. Example: `vm://wcc/GetIntegrationSystems`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`is.system.wid`	The Workday ID for the integration system whose service configurations you want to retrieve and parse. Automatically populates with `lp.getIntegrationSystemRefWID()`.

©2026 Workday, Inc. All rights reserved

Reference: PagedGet Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **PagedGet** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **PagedGet** subassembly.
Execute When	`execute-when`	A condition that determines whether the **PagedGet** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **PagedGet** component supports a `vm://` endpoint address. Example: `vm://wcc/PagedGet`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`is.paged.get.request.current.page.xpath`	The XPath location in the request document where the `PagedGet` component sets the number of the pages being requested.
`is.paged.get.response.current.page.xpath`	The XPath expression to extract the current page value from the web service response message.
`is.paged.get.response.total.pages.xpath`	The XPath expression to extract the total pages value from the web service response message.
`is.paged.get.response.total.results.xpath`	The XPath expression to extract the total results page value from the web service response message.
`is.paged.get.process.endpoint`	The **local-in** endpoint of the subassembly that processes each page of results. Example: `vm://payrollInterface/ProcessPayeesLocalIn`. Note: Workday ignores aggregate parameters when you set this parameter. These `MediationContext` properties are available in the processing subassembly: • `is.paged.get.request.current.page.xpath` • `is.paged.get.response.current.page.xpath` • `is.paged.get.response.total.pages.xpath` • `is.paged.get.response.total.results.xpath`
`is.paged.get.aggregate.header`	The header value for aggregated responses in the form of an XML fragment. Example: `<AllData>`.
`is.paged.get.aggregate.footer`	The header value for aggregated responses in the form of an XML fragment. Example: `</AllData>`.

©2026 Workday, Inc. All rights reserved

Parameter	Description
`is.paged.get.aggregate.xpath`	The XPath expression to apply to the response message when aggregating.
`is.paged.get.namespaces`	Specifies namespaces that aren't available on the document.
`is.paged.get.application`	The application group of the web service you're invoking. Example: `Human_Resources`.
`is.paged.get.version`	The version of the public web service you're invoking. As best practice, use the latest WWS version at the time you're building the integration. Don't use a dynamic latest version that changes as the integration runs. For assemblies with a version before 2016.45, this parameter automatically populates with `util.assemblyVersionAsWWSVersion`. For assemblies with a version 2016.45 or later, you must explicitly supply the Web services version. Example: `v27.2`.
`is.paged.get.page.zero`	Specifies whether a response page should be generated for cases where the total number of pages is zero.
`is.paged.get.parallel`	Specifies whether to retrieve web service responses in parallel, enabling the subassembly named in the `is.paged.get.process.endpoint` parameter to process each paged response. Automatically populates with `false`, meaning that web service responses are aggregated rather than retrieved in parallel. Workday processes parallel responses using unique contexts that enable read-only access to the property values and variables that were set before the **PagedGet** was entered. When the subassembly finishes with each page, you lose any properties or variables set during that processing. The **PagedGet** still sets the `is.paged.get.total.pages` and `is.paged.get.total.results` properties on the calling `MediationContext`. The `is.paged.get.current.page` property is set in the parallel paged contexts. Workday processes nonparallel responses in the same context and processes the pages in the subassembly. This means that you can set properties and variables in the subassembly and check the state of previous pages.
`is.paged.get.store.requests`	Specifies whether to store the web service request messages for each page of data in the message store.

©2026 Workday, Inc. All rights reserved

Parameter	Description
`is.paged.get.overall.timeout.seconds`	The overall timeout in seconds. The default, if unset, is 6 hours. The maximum timeout you can set is 30 hours.
`is.paged.get.timeout.seconds`	The timeout for processing each page. The default, if unset, is 6 hours. Can't exceed the overall timeout value.

Out Parameters
As the **PagedGet** subassembly processes, it passes these out-parameter values for use by downstream components:

Out-Parameter	Description
`is.paged.get.last.page`	The final page returned by the web service operation.
`is.paged.get.current.page`	The number of the page currently being processed.
`is.paged.get.total.pages`	The total number of pages to be processed.
`is.paged.get.total.results`	The total number of results returned by the web service operation.

Reference: PagedGetLocalPaging Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **PagedGetLocalPaging** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **PagedGetLocalPaging** subassembly.
Execute When	`execute-when`	A condition that determines whether the **PagedGetLocalPaging** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **PagedGetLocalPaging** component supports a `vm://` endpoint address. Example: `vm://wcc/PagedGet`.

©2026 Workday, Inc. All rights reserved

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`is.paged.get.process.endpoint`	Specifies the **local-in** endpoint of the subassembly that will process each page of

©2026 Workday, Inc. All rights reserved

Parameter	Description
	results. Example: `vm://payrollInterface/ProcessPayeesLocalIn`. Note: Workday ignores aggregate parameters when you set this parameter. These `MediationContext` properties are available in the processing subassembly: • `is.paged.get.request.current.page.xpath` • `is.paged.get.response.current.page.xpath` • `is.paged.get.response.total.pages.xpath` • `is.paged.get.response.total.results.xpath`
`is.paged.get.page.size`	The number of records in each page of data to be retrieved. Automatically populates with `100`.
`is.paged.get.response.reference.wids.xpath`	The XPath expression used to find the WIDs in the get all WIDs invocation response message. Example: `//wd:Payee_Reference/wd:ID[@wd:type='WID']`.
`is.paged.get.request.reference.wids.parent.xpath`	The XPath expression used to find the parent element of the request message where WIDs will be indicated when getting each page of data.
`is.paged.get.request.reference.wids.element.name`	The element name for the WID to be used in the request for each page of data.
`is.paged.get.application`	The application group of the web service you're invoking. Example: `Human_Resources`.
`is.paged.get.version`	The version of the public web service you're invoking. As best practice, use the latest WWS version at the time you're building the integration. Don't use a dynamic latest version that changes as the integration runs. For assemblies with a version prior to 2016.45, this parameter automatically populates with `util.assemblyVersionAsWWSVersion`. For assemblies with a version 2016.45 or later, you must explicitly supply the Web services version. Example: `v27.2`.
`is.paged.get.xpath.ns.prefix`	The prefix for the Workday XML namespace used in all XPath expressions. Automatically populates with `wd`.

Reference: PdfPrintStep Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **PdfPrintStep** subassembly within the assembly.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Routes Response To	`routes-response-to`	The destination of the response message for the **PdfPrintStep** subassembly.
Execute When	`execute-when`	A condition that determines whether the **PdfPrintStep** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **PdfPrintStep** component supports a `vm://` endpoint address. Example: `vm://wcc/PdfPrintStep`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`pdf.report.design.variable`	The name of the integration variable that contains the report design, which defines the layout for the printed custom report. Example: `'test.report.design'`. The report design definition is contained in a file with the extension.rptdesign.
`pdf.workday.report.variable`	The name of the integration variable that contains the raw Workday report data. Example: `'test.workday.report'`.
`pdf.timezone`	The Java time zone for formatting purposes. Example: • `'US/Eastern'` • `'GMT'` • `'CET'`
`pdf.language`	The language of the report for formatting purposes. Example: • `'en_US'` • `'fr_FR'` • `'de_DE'`
`pdf.locale`	The locale of the report for formatting purposes. Example: • `'en_US'` • `'fr_FR'` • `'de_DE'`
`pdf.business.form.layout.wid`	The Workday ID of the Business Form Layout being used. Enables retrieval of images defined as tokens on the layout.
`pdf.report.design.wid`	The Workday ID of the report design being used. Enables translation.
`pdf.apply.report.design.per.row`	Specifies whether Workday applies the report design to individual report rows, rather than to the report as a whole.

©2026 Workday, Inc. All rights reserved

Related Information
Reference
The Next Level: Using BIRT in Studio Integrations

Reference: PutIntegrationEvent Subassembly Properties

Common Table

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **PutIntegrationEvent** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **PutIntegrationEvent** subassembly.
Execute When	`execute-when`	A condition that determines whether the **PutIntegrationEvent** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **PutIntegrationEvent** component supports a `vm://` endpoint address. Example: `vm://wcc/PutIntegrationEvent`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`is.event.wid`	The Workday ID of the integration event to which Workday will link the message. If you don't specify this value and it isn't present in the `lp` MVEL variable, Workday uses the `is.system.wid` parameter instead, linking the message to the integration system. Automatically populates with `lp.isSet() ? lp.getIntegrationEventWID() : null`.
`is.system.wid`	The Workday ID of the integration system. Automatically populates with `lp.isSet() ? lp.getIntegrationSystemRefWID() : null`. Workday only uses this parameter if the integration event ID isn't available.
`is.process.desc`	The integration event description. Workday displays this value in its Process Monitor. Required when creating a new integration event.
`is.event.initiated`	The date and time when the integration event initiates. If you enter the string value `'current'`, the current date and time is sent to Workday.

©2026 Workday, Inc. All rights reserved

Parameter	Description
`is.event.completed`	The date and time when the integration event completes. If you enter the string value `'current'`, the current date and time is sent to Workday.
`is.response.msg`	The most recent message or activity for the integration event. A free-form text field.
`is.event.member.wids`	A comma-separated list of Workday IDs representing members linked to this integration event.
`is.percent.complete`	A value that Workday uses to maintain the Percent Complete progress bar. The value can be any type that can be converted to a Java BigDecimal object. It must be a value between zero and 100.

Reference: PutIntegrationMessage Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **PutIntegrationMessage** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **PutIntegrationMessage** subassembly.
Execute When	`execute-when`	A condition that determines whether the **PutIntegrationMessage** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **PutIntegrationMessage** component supports a `vm://` endpoint address. Example: `vm://wcc/PutIntegrationMessage`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`is.event.wid`	The Workday ID of the integration event to which Workday will link the message. If you don't specify this value and it isn't present in the `lp` MVEL variable, Workday uses the `is.system.wid` parameter instead, linking the message to the integration system. Automatically populates with `lp.isSet() ? lp.getIntegrationEventWID() : null`.

©2026 Workday, Inc. All rights reserved

Parameter	Description
`is.system.wid`	The Workday ID of the integration system. Automatically populates with `lp.isSet() ? lp.getIntegrationSystemRefWID() : null`. Workday only uses this parameter if the integration event ID isn't available.
`is.message.severity`	Specifies the message severity level: `'CRITICAL'`, `'DEBUG'`, `'ERROR'`, `'INFO'`, or `'WARNING'`. Automatically populates with `'INFO'`.
`is.message.summary`	The message summary sent to Workday.
`is.message.detail`	The message detail sent to Workday.
`is.message.detail.richtext`	Determines whether the message detail is formatted as rich text. Automatically populates with `true`.
`is.document.variable.name`	The name of a variable in the `MediationContext` that contains the Atom schema output of an assembly **store** step. The variable content is used to add a reference to the integration event. The `store` step used to create the document must have the schema attribute set as follows: `schema="http://www.w3.org/2005/Atom"`.
`is.document.file.name`	The filename for the document. If you don't provide one, Workday uses the document title from the Atom document.
`is.document.owner.wid`	The Workday ID of the document owner. If you don't provide one, Workday uses the current integration system user's ID.
`is.document.labels`	A comma-separated list of labels that are applied to the document when it's within the scope of the integration event ID or associated Business Process.
`is.document.deliverable`	Specifies whether the document added to the integration event is automatically delivered to a customer or external provider. Automatically populates with `'true'`.
`is.document.delivery.services`	A comma-separated list of integration service names that's used to tag the document being attached to the integration event. By default, Workday uses all delivery services associated with the integration system.
`is.document.retrieved`	Specifies whether the document is treated as having been retrieved by the retrieval service. Automatically populates with `'false'`.

©2026 Workday, Inc. All rights reserved
Workday Proprietary and Confidential

Parameter	Description
`is.message.targets`	A comma-separated list of references to target objects for this message. These targets become object links in the Workday application **View Integration Message** tab.
`is.message.targets.type`	A target reference type. Automatically populates with `'WID'`.
`is.message.secured.instance.refs`	A comma-separated list of secured instance references for this message.
`is.message.secured.instance.refs.type`	Enables the default type for secured instance refs to be overridden. Automatically populates with `'WID'`.
`is.message.storage.enabled`	Determines whether the message is stored. Automatically populates with `'false'`.

Reference: SalesforceConnector Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **SalesforceConnector** subassembly within the assembly.
Routes Response To	`routes-response-to`	The destination of the response message for the **SalesforceConnector** subassembly.
Execute When	`execute-when`	A condition that determines whether the **SalesforceConnector** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **SalesforceConnector** component supports a `vm://` endpoint address. Example: `vm://wcc/SalesforceConnector`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Transport Class	`transport-class`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`wd.sf.auth.username`	Your salesforce.com developer account username.
`wd.sf.auth.password`	Your salesforce.com developer account password.
`wd.sf.msg.varname`	The XML message that Workday sends to the salesforce.com endpoint.

©2026 Workday, Inc. All rights reserved

Parameter	Description
`wd.sf.auth.token`	Your salesforce.com developer account security token.
`wd.sf.auth.endpoint`	An optional salesforce.com endpoint. A default endpoint is already configured as part of the subassembly's `http-out` transport.
`wd.sf.api.custom.version`	An optional salesforce.com API version. If you don't provide a value, the endpoint API version is used.

The PrismAnalytics Subassembly

Steps: Send CSV Data to Prism Analytics

Prerequisites

Security: *Security Administration* domain in the System functional area.

Context

Workday Studio's **PrismAnalytics** component enables you to send CSV data to Workday Prism Analytics. Before you configure an assembly to use the **PrismAnalytics**, you must register a Prism API client in Workday and obtain a REST API endpoint. You must also create a JSON description of the source CSV data.

Steps

1. Register a Workday Prism Analytics API Client .

   In Workday, register an integrations API client with Prism Analytics as its scope. Obtain the **Client ID**, **Client Secret**, and **Refresh Token** values that the **PrismAnalytics** component requires as parameters.
2. Obtain the Workday REST API Endpoint .

   In Workday, obtain the Workday REST API endpoint that the **PrismAnalytics** component requires as a parameter.
3. Describe the Source CSV Data .

   Describe the source CSV data in JSON text for passing to the **PrismAnalytics** component as a variable.
4. Configure an Assembly with the PrismAnalytics Subassembly .

   In Workday Studio, create an assembly that acquires CSV data and passes it to a properly configured **PrismAnalytics** component.

Related Information
Reference
Workday 31 What’s New Post: Workday Studio

Register a Workday Prism Analytics API Client

Prerequisites

Security: *Security Administration* domain in the System functional area.

©2026 Workday, Inc. All rights reserved

Context

To use Workday Studio's **PrismAnalytics** component, you must register an API client in Workday with Prism Analytics as its scope. The component requires the **Client ID** and **Client Secret** values that Workday generates during the registration process. It also requires a **Refresh Token** value.

Steps

1. Access the **Register API Client for Integrations** task. As you complete this task, consider:

Option	Description
Non-Expiring Refresh Tokens	Prevents the refresh token from timing out. Automatically selected by Workday. Don't clear.
Scope (Functional Area)	Select **Prism Analytics**.
Include Workday Owned Scope	Provides access to core Workday domains that aren't in any functional areas. Select this check box.

2. Copy the **Client ID** and **Client Secret** values. Studio's **PrismAnalytics** component requires them for its `wd.prism.component.client.id` and `wd.prism.component.client.secret` parameters.
3. From the related actions menu, select **API Client > Manage Refresh Tokens for Integrations**.
4. Select a **Workday Account** to associate with the refresh token. The account must have permission to create Prism tables in Workday.

   Note: You can't perform these steps as an Integration System User.
5. Select **Generate New Refresh Token**.
6. Copy the **Refresh Token** value. The **PrismAnalytics** component requires it for its `wd.prism.component.refresh.token` parameter.

Obtain the Workday REST API Endpoint

Prerequisites

Security: *Security Administration* domain in the System functional area.

Context

Studio's **PrismAnalytics** component requires a Workday REST API endpoint. You can find this endpoint in Workday.

Steps

1. Access the **View API Clients** report.
2. Copy the **Workday REST API Endpoint** value.

   Example: `https://i-07d55d20e81e2d191.workdaysuv.com/ccx/api/v1/super`.
3. Delete the `https://` at the beginning and the `/ccx/api/v1/<tenant_name>` at the end.

   Example: `i-07d55d20e81e2d191.workdaysuv.com`.
4. Copy the text. Studio's **PrismAnalytics** component requires it for its `wd.prism.component.api.hostname` parameter.

©2026 Workday, Inc. All rights reserved

Describe the Source CSV Data

Context

Workday Studio's **PrismAnalytics** component requires a description of the source CSV data. You can create the JSON text in the tool of your choice and use it in Studio later. Example: paste it into a text message on the **Message Builder** tab of a **write** step.

Steps

1. Create JSON text containing 2 main nodes:
    - `parseOptions`: information about the format of the CSV data, including the character set and the delimiter character.
    - `fields`: information about each field in the CSV data, including data type and formatting.
2. In the `parseOptions` node, you can define these fields:

Field Name	Type	Required	Description
`charsetName`	Object	Yes	The CSV data file's character set. Example: `"charset": { "id": "Encoding=UTF-8" }` Currently, Workday supports the UTF-8 encoding format only. Workday ultimately stores the data in Unicode format. Characters with no Unicode equivalent are stored as the Unicode replacement character, a question mark on a black diamond background.
`fieldsEnclosedBy`	String	No	The character used to enclose fields in the CSV data. The default is double quote marks. If a double quote mark is used within a field, escape it by preceding it with another double quote mark. Example: `"fieldsEnclosedBy": "\""`

©2026 Workday, Inc. All rights reserved

Field Name	Type	Required	Description
`fieldsDelimitedBy`	String	Yes	The character used to separate field values in the CSV data. Example: `"fieldsDelimitedBy": ","`
`headerLinesToIgnore`	Number	No	The number of lines for the parser to ignore as a header. Set to `0` if the CSV data doesn't have a header. Example: `"headerLinesToIgnore": 1`
`type`	Object	Yes	The input file's type. Currently, Workday supports delimited files only. Example: `"type": { "id": "Schema_File_Type=Delimit }` Note: This value is case-sensitive. Note the upper-case D in Delimited.

3. In the `fields` node, you can define these fields:

Field Name	Type	Required	Description
`ordinal`	Number	Yes	The order of the field in the CSV dataset (index). The ordinal values: • Must start with 1. • Must be contiguous. • Can't skip any number. • Have a maximum of 1000.
`name`	String	Yes	The name of the CSV dataset field. Field names:

©2026 Workday, Inc. All rights reserved

Field Name	Type	Required	Description
			• Must be unique in the dataset. • Can contain up to 255 characters. • Can only include alphanumeric and underscore characters. • Must start with a letter. • Can't end with an underscore character. • Can't begin with WPA_.
`description`	String	No	The field's description. Maximum 1000 characters. Example: `"description": "The worker's full name."`
`type`	Object	Yes	The field's type. Options are: • Text • Numeric • Boolean • Date Example: `"type": { "id":"Schema_Field_Type=Da`
`parseFormat`	String	Yes (for Date types)	The date's format. Example: `"parseFormat": "dd-MM-yy HH24:mm:ss"`
`precision`	Number	Yes (for Numeric types)	The maximum number of digits in a numeric value. Includes all numbers to the left and right of a decimal point but not the decimal point

©2026 Workday, Inc. All rights reserved

Field Name	Type	Required	Description
			itself. Maximum value is 38. Example: `"precision": 38`
`scale`	Number	Yes (for Numeric types)	The number of digits to the right of the decimal point in a numeric value. Must be less than the `precision` value. Example: `"scale": 2`
`defaultValue`	Text	No	The default value for a field, if none is provided in the upload file. Express a default text value in double quotes. Example: `"N/A"`. Default numeric values must agree with the specified `precision` and `scale`. Default date values must be in this format: yyyy-MM-dd'T'HH:mm:ss'Z'.
`required`	Boolean	No	Specifies whether a field must contain a value. Set to `True` if a field is required, otherwise set to `False`. If a field is required, its value can't be null in any insert, update, and upsert data operation.
`externalID`	Boolean	Yes (to take advantage of update, upsert, and delete operations).	Specifies whether a field represents an External ID. Set to `True` to designate a field an External ID, otherwise set to `False`.

©2026 Workday, Inc. All rights reserved

Field Name	Type	Required	Description
			Because External ID fields can't be null, you should specify that any External ID field is required. You can only specify 1 field as External ID. Specify an External ID in new tables if you want to take advantage of update, upsert, and delete functionality. To use update, upsert, and delete functionality with an existing table, you must modify it so it has a single field designated as External ID. The Prism public API doesn't support the modification of a table that contains data. In such cases, make the modification in Workday.

Example

Your source CSV file describes this table:

Employee	Employee ID	Job Title	Hire Date	Annual Salary
Logan McNeil	21001	Vice President, Human Resources	1/1/2010	213298.00
Oliver Reynolds	21003	Chief Information Officer	1/1/2010	394935.00
Maximilian Schneider	21004	Chief Operating Officer	1/1/2010	255373.00
Teresa Serrano	21005	Controller	1/1/2010	291785.00

You describe it using JSON like this:

```
{
  "parseOptions": {
    "fieldsDelimitedBy": ",",
    "fieldsEnclosedBy": "\"",
    "headerLinesToIgnore": 1,
    "charset": {
```

©2026 Workday, Inc. All rights reserved

```
      "id": "Encoding=UTF-8"
    },
    "type": {
      "id": "Schema_File_Type=Delimited"
    }
  },
  "fields": [
    {
      "ordinal": 1,
      "name": "Employee",
      "description": "Employee",
      "precision": 255,
      "scale": 0,
      "parseFormat": null,
"type": {
       "id": "Schema_Field_Type=Text"
      }
    },
    {
      "ordinal": 2,
      "name": "Employee_ID",
      "description": "Employee ID",
      "precision": 38,
      "scale": 0,
      "parseFormat": null,
      "type": {
        "id": "Schema_Field_Type=Numeric"
      }
    },
    {
      "ordinal": 3,
      "name": "Job_Title",
      "description": "Job Title",
      "precision": 255,
      "scale": 0,
      "parseFormat": null,
      "type": {
        "id": "Schema_Field_Type=Text"
      }
    },
    {
      "ordinal": 4,
      "name": "Hire_Date",
      "description": "Hire Date",
      "precision": 0,
      "scale": 0,
      "parseFormat": "MM/dd/yy",
      "type": {
        "id": "Schema_Field_Type=Date"
      }
    },
    {
      "ordinal": 5,
   "name": "Annual_Salary",
      "description": "Annual Salary",
      "precision": 38,
      "scale": 2,
      "type": {
        "id": "Schema_Field_Type=Numeric"
      }
    }
  ],
  "schemaVersion": {
    "id": "Schema_Version=1.0"
```


©2026 Workday, Inc. All rights reserved

```
    }
}
```

Related Information
Reference
Reference: Supported Date Formats for External Data in Tables and Datasets

Configure an Assembly with the PrismAnalytics Subassembly

Context

To send CSV data to Prism Analytics, your assembly must contain a **PrismAnalytics** component that:

- Receives CSV data as input, either in the rootpart of the message or as a variable.
- Receives a variable containing a JSON description of the CSV data.
- Has a number of parameters configured correctly.

Steps

1. Configure your assembly to obtain a source CSV file or source CSV files. Depending on the method you use to acquire the CSV data, you might need to provide a variable name or set a MIME type.

   As you complete the task, consider:

CSV Acquisition Method	Notes
sftp-out component	• Specify the CSV filename in the component's **Input File Pattern** property. You can use wildcards. • Use the component's `get` method to retrieve a single file or its `mget` method to retrieve multiple files. • Studio adds the CSV data to the message rootpart. • No variable name is required. • Studio automatically sets the MIME type correctly.
Retrieval or listener service	• Studio adds the CSV data to a variable. • Name the variable using the `wd.prism.component.csv.input.varname` parameter. • Studio automatically sets the MIME type correctly.

Any other method	• Studio adds the CSV data to the message rootpart or to a variable.
	• If applicable, name the variable using the `wd.prism.component.csv.input.varname` parameter.
	• Set the MIME type of the rootpart or the variable to:
	  • *text/csv*
	  • *plain/csv*
	  • *application/zip*

©2026 Workday, Inc. All rights reserved

2. Create a text message on the **Message Builder** tab of a **write** step and add the JSON text that describes the data. Output a variable. Example: `temp.schema`.

   Note: You can add the JSON text directly to the **Message Builder**. Alternatively, you can write it in the tool of your choice and paste it in.
3. Configure the **PrismAnalytics** component's parameters:

Parameter	Description
`wd.prism.component.schema.varname`	The variable containing the JSON schema that describes the data.
`wd.prism.component.table.name`	The name of the Prism table. Table names: • Must be unique in the Data Catalog. • Can contain up to 255 characters. • Can only include alphanumeric and underscore characters. • Must start with a letter. • Can't end with an underscore character.
`wd.prism.component.client.id`	The Client ID assigned to the API client by Workday. Access the View API Clients report, then view the API Clients for Integrations tab.
`wd.prism.component.client.secret`	The Client Secret assigned to the API client by Workday. Access the View API Clients report, then view the API Clients for Integrations tab.
`wd.prism.component.refresh.token`	The Refresh Token assigned to the API client by Workday. Access the View API Clients report, then view the API Clients for Integrations tab.
`wd.prism.component.operation`	The operation performed on the Prism table. Replace is the only option currently supported. It creates a table or replaces one of the same name.
`wd.prism.component.csv.input.varname`	(Optional.) The variable containing the CSV input. Not required if you're adding the CSV data to the message rootpart. Note: Studio doesn't automatically select this parameter for inclusion when you add a **PrismAnalytics** component to an assembly. Select it manually if required or add it later on the component's **Parameters** tab.
`wd.prism.component.table.description`	(Optional.) A description for the table. Can contain up to 255 characters.
`wd.prism.component.table.description.displayname`	(Optional.) The display name for the table. Can contain up to 255 characters.
`wd.prism.component.table.field.varname`	(Optional.) Defines a Prism table. Write the definition to a variable then pass it to this parameter using vars['variable-name'].

©2026 Workday, Inc. All rights reserved

Next Steps

Deploy and launch the integration.

Reference: PrismAnalytics Subassembly Properties

Common Tab

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the **PrismAnalytics** subassembly.
Routes Response To	`routes-response-to`	The destination of the response message for the subassembly within the assembly.**PrismAnalytics** subassembly.
Execute When	`execute-when`	A condition that determines whether the **PrismAnalytics** subassembly executes.
Endpoint	`endpoint`	The target endpoint address for the subassembly. The **PrismAnalytics** component supports a `vm://` endpoint address. Example: `vm://wcc/PrismAnalytics`.

Advanced Tab

Property	XML Attribute Name	Description
Store Message	`store-message`	Specify *none* unless otherwise advised by Workday Support. To store messages, use Studio's **store** step.
Transport Class	`transport`	Specifies an alternative implementation class on a per-assembly basis, if required. You can globally change the implementation classes by editing the WEB-INF/classes/spring/assembly-bean.xml file.
Ignore Dynamic Endpoints	`ignore-dynamic-endpoints`	Specifies whether to ignore the dynamic WS-Addressing (WSA) value in the incoming message's header or use it to override the value in the transport's **Endpoint** property. Use the **set-dynamic-endpoint** step to specify WS-Addressing details.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Clone Request	`clone-request`	Specifies whether to chain processing between subassemblies. Set to **true** to create a copy of the original message so that any subassembly modifying the request message won't affect the others. Set to **false** to pass the original message.
Propagate Abort	`propagate-abort`	Specifies whether an abort flag propagates from a subassembly to the calling assembly. A subassembly stops processing when it marks the `MediationContext` as aborted.
Unset Properties	`unset-properties`	Specifies whether all **local-out** properties reset after the **local-out** transport calls a subassembly.

Parameters Tab

Parameter	Description
`wd.prism.component.schema.varname`	The variable containing the JSON schema that describes the data.
`wd.prism.component.table.name`	The Prism table's identifier in the Data Catalog. Table names: • Must be unique in the Data Catalog. • Can contain up to 255 characters. • Can only include alphanumeric and underscore characters. • Must start with a letter. • Can't end with an underscore character. You can't use this component to update a Prism table name, only to create one.
`wd.prism.component.client.id`	The Client ID assigned to the API client by Workday. Access the View API Clients report, then view the API Clients for Integrations tab.
`wd.prism.component.client.secret`	The Client Secret assigned to the API client by Workday.
`wd.prism.component.refresh.token`	The Refresh Token assigned to the API client by Workday. Access the View API Clients report, then view the API Clients for Integrations tab.
`wd.prism.component.operation`	The operation performed on the Prism dataset. Examples: `Replace`, `Append`.

©2026 Workday, Inc. All rights reserved

Parameter	Description
	`Replace` creates a table or replaces one of the same name.
`wd.prism.component.csv.input.varname`	(Optional.) The variable containing the CSV input. Not required if you're adding the CSV data to the message rootpart. Note: Studio doesn't automatically select this parameter for inclusion when you add a **PrismAnalytics** component to an assembly. Select it manually if required or add it later on the component's **Parameters** tab.
`wd.prism.component.table.description`	(Optional.) A description for the table. Can contain up to 255 characters.
`wd.prism.component.table.display.name`	(Optional.) The display name for the table. Can contain up to 255 characters. You can't use this component to update a Prism table display name, only to create one.
`wd.prism.component.table.field.varname`	(Optional.) Defines a Prism table. Write the definition to a variable then pass it to this parameter using vars['variable-name'].

Error Handler Components

Concept: Error Handler Components

Studio provides 3 different **Error Handlers** on the **Palette**:

Name	Function
send-error	Enables you to process the error using standard components and steps. A common pattern is to use **send-error** to report the error to the integration event by invoking the **PutIntegrationMessage** common component.
log-error	Logs the current error to the output log. Includes the error message and the stack trace of the Java exception. Use when there's no need to individually expose errors on the integration event but you want to log the message at the time of the error. Avoid logging large messages as they can degrade performance and might be truncated, leading to loss of diagnostic data.
custom-error-handler	Enables you to supply a Spring bean that uses Java to process the error. Use when error handling requires complex Java code.

You can add an error handler as a child element of a mediation component or as an individual element on the assembly. Error handlers added to mediation components are local. Error handlers on the assembly are global.

©2026 Workday, Inc. All rights reserved

When Workday encounters an error in a Studio integration, it stops normal processing and unwinds the chain of processed components, looking for an error handler to invoke. It always attempts to process errors locally first. If no local error handler is available, responsibility passes to the global error handler. Use local error handlers when you want fine control over what happens when errors occur or need to report the detailed context of what the integration was doing at the time of the error. Think of global error handlers as a failsafe.

If every error handler has fired and an error still isn't marked as handled, Workday processes the message, generates an error from the exception, and marks the assembly as terminated. It describes the integration as Completed with Errors.

Notable default behaviors:

- Local error handlers only process errors that arise in their parent mediation component. However, they can also process errors in downstream components. To enable this behavior, set the **Handle Downstream Errors** property of the mediation component containing the error handler to **true**.
- Workday marks errors as handled when an error handler is invoked. However, if you set an error handler's **Rethrow Error** property to **true**, Workday doesn't clear the error. As a result, it's handled by the next in-scope upstream error handler or global error handler. Use this pattern to report the detail of an error locally but to handle the overall failure or completion at a higher level in the assembly.
- When Workday handles an error, it restarts message processing from the invoked error handler, continuing on the response path towards the element where the error was raised. However, you can specify that processing should instead resume at the element where the error was raised. To enable this behavior, set the **Continue After Error** property of the mediation component containing the error handler to **recover**. Use this behavior in situations where the code invoked by the error handler can correct the error condition, enabling a retry of the failed operation to succeed.

You can exercise greater control over error handlers by adding MVEL condition expressions on their **Properties** view. When condition expressions are present, Workday only invokes the error handler if they all evaluate to true.

You can force error raising using the `context.setError` or `context.SetException` methods in MVEL or Java code. When using the `setError` method, you can configure an error ID that can be detected by conditional expressions on the error handler. Use this technique when you're handling an error locally but want to pass some detail of the error type to an upstream error handler for common handling.

You can also handle errors using **Route** components:

- The **failover-strategy** attempts to use a route, and if an error occurs, executes the failover route, marking the error as handled.
- The **custom-strategy** enables you to supply a Spring bean. Spring beans implement the `RoutingStrategy` interface, which has an `isHandleError` method. Use this method to specify that the strategy handles errors.

Reference: Custom-Error-Handler Component Properties

Common Tab

Property	XML Attribute Name	Description
Error Handler ID	`id`	The unique ID of the **custom-error-handler** component within the assembly.
Rethrow Error	`rethrow-error`	Specifies whether the **custom-error-handler** component passes the error back to any upstream error handlers when it has completed processing. The

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		default value is **false**, meaning that the **custom-error-handler** component processes the error alone, then marks it as having been handled. You can also mark errors as having been handled by calling the `setErrorHandled` method on the `MediationContext` object. Example: `mediationContext.setErrorHandled(tr`
Spring Bean	`ref`	The ID of a Spring bean that implements one of these 2 interface classes: • `OnErrorHandler`. • `ConditionalOnErrorHandler`, which extends `OnErrorHandler` with a Boolean method, `invokeHandleError(MediationCon context)`. If this method returns `true`, Workday invokes the `handleError(MediationContext context)` method. Note: • If you implement the `ConditionalOnErrorHandler` interface, you override any conditional expressions you defined.

Expressions Tab

Property	XML Attribute Name	Description
Condition Expression	`condition-expression`	Enables you to enter MVEL condition expressions directly. If all the expressions evaluate to `true`, or if you don't provide an expression, the error handler is invoked. Otherwise, the error handler isn't invoked, error handling continues along the processing chain, and Workday adds an **Audit Log** entry of type `ERROR_HANDLER_SKIPPED`.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		To add a new expression below any existing rows, click **Add**, then enter the expression. To add a new expression above an existing row, place the cursor in that row, then click **Insert** on the **Add** drop-down menu. Use the buttons to the right of the **Expressions** tab to delete rows or reorder them.

Reference: Log-Error Component Properties

Common Tab

Property	XML Attribute Name	Description
Error Handler ID	`id`	The unique ID of the **log-error** component within the assembly.
Rethrow Error	`rethrow-error`	Specifies whether the **log-error** component passes the error back to any upstream error handlers when it has completed processing. The default value is **false**, meaning that the **log-error** component processes the error alone, then marks it as having been handled. You can also mark errors as having been handled by calling the `setErrorHandled` method on the `MediationContext` object. Example: `mediationContext.setErrorHandled(tr`
Level	`level`	The Log4J logging level. There are 5 options: • **debug** • **info** • **warn** • **error** • **fatal**

Advanced Tab

Property	XML Attribute Name	Description
Category	`category`	Specifies part of the Log4J logger name. The default value is `DEFAULT`.

©2026 Workday, Inc. All rights reserved

Expressions Tab

Property	XML Attribute Name	Description
Condition Expression	`condition-expression`	Enables you to enter MVEL condition expressions directly. If all the expressions evaluate to `true`, or if you don't provide an expression, the error handler is invoked. Otherwise, the error handler isn't invoked, error handling continues along the processing chain, and Workday adds an **Audit Log** entry of type `ERROR_HANDLER_SKIPPED`. To add a new expression below any existing rows, click **Add**, then enter the expression. To add a new expression above an existing row, place the cursor in that row, then click **Insert** on the **Add** drop-down menu. Use the buttons to the right of the **Expressions** tab to delete rows or reorder them.

Reference: Send-Error Component Properties

Common Tab

Property	XML Attribute Name	Description
Error Handler ID	`id`	The unique ID of the **send-error** component within the assembly.
Rethrow Error	`rethrow-error`	Specifies whether the **send-error** component passes the error back to any upstream error handlers when it has completed processing. The default value is **false**, meaning that the **send-error** component processes the error alone, then marks it as having been handled. You can also mark errors as having been handled by calling the `setErrorHandled` method on the `MediationContext` object. MVEL example: `context.setErrorHandled(true)`.
Routes To	`routes-to`	The ID of the assembly element towards which the **send-error** component directs the message.

©2026 Workday, Inc. All rights reserved

Expressions Tab

Property	XML Attribute Name	Description
Condition Expression	`condition-expression`	Enables you to enter MVEL condition expressions directly. If all the expressions evaluate to `true`, or if you don't provide an expression, the error handler is invoked. Otherwise, the error handler isn't invoked, error handling continues along the processing chain, and Workday adds an **Audit Log** entry of type `ERROR_HANDLER_SKIPPED`. To add a new expression below any existing rows, click **Add**, then enter the expression. To add a new expression above an existing row, place the cursor in that row, then click **Insert** on the **Add** drop-down menu. Use the buttons to the right of the **Expressions** tab to delete rows or reorder them.

Transform Steps

Convert Text to XML with a Text Schema File

Prerequisites

Create an assembly containing a **textschema** step. Obtain the sample text that will form the basis of your text schema.

Context

The **textschema** step enables you to convert text to XML using a text schema. To develop a text schema, you must know the structure of the data that you want to transform. The process is considerably simplified by having a small sample document containing representative data. This topic assumes that you have such a sample.

You can use the **textschema** step to convert XML to text or to transform delimited files, but in most cases it's simpler to use the **XTT** step and the **csv-to-xml** step, respectively.

Steps

1. In the **textschema** step's **Properties** view, click **Create Text Schema File**.
2. Select a parent folder for the text schema file and provide a filename. If required, specify the target namespace URI.
3. Studio opens the Text Schema Editor and displays the new text schema in its **Tree** view.

©2026 Workday, Inc. All rights reserved

4. Cut and paste your sample text into the area in the upper right of the Text Schema Editor, replacing `initial-text`.

   Note: To strip the terminating linefeed from input files, add this attribute to the root schema element: `ts:eofStrip="
"`.
5. Begin describing the sample text. Select the first field, then right-click it and select **Create Field**.

   Note: If you don't have a sample of the text you want to convert but know its structure, you can edit the text schema's **Source** view directly.
6. On the **Identity** page of the **Create Field Wizard**, provide a basic description of the field. As you complete the task, consider:

Option	Description
Name	A name for the field.
Occurrence	The options are: • **Default**: The field is mandatory and occurs only once. • **Required**: Same as default. • **Optional**: The field is optional can occur only once. • **Multiple Optional**: The field is optional and can occur multiple times. • **Multiple Required**: The field is mandatory and can occur multiple times.
Start Tag	A regular expression that specifies the text that causes an element to be generated. Note that the text matching this expression doesn't form part of the output value. If you want the matched text to display in the output value, use the `ts:format` option.
End Tag	A regular expression that specifies the text that marks the end of an element. Note that the text matching this expression doesn't form part of the output value. If you want the matched text to display in the output value, use the `ts:format` option.
Truncate	Specifies whether to truncate overflowing data in a fixed-length field or throw an error. The default value is `true`.
Separator	A regular expression that specifies a character that separates repeated fields. Enabled only if you select the **Multiple Optional** or **Multiple Required** occurrence options.
Omit	Specifies whether to omit the field from the output in a **text-to-XML** or **XML-to-text** transformation. The default is no omission.

7. On the **Delimit** page of the **Create Field Wizard**, describe how many characters from the sample text represent a field:

Option	Description
Format	A regular expression that identifies valid patterns of text for a field. Use this expression

©2026 Workday, Inc. All rights reserved

Option	Description
	as an alternative to **Start Tag** and **End Tag** to ensure that all of the matched text displays in the output. You can also use it when dealing with multiply occurring values so it's clear when the end of the sequence occurs.
Fixed Length	Specifies the length of fixed-length fields. Includes quotation marks and escape characters.
Quoted	Indicates whether string values are treated as quoted strings. The options are: • **Default**: • **always**: • **always-including-empty**: • **optional**: • **never**:
Padding	A regular expression that identifies any padding text within fixed-length fields.
Preferred Padding	Specifies the value to be used if the **Padding** expression can match more than 1 possible string.
Align	Specifies the alignment of fixed-length fields that require padding.

8. On the **Interpret** page of the **Create Field Wizard**, describe how many characters from the sample text represent a field:

Option	Description
Type	The field's data type.
Default Value	A default value for the field.
Date Format	Specifies a pattern for dates or datetimes. Supports any pattern allowed by Java's `SimpleDateFormat`.
Date Language	Specifies the date language to use when interpreting dates. Ensures that the name is handled in the appropriate language when the date format uses the names of months. The default value is`EN`.
Two Digit Year	Specifies a value for interpreting 2-digit year dates. The text schema parser interprets 2-digit year dates that occur before that value as occurring in the current century. Example: if you enter `18`, the parser interprets `14` as `2014` and `96` as `1996`.
Number Format	Specifies a pattern for numbers. Supports any pattern allowed by Java's `DecimalFormat`.
Decimal Separator	Specifies the separator used for decimal numbers.

©2026 Workday, Inc. All rights reserved

Option	Description
Grouping Separator	Specifies the thousands grouping separator for numbers.

9. Repeat steps 6-9 for each field in the sample text, adding new elements in the **Tree** view. When you've created all of the required field definitions, delete the remaining `Unparsed` child element.
10. To wrap all of the elements you've created into a complex type, select them, right-click them, and select **Wrap Elements**. Then provide a name and an occurrence value for the wrapper element.
11. In the wrapper element's **Properties** view, select */n* from the **End Tag** drop-down list.
12. The default schema root element is named `File`. On the **Schema** tab of the element's **Properties** view, provide a more meaningful name.
13. In the **Properties** view of the top level `schema` element, the **Root Element** property has the value `tns:File`. Replace `File` with the meaningful name you selected for the root element.
14. To remove any extra spaces from the generated output, select the **Properties** view of the top level `schema` element. Select `+` from the **Padding** drop-down list. Save the schema.

Concept: Transform Steps

Transform steps enable you to perform individual operations that restructure messages as they pass through your integration. Workday Studio provides these **Transform** steps on the **Palette**:

Transform Step	Function	Notes
xslt	Transforms an XML document using an XSL Transformation (XSLT) stylesheet.	• Supports XSLT 2.0. • Compatible with assemblies with a version number up to 2017.05.
XSLT+	Transforms an XML document using an XSL Transformation (XSLT) stylesheet.	• Supports XSLT 3.0 streaming functionality. • Compatible with assemblies with a version number after 2017.05.
fop	Generates PDF and RTF documents from an XML file.	• Invokes the Apache Formatting Objects Processor (FOP) implementation. • Implements an Apache FOP XML configuration file for registering custom fonts.
text-excel	Converts Excel files into comma-separated-value (CSV) text format files. Also converts CSV files into Excel files.	
textschema	Converts text to XML, or XML to text, using a text schema file.	
wrap-soap	Wraps the XML message in a SOAP envelope for forwarding to a SOAP-based web service.	

©2026 Workday, Inc. All rights reserved

Transform Step	Function	Notes
xml-to-java	Unmarshals XML to Java objects using JAXB.	• Implements XPath to extract XML from messages. • Implements JAXB to convert the XML into Java objects. • Inserts converted Java objects into the `MediationContext`.
json-transformer	Takes a JSON document as input and transforms it by mapping values to JSON Paths.	
xml-to-json	Converts an XML document to JSON format.	
json-to-xml	Converts a JSON document to XML format.	
java-to-xml	Marshals Java objects to XML using JAXB.	• Extracts Java objects from the `MediationContext`. • Implements JAXB to convert Java objects to XML. • Implements XPath to insert converted XML into messages.
xml-to-csv	Converts XML documents to CSV format, while streaming the input and output documents, producing a low-memory footprint.	
csv-to-xml	Converts CSV documents to XML format, while streaming the input and output documents, producing a low-memory footprint.	
character-conversion	Performs character-conversion operations on the text content of an incoming message.	

mtable-builder	Creates and stores a `mtable` object in a `MediationContext` property.	• Supports these child elements:
		  • **csv-mtable-reader**
		  • **json-mtable-reader**

mtable-writer	Retrieves a `mtable` object from a `MediationContext` property.	• Supports these child elements:
		  • **csv-mtable-writer**
		  • **json-mtable-writer**

©2026 Workday, Inc. All rights reserved

Transform Step	Function	Notes
	Displays the contents of a `mtable` object in an output message.	
etv	Transforms and validates elements in an XML document in accordance with attributes attached to those elements.	
xtt	Converts XML to text in accordance with attributes defined within the XML.	

Reference: XSLT Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **xslt** mediation step within the assembly.
Input	`input`	Specifies where the **xslt** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **xslt** mediation step directs the output. There are 5 possible values:

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Url	`url`	The location of the XSLT file used by the **xslt** mediation step.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **xslt** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **XSLT** step executes.
Use File Backed Managed Data	`use-file-backed-managed-data`	Specifies whether to stream all output to your local disk once you reach a data threshold.
Messages Property	`messages-property`	The `MediationContext` property that you intend to configure on a `List` object.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Reuse Strategy	`reuse-strategy`	Specifies the XSLT transformer reuse strategy. There are 4 options: • **default**: the `MediationContext` automatically determines your reuse strategy. • **no-reuse**: your server doesn't reuse transformer objects. • **templates**: your server builds a `Templates` object, from which it creates fresh transformer objects. • **transformer-pool**: your server uses a pool of transformers. We only recommend this option if the transformer objects don't cache values. Example: Saxon caches the results of `document()` calls.
Secure Processing	`secure-processing`	Specifies whether to enable processing that could compromise security. If set to **true**, the **xslt** mediation step processes XML securely, setting limits on XML constructs to avoid conditions such as denial-of-service attacks. If set to **false**, the **xslt** mediation step ignores security issues such as limits on XML constructs.
Transformer Factory	`transformer-factory`	The class name of the XSLT `TransformerFactory` used by the **xslt** mediation step. Example: the value for Saxon is `net.sf.saxon.TransformerFactoryImpl`

©2026 Workday, Inc. All rights reserved

Reference: XSLT+ Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **XSLT +** mediation step within the assembly.
Input	`input`	Specifies where the **XSLT+** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **XSLT +** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Url	`url`	The location of the XSLT file used by the **XSLT+** mediation step.
Secure Processing	`secure-processing`	Specifies whether to enable processing that could compromise security. If set to **true**, the **XSLT+** mediation step processes XML securely, setting limits on XML constructs to avoid conditions such as denial-of-service attacks. If set to **false**, the **XSLT+** mediation step ignores security issues such as limits on XML constructs.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **XSLT+** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **XSLT+** step executes.
Messages Property	`messages-property`	The `MediationContext` property that you intend to configure with a `List` object.
Schema Validation Mode	`schema-validation-mode`	Specifies whether to validate the inputs and the outputs, based

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		on the schemas provided, using the Saxon XSLT processor.

Schema URLs Tab

Property	XML Attribute Name	Description
Schema URL	`schema-url`	The location of the schema file that the XSLT+ mediation step uses to validate the output document.

Reference: FOP Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **fop** mediation step within the assembly.
Input	`input`	Specifies where the **fop** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **fop** mediation step directs the output. There are 5 possible values:

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Author	`author`	Specifies an `author` value in the metadata of the document.
Base Url	`base-url`	The base URL that's used to resolve relative URLs. This location is also used for relative font locations, if no `font-base-url` is specified. It can be a relative path to the project's `WSAR-INF` or an absolute URL. Example: `./fop`
Creation Date	`creation-date`	Specifies a `creation date` value in the metadata of the document. Enter the value using the US date format. Example: `5/14/08 4:50 PM`.
Creator	`creator`	Specifies a `creator` value in the metadata of the document.
Keywords	`keywords`	Specifies a `keywords` value in the metadata of the document.
Title	`title`	Specifies a `title` value in the metadata of the document.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
User Password	`user-password`	Enables you to set a user password for PDF and RTF documents. The password enables these document privileges: • `allow-copy-content` • `allow-edit-annotations` • `allow-edit-content` • `allow-print`

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **fop** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **fop** step executes.
Allow Copy Content	`allow-copy-content`	Specifies whether you can copy and paste content from the document when you access it using the `user-password`.
Allow Edit Annotations	`allow-edit-annotations`	Specifies whether you can edit annotations within the document when you access it using the `user-password`.
Allow Edit Content	`allow-edit-content`	Specifies whether you can edit content within the document when you access it using the `user-password`.
Allow Print	`allow-print`	Specifies whether you can print the document when you access it using the `user-password`.
Configuration Xml	`configuration-xml`	The path to the Apache FOP XML configuration file, which must be on your local disk and not a URL. This configuration file is the only way to register custom fonts.
Creation Date Format	`creation-date-format`	The date format used to parse the `creation-date` XML attribute. Supports these values: • `full` • `long`

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• `medium` • `short`
Creation Date Locale Country	`creation-date-locale-country`	The `country` value used to parse the `creation-date` XML attribute.
Creation Date Locale Language	`creation-date-locale-language`	The `language` value used to parse the `creation-date` XML attribute. The value must be a lowercase two-letter ISO-639 language code.
Creation Time Format	`creation-time-format`	The time format used to parse the `creation-date` XML attribute. Supports these values: • `full` • `long` • `medium` • `short`
Font Base Url	`font-base-url`	The base URL that's used to resolve relative font URLs. Example: `./fop/fonts`
Hyphenation Base Url	`hyphenation-base-url`	The base URL that's used to resolve relative URLs to hyphenation-pattern files.
Owner Password	`owner-password`	Enables you to set an owner password for PDF and RTF documents. The password enables these document privileges: • `allow-copy-content` • `allow-edit-annotations` • `allow-edit-content` • `allow-print`
Producer	`producer`	Specifies a `producer` value in the metadata of the document.
Target Resolution	`target-resolution`	The output resolution in dots per inch (DPI) for bitmap images generated by bitmap renderers, such as the TIFF renderer and Apache Batik.

©2026 Workday, Inc. All rights reserved

Reference: Text-Excel Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **text-excel** mediation step within the assembly.
Input	`input`	Specifies where the **text-excel** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **text-excel** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Style	`style`	The conversion style of the **text-excel** mediation step. There are 3 options: • **XLSX/XLS to Text**: converts an Excel file of either XLSX or XLS format to a CSV file. • **Text to XLS**: converts a CSV file to an Excel file in the XLS format. • **Text to XLSX**: converts a CSV file to an Excel file in the XLSX format. Note: This component streams when converting to or from XLSX.
Excel Style	`excel-style`	Specifies whether the **text-excel** mediation step converts an entire Excel workbook or a single Excel sheet.
Excel Sheet Number	`excel-sheet-number`	The target sheet number to convert in an Excel file. If you configure the **Excel Style** property to **workbook**, Workday ignores the **Excel Sheet Number** setting for your **text-excel** mediation step.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **text-excel** step that displays in the consolidated log. You can use this feature to describe the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **text-excel** step executes.
Use Values	`usevalues`	Specifies whether the **text-excel** mediation step uses the underlying values of the spreadsheet, rather than their formatted values as they display on-screen.

Related Information
Reference
Workday 31 What's New Post: Workday Studio

Reference: TextSchema Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **textschema** mediation step within the assembly.
Input	`input`	Specifies where the **textschema** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **textschema** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Url	`url`	The location of the text schema XSD file used by the **textschema** mediation step, given as a relative path or an absolute URL. You can use an MVEL expression to extract the URL from the `MediationContext`. Example: `@{parts[1].xpath('/script/location')}`
Style	`style`	The conversion style of the **textschema** mediation step. There are 2 options: • **text2xml**: converts text to XML. • **xml2text**: converts XML to text.

©2026 Workday, Inc. All rights reserved

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **textschema** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **textschema** step executes.
Use File Backed Managed Data	`use-file-backed-managed-data`	Specifies whether to stream all output to your local disk once you reach a data threshold.

Reference: Wrap-Soap Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **wrap-soap** mediation step within the assembly.
Input	`input`	Specifies where the **wrap-soap** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Output	`output`	Specifies where the **wrap-soap** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **wrap-soap** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **wrap-soap** step executes.
Interface	`interface`	The immediate child of the SOAP Body element.
Soap Action	`soap-action`	The value of the SOAP action header. You must specify this value for SOAP HTTP requests.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Soap Version	`soap-version`	The version of the SOAP envelope.

Reference: XML-to-Java Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **xml-to-java** mediation step within the assembly.
Input	`input`	Specifies where the **xml-to-java** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Property	`property`	The name of the `MediationContext` property into which the **xml-to-java** mediation step inserts the Java object.
Packages	`packages`	The semicolon separated list of Java packages required by the **xml-to-java** mediation step to convert XML to Java objects using JAXB.
Unwrap JAXB Element	`unwrap-jaxbelement`	Specifies whether the **xml-to-java** mediation step stores the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Java object within the JAXB element.
Namespaces	`namespaces`	The namespaces used by your XPath expression.
XPath	`xpath`	The XPath expression that extracts XML from the message to convert it to Java objects.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **xml-to-java** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **xml-to-java** step executes.
XPath Version	`xpath-version`	The XPath version used by the **xml-to-java** mediation step. Example: Version 1.0, Version 2.0, or Version 3.0.

Reference: JSON Transformer Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **json-transformer** mediation step within the assembly.
Input	`input`	Specifies where the **json-transformer** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **json-transformer** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **json-transformer** mediation step that displays in the consolidated log. You can use this feature to

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **json-transformer** step executes.

Mappings Tab

Property	XML Attribute Name	Description
Json Path	`json-path`	Specifies the location on the JSON tree where the associated **Value** is assigned.
Fail No Match	`fail-no-match`	Determines what happens when the specified JSON path doesn't return a node: • **true**: Raise an error. • **fail**: Ignore the missing path.

Reference: XML-to-JSON Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **xml-to-json** mediation step within the assembly.
Input	`input`	Specifies where the **xml-to-json** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **xml-to-json** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Schema URL	`schema-url`	Specifies a relative URL so that the **xml-to-json** mediation step retrieves the XML schema file as a resource within the assembly web service project.
Schema	`schema`	Specifies where the **xml-to-json** step obtains the XML schema. There are 5 possible values: • **message** • **soapbody** • **attachment** • **rootpart** • **variable**

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Ignore Attributes	`ignore-attributes`	Specifies whether the **xml-to-json** step ignores attributes when converting XML documents to JSON.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **xml-to-json** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **xml-to-json** step executes.

Reference: JSON-to-XML Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **json-to-xml** mediation step within the assembly.
Input	`input`	Specifies where the **json-to-xml** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **json-to-xml** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Root Element Name	`root-element-name`	The name of the outermost wrapping tag of the XML document.
Nested Array Name	`nested-array-name`	A name for anonymous nested arrays in JSON input.
Nested Object Name	`nested-object-name`	A name for anonymous nested objects in JSON input.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **json-to-xml** mediation step that displays in the consolidated log. You can use this feature to describe the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **json-to-xml** step executes.

Reference: Java-to-XML Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **java-to-xml** mediation step within the assembly.
Output	`output`	Specifies where the **java-to-xml** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Property	`property`	The name of the `MediationContext` property from which the **java-to-xml** mediation step extracts the Java objects.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Packages	`packages`	The semicolon separated list of Java packages required by the **java-to-xml** mediation step to convert Java objects to XML using JAXB.
Root Element Name	`root-element-name`	The name of the root element used for marshaling Java objects to XML.
Root Element Namespace	`root-element-namespace`	The namespace of the root element used for marshaling Java objects to XML.
Namespaces	`namespaces`	The namespaces used by your XPath expression.
XPath	`xpath`	The XPath expression that inserts XML into your message.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **java-to-xml** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **java-to-xml** step executes.
XPath Version	`xpath-version`	Specifies whether to use XPath version 1.0, 2.0, or 3.0. Enter the value 1 for assembly versions through Workday 11. Enter 2 or 3 for assembly versions Workday 12 onwards. You can enable Workday to set the version dynamically using an MVEL template that inserts a system property value. Example: `@{props.get('the_version')}`.

Reference: XML-to-CSV Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **xml-to-csv** step within the assembly.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Input	`input`	Specifies where the **xml-to-csv** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **xml-to-csv** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *variable*: save the output as a named variable in the `MediationContext`.
Separator	`separator`	The separator character to insert after each data entry in the CSV output file.
Line Separator	`line-separator`	The new line separator character to insert after each data entry in the CSV output file. Supports these values: • `LF` for Linux and Mac OS X and later. • `CR` for Mac OS version 9 or below. • `CRLF` for Microsoft Windows.
Write Header Line	`writeHeaderLine`	Specifies whether the **xml-to-csv** step inserts the names of the sequential child elements from the XML document as comma-separated values in the first line in the output document.
Format	`format`	Specifies whether your CSV output documents are RFC-4180 compliant. You can enter either of these values: • *rfc4180* • *simple* If you enter *rfc4180*, then your CSV output documents will comply with RFC-4180 standards. If you enter *simple*, or leave the field blank, then the **xml-to-csv** step: • Treats apostrophes as alternative quote marks. • Trims initial whitespaces in values. • Accepts any line endings (CR, LF, or CRLF).

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **xml-to-csv** step that displays in the consolidated log. You can use

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **xml-to-csv** step executes.

Reference: CSV-to-XML Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **csv-to-xml** step within the assembly.
Input	`input`	Specifies where the **csv-to-xml** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **csv-to-xml** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Col Names	`colNames`	The names of the columns that display in your XML output file. Enter the value for this property as a comma-separated list of names. Example: `col1`, `col2`, `col3`.
Namespace Prefix	`namespacePrefix`	The namespace prefix that displays in your XML output file.
Namespace URI	`namespaceURI`	The namespace URI that displays on the root element in your XML output file. Example: if you specify a URI of `http://example.workday.com/SupplyChainManagement/DataTypes.xsd` and a namespace prefix of `ns`, the **csv-to-xml** step creates this root element in your XML output file: `<ns:root xmlns:ns="http://example.workday.com/SupplyChainManagement/DataTypes.xsd">`
Root Name	`rootName`	The name of the root element that displays in your XML output file.
Row Name	`rowName`	The names of the rows that display in your XML output file.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Enter the value for this property as a comma-separated list of names. Example: `row1`, `row2`, `row3`.
Separator	`separator`	The separator character used in your CSV input file.
Format	`format`	Specifies whether your CSV input file is RFC-4180 compliant. You can enter either of these values: • *rfc4180* • *simple* If you enter *rfc4180*, then the **csv-to-xml** step assumes that your CSV input file complies with RFC-4180 standards. If you enter *simple*, or leave the field blank, then the **csv-to-xml** step: • Treats apostrophes as alternative quote marks. • Trims initial whitespaces in values. • Accepts any line endings (CR, LF, or CRLF).
Use First Line As Header	`useFirstLineAsHeader`	Specifies whether the **csv-to-xml** step uses the first line of your CSV input file to name the columns that display in your XML output file.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **csv-to-xml** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **csv-to-xml** step executes.

©2026 Workday, Inc. All rights reserved

Reference: Character-Conversion Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **character-conversion** mediation step within the assembly.
Input	`input`	Specifies where the **character-conversion** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **character-conversion** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Remove Accents	`remove-accents`	Specifies whether the **character-conversion** step replaces accented characters with non-accented characters in the output message. Example: if set to `true`, the **character-conversion** step replaces é with e.
Remove Eols	`remove-eols`	Specifies whether the **character-conversion** step removes end-of-line characters in the output message.
Uppercase	`uppercase`	Specifies whether the **character-conversion** step converts all characters to uppercase in the output message.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **character-conversion** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **character-conversion** step executes.
Replace Controls	`replace-controls`	Specifies whether the **character-conversion** step replaces ASCII control characters with spaces in the output message.

©2026 Workday, Inc. All rights reserved

Reference: Mtable-Builder Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **mtable-builder** mediation step within the assembly.
Input	`input`	Specifies where the **mtable-builder** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Mtable Name	`mtable-name`	The name of the `MediationContext` property that stores the `mtable` object.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **mtable-builder** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **mtable-builder** step executes.

©2026 Workday, Inc. All rights reserved

Reference: Mtable-Writer Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **mtable-writer** mediation step within the assembly.
Output	`output`	Specifies where the **mtable-writer** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Mtable Name	`mtable-name`	The name of the `MediationContext` property that stores the `mtable` object.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **mtable-writer** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Execute When	`execute-when`	A condition that determines whether the **mtable-writer** step executes.

Reference: ETV Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **etv** mediation step within the assembly.
Input	`input`	Specifies where the **etv** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **etv** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **etv** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **etv** step executes.
Append Messages	`append-messages`	Specifies whether new messages replace existing messages or are appended to them.
Message Property	`message-property`	The name of the `MediationContext` property that stores messages created by the **etv** mediation step.

Reference: XTT Step Properties

Common Tab

Property	XML Attribute Name	Description
Step Id	`id`	The unique ID of the **xtt** mediation step within the assembly.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Input	`input`	Specifies where the **xtt** mediation step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message. If the message isn't MIME, the entire message is selected. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **xtt** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **xtt** mediation step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **xtt** step executes.
Append Messages	`append-messages`	Specifies whether new messages replace existing messages or are appended to them.
Message Property	`message-property`	The name of the `MediationContext` property that stores messages created by the **xtt** mediation step.

Mtable-Builder Child Elements

Concept: Mtable-Builder Child Elements

The **mtable-builder** step supports these child elements:

Child Element	Description
csv-mtable-reader	Parses CSV content into a `mtable` object.
json-mtable-reader	Parses JSON content into a `mtable` object.

Reference: CSV-Mtable-Reader Properties

Common Tab

Property	XML Attribute Name	Description
Column Names	`column-names`	A comma-separated list of column names that display on your `mtable` object.
Contains Header Row	`contains-header-row`	Indicates whether your `mtable` object contains a header row.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Index Column	`index-column`	The name of the index column that displays on your `mtable` object.
Normalise Column Names	`normalise-column-names`	Indicates whether the column names on your `mtable` object display with spaces. If set to `true`, the column names on your `mtable` object display with spaces. If set to `false`, the column names on your `mtable` object display without spaces.
Separator	`separator`	The delimiter for separating fields in your `mtable` object.
Use Header Row	`use-header-row`	Indicates whether your `mtable` object uses the header row to display column names.

Reference: JSON-Mtable-Reader Properties

Common Tab

Property	XML Attribute Name	Description
Parse Target	`parse-target`	The name of the root JSON object or array that you want to parse into your `mtable` object.
Ignore Id Column	`ignore-id-column`	Specifies whether the ID column from your JSON input data displays in your `mtable` object. If set to `true`, Workday ignores the ID column from your JSON input data in your `mtable` object. If set to `false`, Workday displays the ID column from your JSON input data in your `mtable` object.

Mtable-Writer Child Elements

Concept: Mtable-Writer Child Elements

The **mtable-writer** step supports these child elements:

©2026 Workday, Inc. All rights reserved

Child Element	Description
csv-mtable-writer	Displays the contents of a `mtable` object in CSV format.
json-mtable-writer	Displays the contents of a `mtable` object in CSV format.

Reference: CSV-Mtable-Writer Properties

Common Tab

Property	XML Attribute Name	Description
Column Names	`column-names`	A comma-separated list of column names that display on your `mtable` object.
Line Separator	`line-separator`	The new line separator character in your `mtable` object. Supports these values: • `LF` for Linux and Mac OS X. • `CR` for Mac OS version 9 or below. • `CRLF` for Microsoft Windows.
Separator	`separator`	The delimiter for separating fields in your `mtable` object.
Write Column Names	`write-column-names`	Indicates whether your `mtable` object uses a header row to display column names in the `MediationContext`.

Reference: JSON-Mtable-Writer Properties

Common Tab

Property	XML Attribute Name	Description
Field Name	`field-name`	The name of the JSON field that displays the contents of the `mtable` object.
Write Mtable as Object	`write-mtable-as-object`	Indicates whether single-row mtable objects display as either: • JSON arrays containing 1 object element and multiple fields (True). • JSON objects with multiple fields (False). The default setting is false.

©2026 Workday, Inc. All rights reserved

Logging and Eventing Steps

Concept: Logging and Eventing Steps

Logging steps enable you to create customized log messages for output to different destinations. Workday Studio provides these **Logging & Eventing** steps on the **Palette**:

Name	Function	Notes
log	Creates a customized Log4j message that's written to Workday.	• Press Ctrl+space when inserting an MVEL expression to access content assist.

cloud-log	Enables you to output customized messages about an integration process to a HTML or CSV, which you can send to the endpoint of your choice.	• The **cloud-log** step generates an output file, but doesn't direct it anywhere. Use another step to handle the file's storage or transfer. Example: use a **Store** step to send the output file to your tenant.
		• The output file that the **cloud-log** step produces always includes 5 columns:
		  • Timestamp
		  • Level
		  • Reference ID
		  • Message Summary
		  • Message Details
		  Use the **Extra Columns** tab to add extra information.
		  The maximum size for the output file is 256 MB. If your output exceeds 256 MB, you're notified that the results are truncated.

Reference: Log Step Properties

Message Builder Tab

The **Message Builder** simplifies defining the message content for some assembly steps.

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **log** step within the assembly.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Input	`input`	Specifies where the **log** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Level	`level`	The Log4J logging level. There are 5 options: • **debug** • **info** • **warn** • **error** • **fatal**

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **log** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **log** step executes.
Category	`category`	Part of the logger name. If you don't specify a value, Workday uses `DEFAULT`.

©2026 Workday, Inc. All rights reserved

You can use the `static-file` child element to specify a character-encoding property for an input file. Example:

```
<cc:log-message>
        <cc:static-file character-encoding="UTF-16" input-
file="logfile.html"/>
    </cc:log-message>
```

If you don't specify a `character-encoding` value, Workday uses the character set specified in the `output-mimetype` property. If a character set isn't specified there either, Workday uses UTF-8.

Reference: Cloud-Log Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **cloud-log** step within the assembly.
Variable Name	`variable-name`	The name of the variable that stores the output file. You can give multiple **cloud-log** steps the same variable name to have their messages concatenated into a single report. You can use different variable names to produce separate output files.
Level	`level`	The severity level of the log message. There are 6 options: • **debug** • **info** • **warn** • **error** • **fatal** • **custom** If you select **custom**, you must specify your custom property using an MVEL expression in the adjacent field.
Header	`header`	The text for the cloud-log file header. If you leave this field blank, Studio uses the integration name as the header text. If your assembly has multiple **cloud-log** steps with defined headers, Studio uses the header from the first one in the flow.
Message	`message`	The text of the log message.
Message Details	`message-details`	A description of the log message.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Reference ID	`reference-id`	The Reference ID of a Workday object. Useful for troubleshooting.
Condition	`condition`	An MVEL boolean expression that specifies the circumstances under which Workday logs the message. If you don't specify a condition, Workday always logs the message.
Output File Type	`output-file-type`	The type of file the **cloud-log** step produces. There are 3 options: • **HTML** • **CSV** • **XLSX** The maximum size for the output file is 256 MB. If output exceeds 256 MB, Studio notifies you and truncates the results.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the log message.
Execute When	`execute-when`	A condition that determines whether the **cloud-log** step executes.

Extra Columns Tab

Property	XML Attribute Name	Description
Key	`key`	The identifier for an additional custom message. Unique within 1 log variable.
Label	`label`	The user-facing label for an additional custom message.
Value	`value`	The text of an additional custom message.

Related Information
Reference
Workday 31 What's New Post: Workday Studio

©2026 Workday, Inc. All rights reserved

Message Manipulation Steps

Concept: Message Manipulation Steps

Message manipulation steps enable you to customize the content of messages as they pass through your integration. Workday Studio provides these **Message Manipulation** steps on the **Palette**:

Name	Function	Notes
copy	Extracts individual parts of incoming messages and copies them to different parts of outgoing messages.	
javascript	Implements a JavaScript script or function to manipulate the incoming messages.	
set-headers	Manipulates the header of a mediation message.	
write	Enables you to construct custom data and set it as the content of a mediation message.	• Press Ctrl+space when inserting an MVEL expression to access content assist.

Reference: Copy Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **copy** step within the assembly.
Input	`input`	Specifies where the **copy** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME,

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Input XPath	`input-xpath`	An XPath expression that extracts a matching section of XML input data from a message part or variable in the `MediationContext`.
Output	`output`	Specifies where the **copy** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Output XPath	`output-xpath`	An XPath expression that replaces a matching section of XML output data from a message part or variable in the `MediationContext`.
Append to Output Element	`append-to-output-element`	Specifies whether the **copy** step appends the input XML element to the element that matches the `output-xpath` value.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Namespaces	`namespaces`	The namespaces used by your XPath expression in the format `prefix namespace prefix2 namespace2`. Overrides namespace mappings in the assembly.xml file.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **copy** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **copy** step executes.
Stream XPath	`stream-xpath`	Specifies whether the **copy** step provides streaming support for simple XPath expressions.
XPath Version	`xpath-version`	Specifies whether to use XPath version 1.0 or 2.0. Enter the value 1 for assembly versions through Workday 11, and 2 for assembly versions Workday 12 onwards. Workday ignores this property if the **stream-xpath** value is set to **true**. You can enable Workday to set the version dynamically using an MVEL template that inserts a system property value. Example: `@{props.get('the_version')}`.

Reference: JavaScript Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **javascript** step within the assembly.
Input	`input`	Specifies where the **javascript** step obtains data as input. There are 5 possible values: • *message*: the entire message, including

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		`rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **javascript** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. You must specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Function	`function`	A filename containing a JavaScript function, which must be located in the ws/WSAR-INF directory of your project. Specify a value for either the **Function** or **Script** property, but not both. You can't deploy a **javascript** step with both values configured.
Script	`script`	A filename containing a JavaScript script, which must be located in the ws/WSAR-INF directory of your project. Specify a value for either the **Function** or **Script** property, but not both. You can't deploy a **javascript** step with both values configured.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **javascript** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **javascript** step executes.

Reference: Set-Headers Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **set-headers** step within the assembly.
Output	`output`	Specifies where the **set-headers** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message,

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **set-headers** step. Displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **set-headers** step executes.
Clear All	`clear-all`	Clears all headers from mediation messages.
Clear All Exclude	`clear-all-exclude`	A list of header names that won't be deleted when **Clear All** is set to `true`.

Headers Tab

Add Headers

Property	XML Attribute Name	Description
Name	`name`	The name of the header that you want to add to a mediation message.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Value	`value`	The value of the header that you want to add to a mediation message.

Remove Headers

Property	XML Attribute Name	Description
Name	`name`	The name of the header that you want to remove from a mediation message.

Reference: Write Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **write** step within the assembly.
Input	`input`	Specifies where the **write** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **write** mediation step directs the output. There are 5 possible values: • *message*: the entire message, including

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		`rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment to the message. Specify the attachment index. • *rootpart*: the content of the root part of a MIME message, or if the message isn't MIME, the message itself. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **write** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **write** step executes.

Validation Steps

Concept: Validation Steps

Validation mediation steps enable you to evaluate messages as they pass through your integration. Workday Studio provides these **Validation** steps on the **Palette**:

Name	Function	Notes
eval	Enables you to evaluate expressions directly.	
validate	Validates a mediation message to ensure that it adheres to an XML schema or a Document	• On validation failure, either raises an error or discards the message,

©2026 Workday, Inc. All rights reserved

Name	Function	Notes
	Type Definition (DTD), or that it contains well-formed XML.	depending on the value of the **Filter** property. • You can specify a custom error message. • You can provide a schema or DTD for validating selected elements only, and select those elements from a mediation message by supplying XPath expressions.
validate-exp	Validates an incoming message using one or more MVEL expressions.	• MVEL expressions are defined as a sequence of child elements.
validate-xpath	Validates a message using an XPath expression, which is evaluated as a Boolean.	• Validation is considered successful if the returned XPath expression isn't empty.

Reference: Eval Step Properties

Expressions Tab

Property	XML Attribute Name	Description
Expression	`expression`	Enables you to enter MVEL condition expressions directly. To add a new expression below any existing rows, click **Add**, then enter the expression. To add a new expression above an existing row, place the cursor in that row, then click **Insert** on the **Add** drop-down menu. Use the buttons to the right of the **Expressions** tab to delete rows or reorder them.

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **eval** step within the assembly.

©2026 Workday, Inc. All rights reserved

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **eval** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **eval** step executes.
Language	`language`	Specifies the expression language you want to use. MVEL is the only language currently supported.

Reference: Validate Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **validate** step within the assembly.
Input	`input`	Specifies where the **validate** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Studio selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Schema	`schema`	The location of the schema file that the step uses to validate the message. If you specify a location instead of an MVEL

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		template, Workday uses pooled validators.
Namespaces	`namespaces`	The namespaces used by your XPath expression in the format `prefix namespace prefix2 namespace2`. Overrides namespace mappings in the assembly.xml file.
Xpath	`xpath`	An XPath expression that selects elements for validation.
Failure Message	`failure-message`	A custom error message that overrides the default error message when a validation error occurs. Error messages display only when you set the **Filter** property to **false**.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **validate** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **validate** step executes.
Mode	`mode`	The type of validation performed. The options are: • **schema** : validates against a W3C XML schema. • **dtd**: validates against a Document Type Definition. • **well-formed**: verifies that the XML is well-formed.
DTD	`dtd`	The location of the DTD file that the step uses to validate the message.
Replace Message	`replace-message`	Indicates whether to use the custom error message set as the **Failure Message** property when validation fails.
XPath Version	`xpath-version`	Specifies whether to use XPath version 1.0 or 2.0. Enter the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		value 1 for assembly versions through Workday 11, and 2 for assembly versions Workday 12 onwards. Workday ignores this property if the **stream-xpath** value is set to **true**. You can enable Workday to set the version dynamically using an MVEL template that inserts a system property value. Example: `@{props.get('the_version')}`.
Filter	`filter`	Specifies whether to interrupt mediation message processing when validation fails. The default value is **false**, meaning Studio passes validation errors to error-handlers defined within the component. If set to **true**, Studio executes response steps for the component but discards the message and invokes no further steps. It doesn't raise an error.

Reference: Validate-Exp Step Properties

Expressions Tab

Property	XML Attribute Name	Description
Value	`expression`	Enables you to enter MVEL condition expressions directly. To add a new expression below any existing rows, click **Add**, then enter the expression. To add a new expression above an existing row, place the cursor in that row, then click **Insert** on the **Add** drop-down menu. Use the buttons to the right of the **Expressions** tab to delete rows or reorder them.
Failure Message	`failure-message`	Text to add to the exception message when a validation expression fails.
Error Number	`error-number`	An error number to add to the exception message returned when a validation expression fails.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Replace	`replace`	Indicates whether to replace the failure message returned by Workday with the text specified in the **Failure Message** property.

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **validate-exp** step within the assembly.
Filter	`filter`	Specifies whether to interrupt mediation message processing when validation fails. The default value is **false**, meaning Workday passes validation errors to error-handlers defined within the component. If set to **true**, Workday executes response steps for the component but discards the message and invokes no further steps. It doesn't raise an error.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **validate-exp** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **validate-exp** step executes.

Reference: Validate-Xpath Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **validate-xpath** step within the assembly.
Input	`input`	Specifies where the **validate-xpath** step obtains data as input. There are 5 possible values:

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Filter	`filter`	Specifies whether to interrupt mediation message processing when validation fails. The default value is **false**, meaning Workday passes validation errors to error-handlers defined within the component. If set to **true**, Workday executes response steps for the component but discards the message and invokes no further steps. It doesn't raise an error.
Namespaces	`namespaces`	The namespaces used by your XPath expression in the format `prefix namespace prefix2 namespace2`. Overrides namespace mappings in the assembly.xml file.
Xpath	`xpath`	An XPath expression that selects elements for validation.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **validate-xpath** step that displays in the consolidated log. You can use this feature to describe the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **validate-xpath** step executes.
XPath Version	`xpath-version`	Specifies whether to use XPath version 1.0 or 2.0. Enter the value 1 for assembly versions through Workday 11, and 2 for assembly versions Workday 12 onwards. Workday ignores this property if the **stream-xpath** value is set to **true**. You can enable Workday to set the version dynamically using an MVEL template that inserts a system property value. Example: `@{props.get('the_version')}`.

Encoding Steps

Concept: Encoding Steps

Encoding steps enable you to encode, decode, compress, and decompress the contents of the `MediationContext`.

Workday Studio provides these **Encoding** steps on the **Palette**:

Name	Function	Notes
pgp-encrypt	Retrieves a public key and encrypts outbound data using PGP.	• To encrypt messages using keys stored in Workday, configure a delivery service for your integration. Studio integrations don't have access to keys stored in Workday. • You can also reference a local PGP key using the file:// protocol, then deploy it with the integration. However, the delivery service method is best practice for Studio integrations.
pgp-decrypt	Retrieves a private key and decrypts inbound data that was encrypted using PGP.	

©2026 Workday, Inc. All rights reserved

Name	Function	Notes
base64-decode	Decodes a Base64 encoding of attachment data to binary format.	
base64-encode	Encodes binary attachment data in Base64-encoding format.	
zip	Transforms the root part and attachments from the `MediationContext` into a ZIP file.	
unzip	Extracts the contents of a ZIP file.	• If the ZIP contains more than 1 file, a multipart structure like the mediation message is preferred. Example: if you use the default output option of *message*, the first file in the ZIP becomes the root part and the other files in the ZIP become attachments.
compress	Transforms a single file in the `MediationContext` to a GZIP or BZIP2 file.	• Unlike the ZIP step, this step doesn't support file archiving. Use it to compress single files.
decompress	Extracts the contents of a single GZIP or BZIP2 file.	

Concept: PGP Filenames

As a security measure, when a plaintext file is encrypted with PGP, the encrypted file can be given a filename that's unrelated to the original, giving no hint of its contents. However, the original filename isn't discarded. It's embedded in the encrypted file for later retrieval by the recipient's decrypting system. Example: you encrypt a file named myPlaintext.txt. The resulting file is called myEncryption.pgp. Part of its encrypted content is the original filename. When your recipient decrypts it, their system names the resulting file myPlaintext.txt.

Workday Delivery and Retrieval Services

The Workday Delivery and Retrieval Services follow the typical PGP file-naming practice when possible.

The Workday Delivery Service embeds the plaintext filename in the outgoing PGP file so it can be recovered during decryption.

When the Workday Retrieval Service decrypts an incoming PGP file attached to an integration event, it uses the embedded plaintext filename to name the decrypted file.

Sometimes, however, an incoming PGP file has no embedded plaintext filename. If so, the Workday Retrieval Service bases the name of the decrypted file on the name of the PGP file:

- If the PGP filename ends with .pgp, the Retrieval Service removes that suffix from the decrypted filename. Example: the PGP file myFile.csv.pgp is decrypted as myFile.csv, the PGP file myFile.pgp is decrypted as myFile.
- If the PGP filename ends with .xlsx, the Retrieval Service leaves it unchanged. Example: the PGP file myFile.xslx is decyrpted as myFile.xslx.

©2026 Workday, Inc. All rights reserved

- In all other cases, the Retrieval Service adds the suffix .plaintext. Example: the PGP file myFile.doc is decrypted as myFile.plaintext.

Workday Studio

When you use Studio's **pgp-encrypt** step to encrypt a message, you can specify the plaintext filename using the **Decrypted Filename** step property. Studio then embeds that filename in the encrypted file for subsequent retrieval in the normal manner.

Likewise, Studio's **pgp-decrypt** step uses any embedded plaintext filename to name the decrypted file. You can use the **Decrypted Filename** step property to provide a filename for situations where none is embedded.

Troubleshooting

Problems can arise when a remote party encrypting a file unintentionally sets the embedded plaintext filename to some value other than that required by a recipient system. Because the PGP filename is entirely unrelated to the plaintext filename, there is no way to deduce the latter. The recipient has no options in this scenario.

Example: a Retrieval Service is configured to retrieve and decrypt a file. When run, it collects a PGP encrypted file named ExternalStudents.pgp, but a file named _CONSOLE is attached to the integration event for processing by an integration. In this case, the remote system has set _CONSOLE as the embedded plaintext filename. The result is the file is identified to Workday integrations as _CONSOLE.

The only way to prevent such problems is to ensure that senders of PGP files understand how to correctly embed the required plaintext filename.

Reference: PGP-Encrypt Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **pgp-encrypt** step within the assembly.
Input	`input`	Specifies where the **pgp-encrypt** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **pgp-encrypt** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Ascii Armored	`ascii-armored`	Indicates whether the **pgp-encrypt** step generates PGP data in ASCII-armored format (a PGP version of Base64) or in pure binary. ASCII-armored formatting produces email-friendly text, including a header and a footer.
Certificate	`certificate`	The certificate used for encryption, provided by 1 of these types of URL: • An attachment URL for retrieving the public key from the tenant. Must follow the pattern

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		`attachment:certificate/WIDvalue`, where `WIDvalue` is the WID of the public key stored in the tenant. Example: you can configure the reference using an integration attachment in the form `attachment:certificate/@{intsys.getAttributeReferenceDEncryption Key', 'WID')}` where PGP `Encryption Key` is the name of the integration attribute. • An mctx URL for retrieving the public key from the context variable where the integration stored it. Example: `mctx:vars/MyPublicKey` where `MyPublicKey` is the name of the context variable. • The filename of a public key stored in the assembly's WSAR-INF folder or in one of its subfolders. Examples: `PublicKey.asc` or `/keys/PublicKey.asc`.
Containing Integrity Check	`containing-integrity-check`	Indicates whether the **pgp-encrypt** step generates a message signature for integrity checking by the receiver.
Decrypted Filename	`decrypted-filename`	A name for the decrypted file. Example: `yourname.plaintext.mail`. Workday includes the specified value in the hash string.
Pgp26x Compatible	`pgp26x-compatible`	Indicates whether the **pgp-encrypt** step generates encrypted data that is backward compatible with the older pgp26x version.
Private Key Passphrase	`private-key-passphrase`	Specifies the passphrase for the private key.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Signing Private Key	`signing-private-key`	The private key used to sign messages as a filename, using the `file://` protocol. Note: Best practice for Workday Studio integrations is to use a delivery service rather than the **pgp-encrypt** step to perform encryption.
Textmode	`textmode`	Specify `true` to indicate that the data to be encrypted is in text format, containing <CR><LF> line endings. Note: If you specify `true` for data that doesn't contain <CR><LF> line endings, Workday reports an error.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **pgp-encrypt** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it's operating.
Execute When	`execute-when`	A condition that determines whether the **pgp-encrypt** step executes.

Related Information
Reference
Workday 31 What's New Post: Workday Studio

Reference: PGP-Decrypt Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **pgp-decrypt** step within the assembly.
Input	`input`	Specifies where the **pgp-decrypt** step obtains data as input. There are 5 possible values: • *message*: the entire message, including

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		`rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **pgp-decrypt** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Private Key	`private-key`	The filename or URI for the Workday private key. Use the `file://` protocol to specify it as a filename.
Passphrase	`passphrase`	Specifies the passphrase for the private key.
Decrypted Filename	`decrypted-filename`	Sets the `content-disposition` attribute on the output of the **pgp-decrypt** step or, if the output is a variable, the filename. If the **pgp-encrypt** step specifies a **decrypted-filename** value, however, that value takes precedence.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **pgp-decrypt** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **pgp-decrypt** step executes.

Reference: Base64-Decode Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **base64-decode** step within the assembly.
Input	`input`	Specifies where the **base64-decode** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **base64-decode** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **base64-decode** step that displays in

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **base64-decode** step executes.

Reference: Base64-Encode Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **base64-encode** step within the assembly.
Input	`input`	Specifies where the **base64-encode** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **base64-encode** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **base64-encode** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **base64-encode** step executes.

Reference: Zip Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **zip** step within the assembly.
Input	`input`	Specifies where the **zip** step obtains data as input. There are 5 possible values: • *message*: the entire message, including

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		`rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **zip** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Format	`format`	The format of the zipped file. The options are: • *zip* • *tar* Note: The TAR format supports archiving but not compression. To compress TAR files, use either of these commands: • gzip, which replaces the .tar file extension with .tar.gz or .tgz • bzip2, which replaces the .tar file extension with .tar.bz2 You can compress and decompress TAR archives produced in the **zip** step using the **compress** and **decompress** steps, respectively.
Level	`level`	The level of compression in the range zero to 9, with 9 being maximum compression. The default value is -1.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **zip** step. Displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **zip** step executes.
Comment	`comment`	A comment that's embedded in the ZIP file. The comment is an arbitrary ASCII string of up to 65,535 bytes.

©2026 Workday, Inc. All rights reserved

Reference: Unzip Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **unzip** step within the assembly.
Input	`input`	Specifies where the **unzip** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **unzip** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Format	`format`	The format of the zipped file. The options are: • *zip* • *tar* Note: The TAR format supports archiving but not compression. To compress TAR files, use either of these commands: • gzip, which replaces the .tar file extension with .tar.gz or .tgz • bzip2, which replaces the .tar file extension with .tar.bz2 You can compress and decompress TAR archives produced in the **zip** step using the **compress** and **decompress** steps, respectively.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **unzip** `MediationContext`.
Execute When	`execute-when`	A condition that determines whether the **unzip** step executes.
Filename	`filename`	The name of the file created when the content unzips.

©2026 Workday, Inc. All rights reserved

Reference: Compress Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **compress** step within the assembly.
Input	`input`	Specifies where the **compress** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **compress** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Format	`format`	The format of the zipped file. The options are *gzip* and *bzip2*. The default format is *gzip*.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **compress** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **compress** step executes.
Filename	`filename`	The name of the file created when the content unzips.

Reference: Decompress Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **decompress** step within the assembly.
Input	`input`	Specifies where the **decompress** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **decompress** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Format	`format`	The format of the zipped file. The options are *gzip* and *bzip2*. The default format is *gzip*.

©2026 Workday, Inc. All rights reserved

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **decompress** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **decompress** step executes.
Filename	`filename`	The name of the file created when the content unzips.

Other Steps

Concept: Other Steps

Steps enable you to perform individual operations on messages as they pass through your integration. Workday Studio provides these **Other** steps on the **Palette**:

Name	Function
custom	Enables you to apply custom code written using Spring beans to incoming messages.
set-dynamic-endpoint	Configures WS-Addressing (WSA) endpoint information for an out-transport.
store	Creates a persistent copy of a document that's available to other users and/or can be viewed as an attachment to the integration event. The individual and cumulative message size limit for store steps is 5 GiB (compressed). Workday terminates integrations that exceed these limits. Where possible, use splitters or streaming to break up large files in your integrations.
retrieve	Retrieves a persistent copy of a document that was previously stored by a **store** step.
xmldiff	Compares 2 XML input documents.
enqueue-message	Organizes messages into a queue.

©2026 Workday, Inc. All rights reserved

Reference: Custom Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	Specifies the unique ID of the **custom** step within the assembly.
Input	`input`	Specifies where the **custom** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **custom** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Method Name	`method-name`	The name of the Java method within your custom Java class.
Spring Bean	`ref`	The custom Spring bean Java class implemented by the **custom** step.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **custom** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **custom** step executes.

Reference: Set-Dynamic-Endpoint Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **set-dynamic-endpoint** step within the assembly.
Endpoint	`endpoint`	The address of the dynamic endpoint.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **set-dynamic-endpoint** step that displays in the consolidated log. You can use this feature to

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **set-dynamic-endpoint** step executes.

Reference: Store Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **store** step within the assembly.
Input	`input`	Specifies where the **store** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **store** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`. The **store** step stores a message's mimetype along with the message itself. Example: *text/plain*. When you retrieve the message with a **retrieve** step, Workday automatically marks it with the stored mimetype so later steps can process it correctly. The individual and cumulative size limit for store steps is 5 GiB (compressed). Workday terminates integrations that exceed these limits. Where possible, use splitters or streaming to break up large files in your integrations.
Collection	`collection`	The name of the collection that stores your document.
Expires In	`expiresIn`	The period of time before your document expires. Specify the value in XSD duration format. Example: to indicate 1 year, 3 months, 5 days, 8 hours, 45 minutes, and 10 seconds, enter `P1Y3M5DT8H45M10S`.
Summary	`summary`	A summary of your document. Displays in your Workday tenant.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Title	`title`	The title of your document.
Create Document Reference	`createDocumentReference`	Specifies whether the **store** step automatically calls Workday to create a document reference.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **store** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **store** step executes.
Content Disposition	`contentDisposition`	The content-disposition header for your document. Overrides the **title** property value, if specified.
Blobitory URI	`blobitoryURI`	The URI of the `cc-blobitory` service implemented in your Workday tenant. Example: `https://machine-name.workday.com:8080/ccx/cc-blobitory`.
Expires On	`expiresOn`	The date and time at which your document expires. Overrides the **Expires In** property value, if specified. Enter the value for this property in `CCYY-MM-DDThh:mm:ssZ` format. Example: to indicate 11.59 p.m. on December 31, 2009, enter `2009-12-31T23:59:00Z`.
Schema	`schema`	The URI for the schema that references your document. There are 2 possible values: • **http://www.w3.org/2005/Atom** • **urn:com.workday/bsvc/blob**
Entry ID	`entryID`	Specifies an Entry ID for each document. Must be unique in your tenant. If you don't

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		specify an Entry ID value, Workday generates one. If you specify one that already exists, Workday overwrites the document, provided it's mutable. You can't reuse an Entry ID if the existing document is immutable. When updating documents using the REST API: • Use the PUT HTTP method for documents with predefined Entry IDs. • Use the POST HTTP method when the Entry ID isn't specified or evaluates to null or an empty string.
Is Attachment	`isAttachment`	Specifies whether your document is an OMS attachment.
Immutable	`immutable`	Specifies whether you can modify your document after you create it.
Assign Owner	`assignOwner`	Specifies whether Workday assigns you as the owner of your document.
Document Reference Description	`documentReferenceDescription`	A description of your document. Displays in your Workday tenant.

Reference: Retrieve Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	Specifies the unique ID of the **retrieve** step within the assembly.
Output	`output`	Specifies where the **retrieve** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`. When you configure a Document Retrieval Service on a Business Process, Workday uses the string you enter in the **Custom Content Type** field as the mimetype for the document stored by that service. When you retrieve the document with a **retrieve** step, either directly or through a document iterator or document accessor helper object, Workday automatically marks it with the stored mimetype so later steps can process it correctly. If you don’t enter a **Custom Content Type**, Workday uses UTF-8 encoding. To avoid errors, always enter the appropriate **Custom Content Type** for data that isn't UTF-8 encoded.
Collection	`collection`	The name of the collection that stores your document.
Entry	`entry`	The identifier of the document in your collection. Retrievable from the `MediationContext` using an MVEL expression. Example: `@{parts[0].xpath('//blob:entry', 'blob`

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		`urn:com.workday/bsvc/blob')}`
Password	`password`	The password credentials for accessing your document.
Username	`username`	The username credentials for accessing your document.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **retrieve** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **retrieve** step executes.

Reference: Xmldiff Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	Specifies the unique ID of the **xmldiff** step within the assembly.
First Input	`first-input`	Specifies the first data input source for the **xmldiff** step. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME,

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Second Input	`second-input`	Specifies the second data input source for the **xmldiff** step. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **xmldiff** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Copy To Output	`copy-to-output`	Specifies whether the **xmldiff** step copies the output to the mediation message or a variable. If set to **true**, the **xmldiff** step copies the output to the mediation message. If set to **false**, the **xmldiff** step copies the output to a variable.
Ignore Attributes	`ignore-attributes`	Specifies whether the **xmldiff** step ignores attributes when comparing 2 XML documents.
Namespaces	`namespaces`	Namespace override definitions for XPath expressions implemented by the **xmldiff** step.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **xmldiff** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **xmldiff** step executes.
XPath Version	`xpath-version`	Specifies whether to use XPath version 1.0 or 2.0. Enter the value 1 for assembly versions through Workday 11, and 2 for assembly versions Workday 12 onwards. Workday ignores this property if the **stream-xpath** value is set to **true**. You can enable Workday to set the version dynamically using an MVEL template

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		that inserts a system property value. Example: `@{props.get('the_version')}`.

Definitions Tab

Property	XML Attribute Name	Description
Name	`name`	Specifies the name of the `sub-diff` to use when looking up results of the **xmldiff** step.
Ignore Deletions	`ignore-deletions`	Specifies whether the **xmldiff** step ignores deletions in the output.
Different Nodes Only	`result-document-different-nodes-only`	Specifies which nodes display in the output document for the **xmldiff** step. If set to `true`, only top-level nodes display in the output document. If set to `false`, all child nodes display in the output document.
Result Document Inclusion	`result-document-inclusion`	Specifies whether the **xmldiff** step records differences between nodes in the output document. There are 3 possible values: • *Always*: the selected node and its children always display in the result document. • *Different*: the selected node and its children only display in the result document if differences are found. • *Never*: the selected node and its children never display in the result document.
XPath	`xpath`	The XPath expression implemented by the **xmldiff** step to find nodes in your input document.

Matcher XPath Tab

On the **Matcher XPath** tab, you can add XPath expressions that locate nonsequential matching nodes in your input documents.

©2026 Workday, Inc. All rights reserved

Reference: Enqueue-Message Step Properties

Common Tab

Property	XML Attribute Name	Description
Step ID	`id`	The unique ID of the **enqueue-message** step within the assembly.
Input	`input`	Specifies where the **enqueue-message** step obtains data as input. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. Only applicable when the input contains a SOAP message. • *attachment*: a MIME attachment within the message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. • *variable*: use a named variable from the `MediationContext`.
Output	`output`	Specifies where the **enqueue-message** step directs the output. There are 5 possible values: • *message*: the entire message, including `rootpart` and attachments. This value creates a new message, effectively removing any attachments. • *soapbody*: the content of the SOAP body from the `rootpart` of the message. • *attachment*: a MIME attachment within the

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		message. Specify the attachment index. • *rootpart*: the content of the `rootpart` of a MIME message. If the message isn't MIME, Workday selects the entire message. This value preserves any attachments on the original message. • *variable*: save the output as a named variable in the `MediationContext`.
Content Type	`contentType`	Overrides the content type of your output message, which is text/xml by default.
Queue Name	`queueName`	The name of the queue where the **enqueue-message** step directs the output.
File Name	`fileName`	Overrides the filename in the header of the mediation message.

Advanced Tab

Property	XML Attribute Name	Description
Description	`description`	A description of the **enqueue-message** step that displays in the consolidated log. You can use this feature to describe the step's function and the data on which it is operating.
Execute When	`execute-when`	A condition that determines whether the **enqueue-message** step executes.

Assembly Modules

Create Assembly Modules

Prerequisites

Create an assembly project.

Context

Workday Studio assembly modules provide parameterized templates for building assemblies. They enable you to create and reuse preconfigured sets of components and steps that perform particular tasks.

©2026 Workday, Inc. All rights reserved

Steps

1. Add the elements that you want to include in your assembly module and configure their properties as required. You can create a module from a functionally complete assembly or from a smaller set of components and steps. Users can adjust parameters when they implement the module.
2. Save your assembly.
3. In the **Project Explorer**, right-click the assembly project, then select **Create Assembly Module...**.
4. Provide a name and description for the module. The description displays as the tooltip for the module.
5. Select an icon for the module. You can use the default icon or upload one of your own.
6. Studio organizes module parameters into collections called pages. A parameter must belong to a page. Each page has a name and a description. Each parameter has a name, a label (which displays in the **Configure Module** wizard), a description, a type, and a default value. There are 6 parameter types:

Type	Description	Example
String	A text value.	`"HelloWorld"`
XMLRequest	An XML literal for a request message. Module users specify a value for the Web service request using either the **Create Literal XML** or **Create Workday Web Service Request** wizard.	`<?xml version="1.0" encoding="UTF-8"? > <env:Envelope xmlns:env="http:// schemas.xmlsoap.org/ soap/envelope/" xmlns:xsd="http:// www.w3.org/2001/ XMLSchema"> <env:Body> <wd:Get_Workers_Request xmlns:wd="urn:com.workday/ bsvc" wd:version="v23.2"> <wd:Request_References wd:Skip_Non_Existing_Instances="t <wd:Worker_Reference> <wd:ID wd:type="Employee_ID">abcdef</ wd:ID> </ wd:Worker_Reference> </ wd:Request_References> </ wd:Get_Workers_Request> </env:Body> </ env:Envelope>`
Prompt	You can use Class Report Fields (CRFs) to provide a prompt for launch parameters of Workday objects of a particular type. To specify a CRF Reference, click **Select reference id**. Note: To specify a CRF reference, you must connect to a Workday tenant.	`fbbe44a681251000039acc2ff657000a`

©2026 Workday, Inc. All rights reserved

Type	Description	Example
File	A filename. You can specify the file extension as a `data` value.	`*.xslt`
Enum	A single literal value or multiple values separated by a comma or space.	`code1 code2 code3`
Boolean	A Boolean value.	`true`

7. When you finish adding parameters and save the module, Studio opens its XMI file in the XML editor. The file contains a `params` element for each parameter that you specified. The `assemblyText` and `diagramText` elements contain the assembly XML and diagram details, encapsulated within `<![CDATA[]]>` tags. You can edit the XMI file directly.
8. Close the XMI file.

Add Assembly Modules to a Project

Prerequisites

Create an assembly project or open an existing assembly project.

Context

Workday Studio assembly modules provide parameterized templates for building assemblies. They enable you to create and reuse preconfigured sets of components and steps that perform particular tasks.

Steps

1. Drag the module from the **Modules** section of the **Palette** to the Assembly Editor. Studio opens the **Configure Module** wizard.

   Note: The **Module Library** displays only those assembly modules that you select on the **Window > Preferences > Workday > Assembly Editor > Modules** preference page.
2. Configure the module's parameters.
3. If the assembly module contains a parameter of type `XMLRequest`, click either **Create Literal XML** or **Create Workday Web Service Request** to specify a value for the Web service request. As you complete this task, consider:

Wizard	Operations	Notes
Create Literal XML	`Get`, `Put`	• Can be used for any Web service. • Select the Web service name and Web service version values for the request from the lists below the text pane.
Create Workday Web Service Request	`Get`	• Use for Workday Web services only. • Studio populates Web service name and version values automatically when you select an operation.

©2026 Workday, Inc. All rights reserved

4. Studio displays a preview of the assembly elements that the module contains. When you complete the wizard, Studio adds those elements to the assembly.

Concept: Assembly Modules

Workday Studio assembly modules provide parameterized templates for building assemblies. They enable you to create and reuse preconfigured sets of components and steps that perform particular tasks. You can create a module from an entire assembly or from a segment of one.

An assembly that you build using a module is no different from an assembly that you build from scratch. They look and behave identically.

Studio saves a module as a template in the XML Metadata Interchange (XMI) format, with an xmi file extension. Access your saved modules from the **Module Library** on the Studio **Palette**. To enable other developers to access your modules, save them in a shared location.

Sharing tried-and-tested assembly modules helps speed up and reduce the cost of integration development.

The **Module Library** displays only those assembly modules that you select on the **Window > Preferences > Workday > Assembly Editor > Modules** preference page.

Subassemblies

Create Subassemblies

Prerequisites

Create an assembly.

Context

A subassembly is a processing chain whose first component is a **local-in** transport. You invoke it with a **local-out** transport, either from within the same assembly or from a separate assembly.

Steps

1. Add and configure the **local-in** transport that defines the beginning of the subassembly.
    a) Add a **local-in** transport at the beginning of the processing chain that you want to turn into a subassembly.
    b) On the **Common** and **Advanced** tabs of its **Properties** view, configure the transport's properties. You can provide an icon for the subassembly and text that displays as its tooltip.
    c) On the **Parameters** tab, define parameters that the subassembly requires as input. You can configure these parameters on the **local-out** transport that calls the subassembly. Studio stores them as `MediationContext` properties.
    d) On the **Out Parameters** tab, define the `MediationContext` properties that the subassembly sets.
    e) Save the assembly.

©2026 Workday, Inc. All rights reserved

2. Add and configure the **local-out** transport that invokes the subassembly.
    a) Add a **local-out** transport to the assembly that invokes the subassembly.
    b) On the **Common** and **Advanced** tabs of its **Properties** view, configure its properties. Specify the subassembly **local-in** as the **endpoint** property. Use the format `vm://assembly-name/local-out-ID`, or click **Select local-in sub-assembly**.
    c) On the **Parameters** tab, set `MediationContext` properties. You can create new parameters by clicking **Create 'set' element**. You can also select any that parameters that you define in the **local-in** transport and assign them values by clicking **Select Parameter**.

       Note: In the **Select Parameters** wizard, an asterisk indicates that the property is required. A question mark indicates that usage is dynamic.
    d) Save the assembly.

Concept: Subassemblies

A subassembly is a processing chain whose first component is a **local-in** transport. The **local-in** defines an entry point to the processing chain, enabling it to be called by a **local-out** transport in the same assembly or in another assembly. You can include several subassemblies and **local-out** transports within the same assembly.

Note: You can't deploy subassemblies to Workday on their own because they don't begin with a **workday-in** transport.

Because you can invoke them at will, subassemblies enable you to encapsulate processing logic for reuse. You can think of them as programming subroutines. The default ID for a **local-in** transport is `SubRoutine`. The default ID for a **local-out** transport is `CallSubRoutine`.

Subassemblies have parameters. You specify them on the **local-in** transport and assign them values on the **local-out**. These parameters define properties in the `MediationContext`.

When you save a subassembly, Workday Studio automatically adds a **local-out** transport that invokes it to the **Workspace Components** category on the **Palette**. This **local-out** transport uses the custom icon from the subassembly's **local-in** transport, if you supplied one. Otherwise it uses a question mark icon.

The **Palette** always displays public subassemblies. It displays private subassemblies only if they occur in the assembly. Studio provides several prepackaged subassemblies in the **Common Components** category on the **Palette**.

To locate the **local-in** transport that a **local-out** references, right-click the **local-out** and select **Declaration in Workspace**.

To locate the **local-out** transports that are referenced by a **local-in**, right-click the **local-in** and select **References in Workspace**.

Related Information
Concepts
Concept: Common Components

Custom Assembly Components

Create Custom Java Classes and Spring Beans

Prerequisites

Create an assembly project.

Context

You can create 4 custom assembly elements from the **Palette** in Workday Studio:

©2026 Workday, Inc. All rights reserved

- **custom-mediation** component
- **custom** step
- **custom-out** transport
- **custom-error-handler**

A fifth custom element, custom routing strategy, is available from the **Route** component's context toolbar.

Steps

1. Open an assembly in the Assembly Editor.
2. Add one of these custom elements to the assembly:
    - **custom-mediation** component
    - **custom** step
    - **custom-out** transport
    - **custom-error-handler**
3. Configure the custom element's properties in its **Properties** view.
4. Click **Create Spring Bean Properties and Java Class**, to the right of the **Spring Bean** field. The properties that Studio displays vary slightly depending on the custom element type. As you complete the task, consider:

Option	Description
Id	Provide a unique Spring bean identifier.
Scope	• Select *singleton* to return the same bean instance with each reference. • Select *prototype* to return a different bean instance with each reference.
Package	Specify the Java package.
Name	Specify the class name.
Input Type	Select the input type to pass to the Java method. You don't need to specify implementation details for accessing the message part and converting it to the appropriate type. Workday inspects the method's first parameter and converts the message part to the same type.
Output Type	Select the method's output (return) type.
Method Name	Enter the name of the Java method that you want to define within the Java class. The default value is `process`.
Include argument for dynamic endpoint	Specifies whether to include an argument for a dynamic endpoint.
Annotate Java class to appear in palette	Select this check box to display the custom element on the **Palette**. This option adds an annotation to the generated Java class, which is located in the **src** folder of your assembly project folder.
Tooltip	Enter the text that displays as the custom element's description.

5. Save the assembly.

©2026 Workday, Inc. All rights reserved

Result

Studio generates the Java class representing the custom element within the package specified. It also adds a reference to the new custom Spring bean at the bottom of the assembly.xml file.

Next Steps

Add the required code to your custom Java class.

Concept: Custom Assembly Elements

You can create 5 types of custom assembly elements in Workday Studio:

Custom Element	Procedure
Mediation component	Add a **custom-mediation** component from the **Components** category on the **Palette**.
Mediation step	Add a **custom** step from the **Other** category on the **Palette**.
Out transport	Add a **custom-out** transport from the **Out Transports** category on the **Palette**.
Error handler	Add a **custom-error-handler** component from the **Error Handlers** category on the **Palette**.
Routing strategy	Select **Create custom-strategy** from a **Route** component's context toolbar.

In each case, the custom element consists of 3 parts:

- The underlying Java implementation, which defines the classes and methods used by the element.
- The Spring bean configuration, which tells Workday how to create instances of the Java classes.
- The assembly component configuration, which places your custom element on the graphical assembly view.

  Custom assembly elements can access message parts in the normal way when processing incoming data. You can also specify the message part where Workday should direct the custom element's output.

To share a custom element, export the Java package as a binary JAR. Other developers can then add the JAR to their project's classpath.

Custom Steps

When writing a Java implementation for a **custom** step, consider these points:

- The method must be declared `public` so it can be called from outside the class.
- The method must be nonstatic.
- Each method must have a unique name.
- Parameter lists must match the supported combinations.
- The method can throw any type of exception.
- If you don't provide a method name, Workday selects any public method (ignoring getters and setters) with a parameter list matching the supported method signatures. If there's any ambiguity, Workday reports an error during deployment, which you can resolve by specifying a method name.

A **custom** step must provide a method that adheres to one of these signatures:

©2026 Workday, Inc. All rights reserved

Signature	Usage	Example
`public` *`return type`* `method(`*`input parameter type`* `input) [throws` *`any Exception`*`]`	Defines the return type of the method and an input parameter.	`public String addTimestamp(byte[] messageContent) throws IOException {`
`public void method(`*`input parameter type`* `input,` *`output parameter type`* `output) [throws` *`any Exception`*`]`	Defines supported input and output parameters.	`public void addTimestamp(byte[] messageContent, OutputStream out) throws IOException {`

The **custom** step supports these input, return, and output parameter types:

Input Types	Return Types	Output Parameter Types
• `java.lang.String` • `byte[]` • `java.io.InputStream` • `javax.activation.DataHandler` • `javax.activation.DataSource` • `org.w3c.dom.Element` • `org.w3c.dom.Document` • `javax.xml.transform.Source` • `javax.xml.transform.stream.StreamSource` • `javax.xml.transform.dom.DOMSource`	• `void` • `boolean` (**custom** step only) • `java.lang.String` • `byte[]` • `java.io.InputStream` • `javax.activation.DataHandler` • `javax.activation.DataSource` • `org.w3c.dom.Element` • `org.w3c.dom.Document` • `javax.xml.transform.Source`	• `java.io.OutputStream` • `javax.xml.transform.Result`

If a **custom** step returns a `void` type or a null value, it doesn't alter the mediation message.

Concept: Custom Element Packaging

Java Class Annotations

If you want to reuse a custom assembly element in different projects, use annotations to have the element display on the **Palette**.

To do so, select the **Annotate Java class to appear in palette** check box when you create the Java class and Spring bean. Studio adds an `@Component` annotation in the generated Java class, which is located in your assembly project's **src** folder. Example:

```
import com.capeclear.assembly.annotation.Component;
import com.capeclear.assembly.annotation.ComponentMethod;
import com.capeclear.assembly.annotation.Property;
import static com.capeclear.assembly.annotation.Component.Type.*;



@Component(
        name = "TimeStamperCustomMediationStepImpl",
        type = mediation,
        toolTip = "This custom code step adds the current time on the first
 line of the message.",
        scope = "singleton",
        smallIconPath = "icons/TimeStamperCustomMediationStepImpl_16.png",
        largeIconPath = "icons/TimeStamperCustomMediationStepImpl_24.png"
        )
```

©2026 Workday, Inc. All rights reserved

You can manually edit this information.

Note that Studio stores icons for the custom element in the project's `src/icons` folder. There are 2 versions, 16x16 and 24x24 pixels. To use custom icons, replace the _16.png and _24.png files with your own images.

Exposing Customized Assembly Element Properties

If your custom Java code has properties that you want users to set, you can display them on the **Bean** tab of the custom element's **Properties** view.

To do so, provide getter and setter methods that follow the bean pattern. Example: for property `abc`, provide getter and setter methods such as `getAbc` and `setAbc`.

Then add the `@Property` annotation to either the `set` or the `get` method. The `name` value defines the label for the property in the **Properties** view and the `toolTip` value defines the text that Studio displays when you mouse-over the label. You can control the order in which properties are listed using an order value. Lower values display first. An `editelement` value specifies the type of button displayed next to the property input field, as follows:

```
// Displays a text box with a Select button to enable you to select a Spring
 bean.
@Property(name="Text", edit=Edit.BEAN_REF)

// Displays a text box with a Select button to enable you to select a Java
 class.
@Property(name="Text", edit=Edit.CLASS)

// Displays a text box with a dialog box that enables you to specify a cron
 expression.
@Property(name="Text", edit=Edit.CRON)

// Displays a text box without a button.
@Property(name="Text", edit=Edit.DEFAULT)

// Displays a Browse button to enable you to specify a file.
@Property(name="Text", edit=Edit.FILE)

// Displays a Browse button to enable you to specify a folder.
@Property(name="Text", edit=Edit.FOLDER)
```

Studio looks for customization classes on the project classpath of the project that contains the assembly.xml file being edited. To distribute your custom component to multiple assembly projects, package the customization in binary form as a JAR file containing the class files and icon images. Other users can include this JAR within the project's classpath to make it available in their Assembly Editor.

The Assembly Editor automatically loads any customized components it finds in its classpath at startup. Right-click on the assembly diagram background and select **Refresh Customizations** to reload any recently added custom components.

Spring Bean Configuration

Concept: Spring Bean Configuration

You can edit Spring bean properties on the **Beans** tab on the Assembly Editor's **Properties** view.

Studio displays all of the assembly's beans on the left of the view beneath a top-level **beans** node. The properties associated with the **beans** node are global, affecting all beans in the assembly.

©2026 Workday, Inc. All rights reserved

Various options display above the beans. Their availability depends on your selection in the beans list. The options are:

Option	Description
Create 'bean'	Creates a Spring bean element in the assembly file. Select the top-level **beans** node to make this option available.
Create 'alias'	Creates an alias for a Spring bean element in the assembly file. Select the top-level **beans** node to make this option available.
Create 'property'	Creates a property element for the selected bean.
Create 'constructor-arg'	Creates a `constructor-arg` element for the selected bean.
Create 'ref'	Creates a `ref` element, enabling other beans to reference the selected property.
Create 'list'	Creates a `list` element for the selected property, enabling you to define the properties of a List Java collection.
Create 'map'	Creates a `map` element for the selected property, enabling you to define the properties of a Map Java collection.
Create 'import'	Creates a top-level `import` element within the assembly's beans definition, enabling you to specify a resource file containing one or more bean definitions. Select the top-level **beans** to make this option available.
Create 'value'	Creates a nested `value` element within the selected property definition, enabling you to specify the property as a human-readable string.

Reference: Global Bean Properties

Property	XML Attribute Name	Description
Default Autowire	`default-autowire`	Inspects the `BeanFactory` to automatically resolve beans.
Default Autowire Candidate	`default-autowire-candidate`	Enables you to limit autowiring to a set of bean candidates based on pattern matching against bean names.
Default Dependency Check	`default-dependency-check`	Specifies whether Studio checks for unresolved bean dependencies.
Default Destroy Method	`default-destroy-method`	Checks every bean for the specified destroy callback method.
Default Init Method	`default-init-method`	An initialization callback method for every bean.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Default Lazy Init	`default-lazy-init`	Specifies whether bean instances are created at startup or when first requested. If you don't want a singleton bean to be preinstantiated when using an `ApplicationContext`, set to **true**.
Default Merge	`default-merge`	Determines whether Studio merges collections by default.

Reference: Individual Bean Properties

Property	XML Attribute Name	Description
Id	`id`	The unique ID of the custom bean element within the assembly.
Abstract	`abstract`	Indicates whether the bean is usable only as a parent bean definition that will serve as a template for child definitions. The default value is `false`.
Autowire	`autowire`	Inspects the `BeanFactory` to resolve beans automatically.
Autowire Candidate	`autowire-candidate`	Enables you to limit autowiring to a set of bean candidates based on pattern matching against bean names.
Class	`class`	The implementation class name.
Dependency Check	`dependency-check`	Indicates whether Studio checks for unresolved bean dependencies.
Depends On	`depends-on`	Forces initialization of one or more beans before the bean using this element initializes. To express dependency on multiple beans, list their names separated by commas, whitespace, or semicolons.
Destroy Method	`destroy-method`	Checks the bean for the specified destroy callback method.
Factory Bean	`factory-bean`	The bean in the current or parent container that holds the instance method whose invocation creates the object.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Factory Method	`factory-method`	The name of the factory method to use when instantiating the bean.
Init Method	`init-method`	A generic initialization method for the bean.
Lazy Init	`lazy-init`	Indicates whether the bean instance should be created at startup or when first requested. If you don't want a singleton bean to be preinstantiated when using an `ApplicationContext`, set to **true**. By default, this property takes its value from the container-level setting.
Name	`name`	A name for the bean.
Parent	`parent`	The parent bean. If specified, the current bean inherits its configuration data.
Primary	`primary`	Designates the current bean definition as the primary autowire candidate. The default value is `false`.
Scope	`scope`	The scope of the objects created from the bean definition. The default value is `singleton`.

Reference: Alias Properties

Property	XML Attribute Name	Description
Alias	`alias`	The alias name.
Name	`name`	The ID of the original bean.

Reference: Property Properties

Property	XML Attribute Name	Description
Name	`name`	The property name.
Spring Bean	`spring-bean`	The ID of another bean in the container.
Value	`value`	The property value.

©2026 Workday, Inc. All rights reserved

Reference: Constructor-Arg Properties

Property	XML Attribute Name	Description
Index	`index`	A zero-based index for the constructor argument.
Spring Bean	`spring-bean`	The ID of another bean in the container. The constructor arguments specified in the bean definition are used to pass in as arguments to the constructor of the referenced bean.
Type	`type`	The type of the constructor argument.
Value	`value`	The value of the constructor argument.

Reference: Ref Properties

Property	XML Attribute Name	Description
Bean	`bean`	References a target bean by specifying one of the values in its **Name** or its **Id**.
Local	`local`	References a local target bean by specifying its **Id** attribute. Enables the XML parser to validate XML ID references within the same file.
Parent	`parent`	References a target bean in a parent container of the current container. Specifies one of the values in its **Name** or its **Id**.

Reference: List Properties

Property	XML Attribute Name	Description
Merge	`merge`	Specifies whether Java List collections are merged. By default, the `merge` property takes its value from the container-level `default-merge` setting.
Value Type	`value-type`	The Java type for `value` elements in the list.

Reference: Map Properties

Property	XML Attribute Name	Description
Merge	`merge`	Specifies whether Java Map collections are merged. By

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		default, the `merge` property takes its value from the container-level `default-merge` setting.
Value Type	`value-type`	The Java type for `value` elements in the list.
Key Type	`key-type`	The Java type for `key` elements in the map.

Message Builder

Create XML Literals Based on Schema or WSDLs

Prerequisites

Create an assembly containing 1 of these elements:

- A **write** step
- A **log** step
- A **log-error** error handler

Context

Workday Studio's Literal Builder enables you to create simple XML documents that conform to selected schema or SOAP request messages that conform to selected WSDLs. You can append WS-Security headers to SOAP request messages.

Steps

1. Add a **text** element to 1 of these assembly elements:
   - A **write** step
   - A **log** step
   - A **log-error** error handler
2. In the right pane, click **Generate XML literal**.
3. In the **Literal Builder** wizard, select the schema or WSDL element that you want to generate the XML literal for. As you complete the task, consider:

Option	Description
Show Messages	By default, Studio displays WSDL elements as messages. To display them as interface definitions with `portTypes`, operations, input messages, and output messages, deselect this option.
Wrap SOAP	Wraps the selected XML definition in a SOAP envelope.
Add WS-Security Header	Adds a SOAP `Header` element to the SOAP request. The SOAP header includes the WS-Security credentials you provide within a `UsernameToken` element. Only available if you select **Wrap SOAP**.

©2026 Workday, Inc. All rights reserved

Option	Description
Wrap XSLT	Creates an XML literal wrapped in an XSL stylesheet. You can save the stylesheet as an XSLT file for later use.
Ask for Choices Selection	Prompts you to select a particular choice from a schema when creating and copying an XML literal or SOAP literal. If you don't select this option, Studio uses the first choice in a particular set.

Result

Studio displays a preview of the XML literal, which you can edit.

Create XPath Expressions Based on Schema or WSDLs

Prerequisites

Create an assembly containing 1 of these steps:

- **java-to-xml**
- **validate**
- **xml-to-java**

Context

In Workday Studio, the XPath Expression Builder enables you to create XPath expressions based on selected schema or WSDLs.

Steps

1. On the **Properties** view of 1 of these steps, click **Build XPath Expression**, next to the **XPath** property:
    - **java-to-xml**
    - **validate**
    - **xml-to-java**
2. On the lower pane of the **XPath Expression Builder** wizard, select the schema or WSDL element that the XPath expression will match. As you complete the task, consider:

Option	Description
Show Messages	By default, Studio displays WSDL elements as messages. To display them as interface definitions with `portTypes`, operations, input messages, and output messages, deselect this option.
Prefix SOAP	By default, Studio prefixes the XPath expressions you create with this expression, as shown in the upper text pane: `env:Envelope/env:Body`. To remove the prefix, deselect this option.

3. If required, you can edit the XPath expression manually in the upper text pane. Example: you might want to use a predicate to filter the nodeset that the XPath returns.

©2026 Workday, Inc. All rights reserved

Concept: Workday Studio Message Builder

Workday Studio's **Message Builder** provides a graphical view that simplifies message content definition for these assembly elements:

- The **write** step
- The **log** step
- The **log-error** error handler

The **Message Builder** tab in the Assembly Editor provides a single location where you can define message content for any relevant element in the assembly. **Message Builder** is also available as a tab in the **Properties** view of individual **write**, **log**, and **log-error** elements.

The left **Message Builder** pane enables you to define the message content type. Use the options at the top to add various types of message elements:

Option	Description
Create 'message-content'	Adds the content of the mediation message to the message.
Create 'text'	Adds a string literal to the message. There are 3 options: • Enter the text manually, then click **Apply changes**. • Click **Insert MVEL Expression**, enter MVEL expressions (in combination with plain text, if required), then click Apply. • Click **Generate XML literal** and select the defining schema or WSDL element.
Create 'exception'	Adds an exception to the message if one exists in the mediation context. For use in the **log-error** error handler only.
Create 'line-separator'	Adds a new-line character to the message.
Create 'xpath'	Adds the result of an XPath expression to the message.
Create 'xslt'	Adds the result of an XSL transformation to the message.
Create 'xml-stream'	Creates the content for the **log** or **write** step.
Create 'regex-match'	Applies a regular expression to the message content and adds all matching occurrences to the mediation message.
Create 'static-file'	Specifies an input file that populates the message with its entire contents.
Create 'message-store-id'	Adds the message store ID of the message. Applies only if you set the `store-message` property for the transport.
Create 'header'	Adds the value of a named message header to the message.
Create 'headers'	Adds complete header sets in the form of name-value header pairs separated by new-line characters.

©2026 Workday, Inc. All rights reserved

Option	Description
Create 'customer-id'	Adds the customer ID from the `MediationContext` to the message. Displays 'null' in the message if the customer ID isn't set.

The right pane is where you define the message element's value. The contents of the value pane vary depending on the message element type.

To reorder or delete message elements, use the buttons on the right of the content pane. Their order in **Message Builder** is mirrored in the XML.

Reference: Message Element Properties

Use the buttons at the top of **Message Builder** to add message elements to **write**, **log**, and **log-error** elements.

Not every message element has properties.

message-content

Property	XML Attribute Name	Description
Limit	`limit`	A limit on the number of characters the message can contain. The default value of zero indicates no character limit.

exception

Property	XML Attribute Name	Description
Escape Xml	`escape-xml`	Enables escaping of XML on exceptions in the **write** step.

xpath

Property	XML Attribute Name	Description
Namespaces	`namespaces`	Enables you to specify any namespace required by the XPath expression in the format `<prefix1> <namespace1> <prefix2> <namespace2>`.
XPath	`xpath`	The XPath expression used to extract a value from the incoming message.
XPath Version	`xpath-version`	Workday Studio supports XPath versions 1.0 and 2.0. The default value is 2 for assembly versions from Workday 12 onward and 1 for earlier versions.

©2026 Workday, Inc. All rights reserved

xslt

Property	XML Attribute Name	Description
Allow Saxon Bytecode Generation		Specifies whether to allow Saxon bytecode generation. The default is `false`.
Reuse Strategy		Specifies the XSLT transformer reuse strategy: • **default**: if you specify the `url` property using the mctx URL protocol or an MVEL expression, the default behavior is `no-reuse`. Otherwise, it's `transformer-pool`. Note: When you set **transformer-factory** to Saxon, the default behavior is `templates`. • **no-reuse**: Workday doesn't reuse transformer objects. • **templates**: Workday builds a Templates object, from which it creates fresh transformer objects. These transformers aren't reused. • **transformer-pool**: Workday uses a pool of transformers. Not recommended if the transformer object caches values.
Secure Processing		Specifies whether a transformer performs XSLT standard processing only. Set to **true** to impose limits on XML constructs. Example: ignore JavaScript or Java extension processing. Set to **false** to process XML according to the XML specifications.
Transformer Factory		The class name of the `TransformerFactory` that you want to use. The default is the JAXP factory. There are 3 other options: • `net.sf.saxon.TransformerFactor` for Saxon. • `org.apache.xalan.processor.Tra` for Xalan.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		• `net.sf.joost.trax.TransformerF` for Joost.
Url		The XSLT file location, either a relative path or an absolute URL. Supports MVEL expression templates. You can load XSLT directly from the message content using the mctx (mediation context) URL protocol. Example: to load an XSLT file contained in an attachment, use `mctx:parts/1`.

xml-stream

Property	XML Attribute Name	Description
Namespaces	`namespaces`	Enables you to specify any namespace required by the XPath expression in the format `<prefix1> <namespace1> <prefix2> <namespace2>`.
XPath	`xpath`	The XPath expression to apply to the message when retrieving XML elements. Workday uses Streaming API for XML (StAX) processing. This library reads XML messages and translates them into events, rather than DOM. It doesn't perform any transformations itself. The XPath expression is the same simplified XPath subset you use in the `xml-stream-splitter` strategy. Workday supports qualified or local element names and asterisks. It doesn't support the full XPath standard. Examples: • `/order/item, /ns:order/ns:order` • `//item, //ns:item, /ns:order/item` • `/*/*, //* /*/*`
XPath Version	`xpath-version`	Workday Studio supports XPath versions 1.0 and 2.0. The default value is 2 for assembly

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		versions from Workday 12 onward and 1 for earlier versions.
Limit	`limit`	Constrains the number of XML fragments to match based on the XPath expression. Useful mainly for performance reasons. If you know there's just 1 matching XML fragment, set the limit to `1` to prevent pull-parsing of the document after the result has been retrieved. The default value is `-1`, which means an unlimited result set.

regex-match

Property	XML Attribute Name	Description
Delimiter	`delimiter`	Specifies a string used to separate 2 or more occurrences. The default value is a single space.
Limit	`limit`	Specifies the number of occurrences to include in the result set. The default value is zero, which means all occurrences.
Regex	`regex`	Defines the matching expression.

static-file

Property	XML Attribute Name	Description
Character Encoding	`character-encoding`	Specifies the character encoding. The default value is `UTF-8`.
Input File	`input-file`	Specifies the file.

header

Property	XML Attribute Name	Description
Include Name	`include-name`	Specifies whether Studio inserts the header name.
Name	`name`	The header name.
Not Set Value	`not-set-value`	A string that's emitted if the header isn't found or is empty.

©2026 Workday, Inc. All rights reserved

Transport Bindings

Concept: Transport Bindings

Transport bindings are optional child elements that you can configure on out-transports in Studio. Transport bindings enable out-transports to support web services in a specified format.

Example: you can configure a **rest-binding** child element on an **http-out** transport when you want to invoke a RESTful web service.

You can configure transport bindings on these out-transports:

- **http-out**
- **ftp-out**
- **ftps-out**
- **sftp-out**
- **xmpp-out**
- **custom-out**
- **email-out**

Studio supports these transport bindings:

Transport Binding	Description	Notes
rest-binding	Enables you to invoke RESTful web services using HTTP request methods.	Supports these HTTP methods: • GET • PUT • POST
wsdl-http-binding	Enables you to invoke web services using HTTP request methods and a WSDL file.	Supports these HTTP methods: • GET • PUT • POST
wsdl-soap-binding	Enables you to invoke SOAP-based web services using HTTP request methods.	Supports these HTTP methods: • GET • PUT • POST Supports these child elements • **custom-policy** • **ws-rm** • **ws-security** • **ws-ut**

Reference: WSDL-HTTP-Binding Properties

Common Tab

Property	XML Attribute Name	Description
Endpoint Name	`endpoint-name`	The endpoint from which the **wsdl-http-binding** element accesses a web service.

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
Interface Name	`interface-name`	The `portType` (or interface) name for the web service.
Service Name	`service-name`	The name of the web service.
Wsdl File	`wsdl-name`	The WSDL file that describes the web service.
Serialize Params	`serializeParams`	Not currently in use.

Reference: WSDL-SOAP-Binding Properties

Common Tab

Property	XML Attribute Name	Description
Endpoint Name	`endpoint-name`	The endpoint from which the **wsdl-soap-binding** element accesses a web service.
Interface Name	`interface-name`	The `portType` (or interface) name for the web service.
Service Name	`service-name`	The name of the web service.
Wsdl File	`wsdl-name`	The WSDL file that describes the web service.
Fault Handling	`fault-handling`	Specifies the error-handling process of the **wsdl-soap-binding** element for HTTP 500 response codes. Options are: • **ignore** • **error** • **abort**

WSDL-SOAP-Binding Child Elements

Concept: WSDL-SOAP-Binding Child Elements

The **wsdl-soap-binding** transport binding supports these child elements:

Child Element	Description
custom-policy	Supports a custom web service security policy for an out-transport.
ws-rm	Supports a messaging security policy for an out-transport.
ws-security	Implements a local file containing the web service security policy for SOAP requests on an out-transport.
ws-ut	Enables the Workday integration server to extract, validate, and propagate username and password

©2026 Workday, Inc. All rights reserved

Child Element	Description
	security credentials on an out-transport for an assembly.

Reference: Custom-Policy Properties

Common Tab

Property	XML Attribute Name	Description
Method	`method`	The method invoked on your custom security policy.
Spring bean	`ref`	The custom Spring bean that implements your custom security policy.

Reference: WS-Rm Properties

Common Tab

Property	XML Attribute Name	Description
File	`file`	The path to the file that describes your messaging security policy configuration.

Reference: WS-Security Properties

Common Tab

Property	XML Attribute Name	Description
File	`file`	The path to the file that describes your web service security configuration.

Reference: WS-Ut Properties

Common Tab

Property	XML Attribute Name	Description
Username	`username`	Inserts a username into the security header of outbound SOAP messages in the `MediationContext`.
Password	`password`	Inserts a password into the security header of outbound SOAP messages in the `MediationContext`.
Password Digest Algorithm	`password-digest-algorithm`	Applies a hashing algorithm to passwords in the security header of outbound

©2026 Workday, Inc. All rights reserved

Property	XML Attribute Name	Description
		SOAP messages in the `MediationContext`. Options are: • *SHA-1* • *SHA-256* • *SHA-384* • *SHA-512* • *none*
Spring Bean	`ref`	An optional Spring bean that authenticates username and password values in the security header of outbound SOAP messages in the `MediationContext`.
Created	`created`	Creates a timestamp element in the security header of outbound SOAP messages in the `MediationContext`.
Nonce	`nonce`	Creates a nonce element in the security header of outbound SOAP messages in the `MediationContext`.

Schema Explorer

Display WSDL and XSD Files

Context

The **Schema Explorer** view displays WSDL and XSD files in read-only format. You can copy content from these files to your clipboard, enabling you to reuse the data in other text editor views in Studio.

Steps

1. Select **Window > Show View > Schema Explorer**.
2. On the **Schema Explorer** view, click **Add WSDL or XSD**.
3. Select a WSDL or XSD file.
4. On the **Schema Explorer** view, expand the WSDL or XSD file to view its structure.

Result

Studio displays the WSDL or XSD file on the **Schema Explorer** view.

Next Steps

Copy content from the WSDL or XSD file to your clipboard.

©2026 Workday, Inc. All rights reserved

Display Workday Web Service WSDLs

Context

The **Schema Explorer** view displays WSDL and XSD files in read-only format. You can copy content from these files to your clipboard, enabling you to reuse the data in other text editor views in Studio.

Steps

1. Select **Window > Show View > Schema Explorer**.
2. On the **Schema Explorer** view, click **Add Workday Web Services**.
3. Select and expand a Workday Web Service.
4. Select **wsdl**.
5. On the **Schema Explorer** view, expand the Workday Web Service WSDL to view its structure.

Result

Studio displays the Workday Web Service WSDL on the **Schema Explorer** view.

Next Steps

Copy content from the Workday Web Service WSDL to your clipboard.

Display Custom RaaS Report Schemas

Context

The **Schema Explorer** view displays WSDL and schema (XSD) files in read-only format. You can copy content from these files to your clipboard, enabling you to reuse the data in other text editor views in Studio.

Steps

1. Select **Window > Show View > Schema Explorer**.
2. On the **Schema Explorer** view, click **Add Custom Raas Report Schema**.
3. On the **Add Custom RaaS Report** window, consider:

Option	Description
Environment	The Workday environment that contains your custom report.
Report name	The name of your custom report.

4. On the **Select Reports** window, select a custom report.
5. Click **Finish**.
6. On the **Schema Explorer** view, expand the custom report schema to view its structure.

Result

Studio displays the custom report schema on the **Schema Explorer** view.

Next Steps

Copy content from the custom report schema to your clipboard.

©2026 Workday, Inc. All rights reserved

Concept: Schema Explorer

The **Schema Explorer** view displays WSDL and XSD files in read-only format. You can copy content from these files to your clipboard, enabling you to reuse the data in other text editor views in Studio.

You can filter the contents of the **Schema Explorer** view by entering keywords in the **type filter text** field.

By default, the **Schema Explorer** view displays on the top right-hand side of the screen. You can drag and drop the **Schema Explorer** view to your preferred location on the Studio user interface.

MVEL

Concept: MVEL Fundamentals

Syntax

MVEL is an expression language for Java-based applications. Its syntax is similar to Java's but it has some additions that make it more efficient at matching content.

Note:

- MVEL uses Java namespaces and classes, but can't itself declare either.
- Unlike Java, MVEL is dynamically typed, with optional typing.
- MVEL expressions can be single statements or multiline scripts. For multiline scripts, use a semicolon to separate each statement.
- MVEL operates on the principle of last value out. It supports the `return` keyword but doesn't generally require it.
- MVEL supports operator precedence rules and uses bracketing to control execution order.
- You can enclose MVEL string literals in single or double quotation marks.
- The semicolon is a reserved character in MVEL. You can't use it in expressions. Use `(char) 59` instead. Example: `props['string']='a;b'` throws an error. This code does not:

```
props['string']= (char) 59
props['string_new']='a'props['string']'b'
```

MVEL expressions can include:

- Boolean expressions
- Method invocations
- Property expressions
- Variable assignments

An example of a simple MVEL property expression:

```
context.errorMessage
```

It extracts the property `errorMessage` from the variable or context object `context`.

MVEL supports all of the usual Java comparison, mathematical, and logical operators. This table summarizes other notable MVEL operators:

Operator	Description	Example
`new`	Instantiates an object.	`new String("foo")`
`with`	Performs multiple operations on a single object instance.	`with (value) { name = 'Foo', age = 50, sex = Sex.MALE }`

©2026 Workday, Inc. All rights reserved

Operator	Description	Example
`assert`	Asserts that a value is true or fails with an AssertionError.	`assert foo != null`
`contains`	Determines whether the value on the left contains the value on the right.	`var contains "Foo"`
`is` or `instance of`	Determines whether the value on the left is a member of the class on the right.	`var instanceof String`
`strsim`	Compares strings and returns their similarity expressed as a percentage.	`"foobol" strsim "foobar"`
`soundslike`	Performs a soundex comparison between 2 strings.	`"foobar" soundslike "fubar"`
`&`	Performs a Bitwise AND operation.	`foo & 5`
`|`	Performs a Bitwise OR operation.	`foo | 5`
`^`	Performs a Bitwise XOR operation.	`foo ^ 5`
`+`	Overloaded operator for concatenating 2 strings.	`"foo" + "bar"`
`#`	Concatenates 2 literals as strings.	`8 # 9`
`in`	Inspects objects inside a collection.	`(foo in list)`
`=`	Assigns a value to a variable.	`var = "foobar"`

Value Testing

In Java, the operator == checks whether references point to the same object. In MVEL, however, the comparison operator == checks whether values are equal. The MVEL expression `foo == 'bar'` is the equivalent of Java's `foo.equals("bar")`.

An example of an MVEL boolean expression:

```
user.name == 'John Smith'
```

It returns `true` if the `name` property of the variable or context object `user` is John Smith.

MVEL employs type coercion. If you compare 2 values of different types, MVEL tries to coerce the value on the right to the type on the left. If that isn't possible, it tries to coerce the value on the left to the type on the right. Example:

```
"123" == 123;
```

This expression returns `true` because MVEL coerces the value on the right to a string before it performs the comparison.

©2026 Workday, Inc. All rights reserved

Lists, Maps, and Arrays

Enclose MVEL lists in square brackets. Separate items with commas. Example:

```
["Paris", "London", "Berlin"]
```

Enclose MVEL maps in square brackets. Separate keys and values with colons. Example:

```
["Foo" : "Bar", "Bar" : "Foo"]
```

Enclose MVEL arrays in braces. Separate items with commas. Example:

```
{"Bill", "Bob", "Ben"}
```

Property Navigation

MVEL provides a single, unified syntax for accessing properties, static fields, lists, maps, and arrays.

In Java, you might access a property from an object using this statement:

```
context.getException().getCause()
```

In MVEL you can access the same property using this expression:

```
context.exception.cause
```

Access lists and arrays in the same manner. Example:

```
parts[2]
```

You access maps in the same way as arrays but can pass any object as the index value. Example:

```
props["foobar"]
```

If a map uses a string as a key, you can treat the map itself as a virtual object. Example:

```
props.foobar
```

MVEL treats strings as arrays for indexing purposes.

This example returns `B`:

```
foo = "Berlin";
foo[0];
```

Assignment

You can assign variables in an MVEL expression for use within the expression or for extraction at runtime.

Because MVEL is a dynamically typed language, you don't need to specify a type when you declare a new variable. But you can do so.

Both of these examples are valid in MVEL:

```
str = "Some string";
String str = "Some string";
```

©2026 Workday, Inc. All rights reserved

When assigning a value to a typed variable, MVEL attempts to perform automatic type conversion. Example:

```
String num = 1;
```

You can cast the value of a dynamically typed variable to another type. Example:

```
String num = 1;
```

Section: Typing and Coercion

MVEL uses dynamic typing, with optional static typing. Native Java objects are statically typed. To interact with them, MVEL must sometimes perform type coercion, which can have implications for performance.

MVEL automatically coerces type to what seems best for the current value. This means you must be careful when working with numbers. Errors can arise in floating point arithmetic because MVEL automatically casts away from infinite precision `java.math.BigDecimal` to `java.lang.Double` and `java.lang.Long`. This can cause problems when the type of the stored value is not what you expect. In general, you should avoid performing noninteger arithmetic with financial values in MVEL.

MVEL's type coercion means you can call Java methods without adjusting inputs. The interpreter or compiler analyzes the types you're passing to the method and determines what coercion is required.

In the event of an array type conflict, MVEL attempts to coerce the entire array to the needed input type.

MVEL performs any necessary coercion on a statically typed variable, provided the assigned value is coercible. If it isn't, an exception occurs.

Concept: MVEL and Studio

In Workday Studio, MVEL enables you to inspect and manipulate message parts and component parameters in your assemblies. You can use MVEL to:

- Extract information from a message.
- Format numbers and dates.
- Replace specific characters in a string.
- Retrieve integration attributes or launch parameters.

You should avoid using MVEL to manipulate large strings of data. Example: converting messages to strings can create performance and scalability issues.

In an MVEL-capable field, you can press Ctrl+Space to display a content assist popup window.

MVEL expressions can be single statements or multiline scripts. For multiline scripts, use a semicolon to separate each line.

Note: Studio evaluates multiline expressions as a unit, not sequentially.

MVEL-capable fields expand to accommodate new lines. After 6 lines, they stop expanding and become scrollable. Studio applies syntax coloring to MVEL expressions.

MVEL implements operator precedence during execution. You can use brackets to control the execution order of MVEL expressions.

Note: Because they're processed in memory, there's a 1 million character limit on XPath methods such as `xpath`, `xpathB`, and `xpathF`. You will not receive an error if your query exceeds this limit. You should use streaming APIs such as the **write** step's `xml-stream` element if you need to process large messages.

©2026 Workday, Inc. All rights reserved

Reference: Studio Helper Objects

Studio provides a number of helper objects that enable you to access an integration's runtime properties using MVEL. You can use the helper objects to make simple changes to runtime values, or to extract them and pass them to components in the assembly.

Helper Object	Purpose

`context`	Provides access to the `MediationContext` object - that is, to the dynamic state of the integration at runtime. The `MediationContext` object links to the current message, mediation properties, mediation variables, and current error information.
	
	Example: This expression retrieves the value of the property `myProperty` from the `MediationContext`:
	
	`context.getProperty('myProperty')`
	
	Example: This expression retrieves the property `someProperty` from the `MediationContext` if the property value is set to `someValue`:
	
	`context.getProperty('someProperty').equals('som`

`da`	The Document Accessor helper object. Provides access to the list of documents that were attached to the integration event before this integration ran. Effectively combines a Get Event Documents web service call with the retrieval of the documents referenced in the web service response. The `da` helper object has some limitations: • It must explicitly include logic to handle cases where an unexpected number of attachments is available. • It can't access the filetype of the attachment. • It can't access the character encoding of the attachment. The **doc-iterator** router strategy handles unexpected numbers of attachments automatically and it can access both MIME type and character encoding. It's often more appropriate for an integration to use that strategy to retrieve input documents, rather than this helper object. Example: This expression retrieves a document from an integration event with filename `eib_transform_output.txt` and adds a variable named `var1`: `da.toVar('eib_transform_output.txt', 'var1')`

©2026 Workday, Inc. All rights reserved

Helper Object	Purpose
	Example: This expression retrieves a document with the given label: `da.getDocsMatching('label')`
`ftp`	Provides access to the list of files returned when issuing the LIST request to an ftp, ftps, or sftp server. Example: This expression retrieves a list of files for a URL: `ftp.list('URL')`

`intsys`	Provides convenient and high-performance access to the integration system configuration as returned by the Get Integration Systems web service call.
	Example: This expression retrieves an integration attribute named `username`:
	`intsys.getAttribute('username')`
	Example: This expression displays as true if an integration service with the given name exists:
	`intsys.isServiceEnabled('name')`
	Note: If you use `intsys` to access an attribute, you must first define a service on the `workday-in` transport for the integration system.

`lp`	Provides access to launch parameters and other information in the integration launch event.
	Example: This expression retrieves the value of a launch parameter with the given name:
	`lp.getSimpleData('name')`
	Example: This expression displays as true if a parameter with the given label exists:
	`lp.exists('label') == true`
	Note: Before using the `lp` variable, you need to assign the launch parameters XML to the variable `wd.launchparameters`.

`message`	Provides access to the current message and its header information. Use this object to assign new content to the message.

©2026 Workday, Inc. All rights reserved

Helper Object	Purpose

	Example: This expression retrieves the root part of a message as text from the `MediationMessage` interface:
	`message.rootPartAsText`
	Example: This expression retrieves `MediationContext` variables:
	`message.variables`

`mtable`	Provides access to the `MTableAdapter` interface, which defines the operations on an `mtable` instance. Example: This expression sets data in an `mtable`: `mtable['my_property'].set(0, 'Data_Column', 'my_test_data')`

`parts`	Provides access to the individual parts of the current message. Most messages in Workday integrations have a single part, the root part. Use `parts[0]` to access the root part.
	Example: This expression retrieves the header `someHeader` from the root part:
	`parts[0].getHeader('someHeader')`
	Example: This expression implements the `xpathB` method to evaluate an XPath Boolean expression:
	`parts[0].xpathB('/a/b')`

`props`	Provides access to `MediationContext` properties.
	Example: This expression retrieves the value of the property `myProperty` from the `MediationContext`:
	`props.myProperty`
	Example: This expression retrieves the value of the property `bean.property` from the `MediationContext`:
	`props['bean.property']`

`spring`	Provides access to the creation and retrieval beans defined in an assembly's Spring configuration.

©2026 Workday, Inc. All rights reserved

Helper Object	Purpose

	Example: This expression retrieves the `name` attribute of the bean called `Fred` in the Spring context:
	`spring.getBean('fred').name`
	Example: This expression accesses the Spring application context:
	`spring.parent ApplicationContext`

`util`	A set of utility functions for performing common operations such as escaping values for insertion into XML or creating an HTTPS URL. Example: This expression checks whether the current message in the `MediationContext` is the last message that the splitter component will generate: `props['is.last.record'] = util.isLastMessageInBatch()` Some `util` functions have crucial limitations. Example: the `xpathToCommaDelimString` function produces a comma-separated list from the string values of the node sequence returned by an XPath expression operating on a given XML string. You can then split and manipulate the comma-separated list using Java string manipulation functions in MVEL. However, the function has these limitations: • It only supports XPath 1. • It can't produce CSV files since it doesn't correctly escape newlines, commas, and quotation characters. • It can't reliably separate data containing commas. • Its requirement for XML as a string encourages nonscalable practices such as converting the current context message or variable to a string. Note: Workday strongly recommends the use of explicit, versioned, web service requests for stability and backward compatibility. Using `MVELUtilHelper.getLatestWWSVersion()` in integrations isn't best practice, and can lead to integrations with unexpected failures or results.
`vars`	Provides access to `MediationContext` variables.

©2026 Workday, Inc. All rights reserved

Helper Object	Purpose
	Example: This expression retrieves the length of the variable `myVariable`: `vars['myVariable'].length` Example: This expression retrieves the length of the variable `myVariable`: `vars['myVariable'].length`

`xmldiff`	Provides programmatic access to the results of document comparison with the **xmldiff** component.
	Example: This expression checks whether there are differences between 2 documents:
	`xmldiff.isDifferent() == true`
	Example: This expression retrieves a list of differences between 2 documents:
	`xmldiff.changes`

Reference: Document Accessor Helper Object

Useful Functions

Name	Arguments	Return Value Type	Notes
`getData`	Integer index	Source	Retrieves the document at the specified zero-based index. Use in combination with the `message.setMessage` helper function to populate the current message with the retrieved data in a memory-efficient manner. Example: this code populates the current message with a UTF-8 XML message retrieved from the first attachment: `message.setMessage(da.getDa ‘text/xml’)` Results in an error if there's no integration

©2026 Workday, Inc. All rights reserved

Name	Arguments	Return Value Type	Notes
			attachment with the specified index.
	String filename	Source	Retrieves the document with the specified filename. Use in combination with the `message.setMessage` helper function to populate the current message with the retrieved data in a memory-efficient manner. Example: this code populates the current message with a UTF-8 XML message attachment with the name input.xml: `message.setMessage(da.getDa 'text/xml')` Results in an error if there's no integration attachment with the specified filename.
`getDocumentIndexes`	List<String> labels	List<Integer>	Returns a Java List of integers giving the indexes of documents tagged with all of the specified labels. Note that if one of the labels is '*' then any label is matched and the function returns all indexes for documents with at least 1 label.
`getDocsMatching`	List<String>	List<Source>	Retrieves documents tagged with all of the specified labels. Note: Workday stores documents smaller than 100 KB in memory and larger documents on disk. Be aware of runtime memory and disk space limits.
	String label	Source	Retrieves documents tagged with the specified label.

©2026 Workday, Inc. All rights reserved

Name	Arguments	Return Value Type	Notes
			Note: Label here means integration attachment label. Synonymous with document tag.
	String[] labels	Source	Retrieves documents tagged with all of the labels in the string array.
`getFileName`	Integer index	String	Retrieves the filename with the specified index.
	String label	List<String>	Retrieves a Java List of the filenames of all the documents tagged with the specified label.
	List<String> labels	List<String>	Retrieves a Java List of the filenames of all the documents tagged with the specified labels.
	String label, integer index	String	Retrieves the nth filename of all the documents tagged with the specified label. Functionally equivalent to this code: `da.getFileNames(label).get(` In most cases, you must use `da.getFileNames(label)` to find the number of matching filenames before you can use an index. The only exception is the limited case where you require the first (zero index) value.
	List<String> labels, integer index	String	Retrieves the nth filename of all the documents tagged with the specified labels. Functionally equivalent to this code: `da.getFileNames(labels).get`

©2026 Workday, Inc. All rights reserved

Name	Arguments	Return Value Type	Notes
			In most cases, you must use `da.getFileNames(labels)` to find the number of matching filenames before you can use an index. The only exception is the limited case where you require the first (zero index) value.
	String[] labels, integer index		Retrieves the nth filename of all the documents tagged with the specified labels. Functionally equivalent to this code: `da.getFileNames(java.util.A` In most cases, you must use `da.getFileNames(labels)` to find the number of matching filenames before you can use an index. The only exception is the limited case where you require the first (zero index) value.
`hasFile`	String filename	Boolean	Determines whether there's an integration attachment with the specified filename.
	String label, String filename	Boolean	Determines whether there's an integration attachment with the specified filename and tagged with the specified label.
	List<String> labels, String filename	Boolean	Determines whether there's an integration attachment with the specified filename and tagged with all of the labels in the list.
`sort`	String order	None	Sorts the attached documents by filename. Valid order values are:

©2026 Workday, Inc. All rights reserved

Name	Arguments	Return Value Type	Notes
			• `"FILENAME_ASCENDING"` • `"FILENAME_DESCENDING"` Any attachments with no filename place last.
`size`	None	Integer	Returns the number of attached documents.
	List<String> labels	Integer	Returns the number of documents tagged with all of the specified labels.
	String label	Integer	Returns the number of documents tagged with the specified label.
	String[] labels	Integer	Returns the number of documents tagged with all of the labels in the string array.
`toVar`	Integer index, String var_name	MessageVariable	Retrieves the document at the specified index and places it in a memory-efficient manner into the specified assembly variable. Example: `da.toVar(0, 'myInputDocument')`
	String filename, String var_name	MessageVariable	Retrieves the document with the specified filename and places it in a memory-efficient manner into the specified assembly variable. Example: `da.toVar('WorkerData.xml', 'myInputDocument')`

Reference: Launch Parameters Helper Object

Useful Properties

Name	Notes
`integrationEventWID`	Returns the WID for the integration event relating to the current launch of the

©2026 Workday, Inc. All rights reserved

Name	Notes
	integration. Useful when making webservice calls such as `Get_Event_Documents`, `Put_Integration_Message`, or `Put_Integration_Event`. Automatically provides a default parameter to the wcc://GetEventDocuments, wcc://PutIntegrationMessage, and wcc://PutIntegrationEvent common components.
`integrationSystem`	Returns the `Integration_System_ID` for the current integration system. Useful when populating webservice calls for event subscription or field override requests.
`integrationSystemRefWID`	Returns the WID for the current integration system. An entirely equivalent alternative to the `integrationSystem` property.
`parentEventWID`	Returns the WID for the current launch event's parent event.
`sentOn`	Returns the date and time of the integration's launch formatted as an ISO-8601 date and time.
`sentOnAsDate`	Returns the date and time of the integration's launch in the form of a `java.util.Date` object.

Useful Functions

Note:

Some of these functions specify a launch parameter by its name alone. Others specify both the launch parameter's name and the name of the integration service that provides it. The latter are useful when a launch parameter with the same name exists in multiple integration services configured on the integration system. Most integrations don't need to accommodate launch parameter names that are duplicated across multiple services, so those functions are less frequently required.

Name	Arguments	Return Value Type	Notes
`exists`	String name	Boolean	Returns true if the named launch parameter has a nonempty value.
	String service_name, String name	Boolean	Returns true if the named launch parameter in the specified integration service has a nonempty value. Preferable to the single argument version only when a launch parameter of the same name exists in multiple integration

©2026 Workday, Inc. All rights reserved

Name	Arguments	Return Value Type	Notes
			services enabled on the integration system.
`getDate`	String name	String	Returns the String-formatted date value of the runtime parameter. The format is YYYY-MM-DDZ or YYYY-MM-DD+/-offset.
	String service_name, String name	String	Returns the String-formatted date value of the runtime parameter of the specified service. The format is YYYY-MM-DDZ or YYYY-MM-DD+/-offset.
`getParameterData`	String name	org.w3c.dom.Element	Returns the XML element `Integration_Runtime_Paramete` which holds the named launch parameter from the `Launch_Integration_Event` message that's sent to the integration at startup.
	String service_name, String name	org.w3c.dom.Element	Returns the XML element `Integration_Runtime_Paramete` which holds the named launch parameter of the specified service from the `Launch_Integration_Event` message that's sent to the integration at startup.
`getReferenceData`	String name, String type	String	Returns the reference ID of the given type for the launch parameter with the specified name. Example: this code returns the employee ID for a launch parameter named Employee: `lp.getReferenceData('Employ 'Employee_ID')`

©2026 Workday, Inc. All rights reserved

Name	Arguments	Return Value Type	Notes
`getReferenceDataList`	String name, String type	List<String>	Returns a Java List containing zero or more reference IDs from a multi-instance launch parameter.
	String service_name, String name, String type	List<String>	Returns a Java List containing zero or more reference IDs from a multi-instance launch parameter of a specified service.
`getSequencedValue`	String name	String	Returns the current value of the sequence generator with the specified name.
	String service_name, String name	String	Returns the current value of the specifed sequence generator declared on the specified service.

Reference: Util Helper Object

Useful Functions

Name	Arguments	Return Value Type	Notes
`cleanString`	String value	String	Used to escape any input characters that must be escaped in XML documents. Example: `<Value>@{util.cleanString(p root/row/ a_value'))}</ Value>`
`getExtForContentType`	String mimetype	String	Enables an integration to obtain a file extension such as .txt or .xml, based on the specified MIME type.
`getFilenameFromContentDisposition`	String content-disposition	String	Extracts a filename from a content-disposition string. Some assembly components such as the document iterator strategy populate the content-disposition

©2026 Workday, Inc. All rights reserved

Name	Arguments	Return Value Type	Notes
			message header with filename information. Example: this code returns the value `testfile.pdf`: `util.getFilenameFromContent filename=testfile.pdf')`
`readFileToVar`	String var_name, String filename		Enables nontext files to be read into a variable. This is the only delivered means by which nontext documents can be loaded. Note that the **write** step's static-file attribute can only load text documents into a message or variable. The file is specified relative to the WSAR-INF directory. Example: this code reads the image file icon-32.png from the WSAR-INF directory into the context variable v1: `util.readFileToVar('v1', 'icon-32.png')`
`readFileToVar`	String var_name, String filename,String mimetype		Enables nontext files to be read into a variable whose MIME type you specify. This is the only delivered means by which nontext documents can be loaded. Note that the **write** step's static-file attribute can only load text documents into a message or variable. The file is specified relative to the WSAR-INF directory. Example: this code reads the image file icon-32.png from the WSAR-INF directory

©2026 Workday, Inc. All rights reserved

Name	Arguments	Return Value Type	Notes
			into the context variable v1 with the MIME type image/png: `util.readFileToVar(‘v1’, ‘icon-32.png’, ‘image/png’)`

Reference: Sample Studio MVEL Expressions

Variable	Sample Expression	Description
`context`	`context.getProperty('myProperty')`	Retrieves the value of the property `myProperty` from the `MediationContext`.
	`context.getProperty('someProperty').equals('someValue')`	Retrieves the property `someProperty` from the `MediationContext` if the property value is set to `someValue`.
`da`	`da.toVar('eib_transform_output.txt', 'var1')`	Retrieves a document from an integration event with filename `eib_transform_output.txt` and adds a variable named `var1`.
	`da.getDocsMatching('label')`	Retrieves a document with the given label.
`ec`	`ec.getAttribute('attributeName')`	Retrieves an attribute name.
	`ec.integrationMapLookup('map', 'key')`	Retrieves an integration map value.
`env`	`env['cc.hostname']`	Retrieves the `cc.hostname` environment property set

©2026 Workday, Inc. All rights reserved

Variable	Sample Expression	Description
		as `-Dcc.hostname` on the command line.
	`env.containsKey('key')`	Displays as true if environment properties contain the given key.
`ftp`	`ftp.list('URL')`	Retrieves a list of files for a URL.
`intsys`	`intsys.getAttribute('username')`	Retrieves an integration attribute named `username`.
	`intsys.isServiceEnabled('name')`	Displays as true if an integration service with the given name exists.
`lp`	`lp.getSimpleData('name')`	Retrieves the value of a launch parameter with the given name.
	`lp.exists('label') == true`	Displays as true if a parameter with the given label exists.
`message`	`message.rootPartAsText`	Retrieves the root part of a message as text from the `MediationMessage` interface.
	`message.variables`	Retrieves `MediationContext` variables.
`mtable`	`mtable['my_property'].set(0, 'Data_Column', 'my_test_data')`	Sets data in an `mtable`.
	`mtable['my_property'].get(0, 'Data_Column')`	Retrieves data from an `mtable`.
`parts`	`parts[0].getHeader('someHeader')`	Retrieves the header `someHeader` from the root part.
	`parts[0].xpathB('/a/b')`	Implements the `xpathB` method to evaluate an XPath Boolean expression.
`props`	`props.myProperty`	Retrieves the value of the property `myProperty` from the `MediationContext`.
	`props['bean.property']`	Retrieves the value of the property `bean.property` from the `MediationContext`.
`spring`	`spring.getBean('fred').name`	Retrieves the `name` attribute of the bean called `Fred` in the Spring context.
	`spring.parent ApplicationContext`	Accesses the Spring application context.
`util`	`props['is.last.record'] = util.isLastMessageInBatch()`	Checks whether the current message in the `MediationContext` is the

©2026 Workday, Inc. All rights reserved

Variable	Sample Expression	Description
		last message that the splitter component will generate.
	`<ns:Message_Summary>@{util.cleanString(props['is.message.sum ns:Message_Summary>`	Implements the `cleanString(String)` function to ensure that an assembly component produces well-formed XML.
`vars`	`vars['myVariable'].length`	Retrieves the length of the variable `myVariable`.
	`vars['myVariable'].mimeType`	Retrieves the MIME type associated with the variable `myVariable`.
`xmldiff`	`xmldiff.isDifferent() == true`	Checks whether there are differences between 2 documents.
	`xmldiff.changes`	Retrieves a list of differences between 2 documents.

Reference: Studio MVEL Auto-Conversion Syntax Samples

Function	Sample Expression	Description
Auto-convert a string to a variable.	`vars['fred'] = '<hello>world</hello>'`	Creates a new variable called `fred` with the content `<hello>world</hello>`. Use this method to create small messages only. Using expressions to create large strings can lead to poor performance. In most cases, you should use the `write` step to create content for a variable.
Supply content and MIME type information to a variable.	`vars['fred'] = [ 'Hello World', 'text/plain' ]`	Creates a new variable called `fred` with the content `Hello World` and MIME type `text/plain`. Use this method to create small messages only. Using expressions to create large strings can lead to poor performance. In most cases, you should use the `write` step to create

©2026 Workday, Inc. All rights reserved

Function	Sample Expression	Description
		content for a variable and configure the MIME type.
Copy from an attachment to a variable.	`vars['fred'] = parts[1]`	Creates a new variable `fred` from the first attachment of the current message. In most cases, you should use the `copy` step to take input from an attachment.
Copy from a message to a variable.	`vars['fred'] = message`	Creates a new variable `fred` from the entire message. In most cases, you should use the `copy` step to take input from the current message.
Supply content and MIME type information to a variable whose value is an expression.	`vars['fred'] = [23 + 5, 'text/plain']`	Creates a new variable called `fred` with content `28` and MIME type `text/plain`.
Create a variable from an XPath fragment in the message root part.	`vars['fred'] = parts[0].xpathF('/dink/donk')`	Creates a new variable called `fred` with an XPath fragment from the root part of the current message. Use this method to create small messages only. Using XPath expressions to create large strings can lead to poor performance.
Nullify a variable.	`vars['fred'] = null`	Removes the variable called `fred`.
Assign content to the root part.	`parts[0] = '<hello>world</hello>'`	Assigns the content `<hello>world</hello>` to the root part. Use this method to create small messages only. Using expressions to create large strings can lead to poor performance. In most cases, you should use the `write` step to assign content to an attachment and configure the MIME type.
Assign content to the first attachment.	`parts[1] = [ 'Hello World', 'text/plain']`	Assigns the content `Hello World` and MIME type of `text/plain` to the first attachment. Use this method to create small messages only. Using expressions to create large strings can lead to poor performance.

©2026 Workday, Inc. All rights reserved

Function	Sample Expression	Description
		In most cases, you should use the `write` step to assign content to an attachment and configure the MIME type.
Assign content to the root attachment and specify the MIME type.	`parts[0] = [ 23 + 5, 'text/plain']`	Assigns root part with content `28` and MIME type `text/plain`.
Assign content from a variable to the message part.	`parts[0] = vars['fred']`	Assigns the content of the variable `fred` to a message part. In most cases, you should use the `copy` step to take input from a variable and assign it to a message part.
Remove the root part.	`parts[0] = null`	Removes the root part.
Copy an attachment.	`parts[2] = parts[1]`	Copies the first attachment into a second attachment. In most cases, you should use the `copy` step to take input from one attachment and assign it to another.
Create a new message with the default MIME type.	`context.message = '<hello>world</hello>'`	Creates a new message `<hello>world</hello>` with a default MIME type of `text/xml`. In most cases, you should use the `write` step to create a new message.
Create a new message with MIME type of `text/plain`.	`context.message = [ 'Hello World', 'text/plain' ]`	Creates a new message `Hello World` with a MIME type of `text/plain`. In most cases, you should use the `write` step to create a new message.
Create a new message from variable content.	`context.message = vars['fred']`	Creates a new message with the same content and MIME type as the variable `fred`. In most cases, you should use the `copy` step to create a new message from a variable.
Create a new message from the root part of the current message.	`context.message = parts[0]`	Creates a new message from the root part of the current message. In most cases, you should use the `copy` step to create a new message from a message part.

©2026 Workday, Inc. All rights reserved

Reference: MVEL Methods

Method	Description	Sample Code	Sample Output
`getClass().getName()`	Retrieves the type of a property or variable.	`props['p.myProperty` `props['p.myProperty ].getClass().getName()`	`java.lang.Boolean`
`getEndpoint()`	Retrieves the endpoint of a RaaS report when using a SOAP request.	`@{intsys.reportServ Alias')}`	`customreport2/ <tenant_name>/ <username>/ report_name_in_XML_format`
`getExtrapath()`	Retrieves the extra path of a RaaS report.	`intsys.reportServic workday-in Report Service Alias>')`	`customreport2/ <tenant_name>/ <username>/ <Custom Report Name>`
`indexOf()`	Retrieves the position of a specific character within an input string.	`props['input.docume = 'FoundationDataCollection/4d196113' props['index.of.separator'] = props['input.document.reference'].indexOf("/")`	`24`
`leftPad()`	Adds characters to the beginning of an input string.	`org.apache.commons. 10,'0')`	`0000001234` `leftPad('12`
`listToCommaDelimString()`	Converts a list of array values to a string and separates them with commas.	`props[p.workers] = [1234, 7890] props['p.workers'] = lp.getWIDs('Pay Group Members') props['p.workers.ids'] = props['p.workers'] ! = empty ? util.listToCommaDelimString(props['p.workers']`	`1234,7890`
`replaceAll()`	Replaces all instances of specified characters within an input string.	`props['p.employee.i = '12345pdf_a.pdf' props['p.current.id'] = props['p.employee.id'].replaceAll('[^0-9]','')`	`12345`
`substring()`	Retrieves part of an input string.	`props['p.test.strin = 'test'`	`te`

©2026 Workday, Inc. All rights reserved

Method	Description	Sample Code	Sample Output
		`props['p.test.substring'] = props['p.test.string'].substring(0,2)`	
`toUpperCase()`	Converts a string to uppercase.	`props['p.test.string'] = 'workday Studio'` `props['p.test.string'].toUpperCase()`	`WORKDAY STUDIO`

`URLEncoder ()`	Retrieves a string containing only valid URL characters.	`props['p.report.param'] = 'Full_Name='`
		`# 'Logan McNeil'`
		`props['REST.URL'] = java.net.URLEncoder.encode(props['p.report.param'], 'UTF-8')`
			`Full_Name=Logan %20McNeil`

Studio Integration Services

Create Attribute Map Services

Prerequisites

Create an assembly project.

Context

The attribute map service enables you to define attributes and maps for your integration. Attributes are configurable values that define your integration. You can create these types of attributes:

- Simple.
- Reference.
- Enumeration.

Maps reconcile internal Workday values with the external attribute values defined on your integration. You can create these types of maps:

- Simple.
- Class Report Field.
- Enumeration.

Steps

1. Open the assembly in the Assembly Editor.
2. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
3. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create attribute-map-service**.
4. On the **Service Name** window, enter a name for your attribute map service.

©2026 Workday, Inc. All rights reserved

5. On the **Integration Attributes** tab, consider:

Option	Description
Name	The name of the attribute. Must be unique across all integration systems.
Type	The attribute type definition.
Add Simple Type Attribute	Adds an attribute type that uses string literal values.
Add Reference Type Attribute	Adds an attribute type that uses references to Workday Class Report Fields (CRFs) and web services.
Add Enumeration Type Attribute	Select **Edit Type** to configure the enumeration. You can: • Use an **Enumeration Reference** to an existing enumeration definition in your tenant by entering its **Type Name**. • Create a new enumeration by entering a unique **Type Name** on the **Enumeration Definition** tab and specifying the values that it allows. Note: When you enter the **Type Name** of an existing definition on the **Enumeration Definition** tab, Workday overwrites the definition.

6. On the **Integration Maps** tab, select **Add** to add a map.
7. As you configure the **Internal Type** and **External Type**, consider:

Option	Description
Simple Type	You can select these simple types: • *Boolean* • *Date* • *Datetime* • *Number* • *Text*
Class Report Field	Select **Browse for CRF**.
Enumeration	Configure your **Enumeration Reference** or **Enumeration Definition**.

Result

Studio saves the attribute map service on the **workday-in** transport of your integration. You can access the data from the service using the MVEL `intsys` expression.

Next Steps

Deploy your integration.

Related Information
Concepts
Concept: Enumerations

©2026 Workday, Inc. All rights reserved

Tasks

Deploy Integrations to Workday
Set Up Attributes and Maps in Benefits Integration

Create Delivery Services

Prerequisites

Create an assembly project.

Context

The delivery service enables you to deliver documents to a target endpoint from your Workday tenant.

Steps

1. Open the assembly in the Assembly Editor.
2. On the **Palette**, expand the **In Transports** tab.
3. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
4. In the Assembly Editor, click the **workday-in** transport.
5. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create delivery-service**.
6. On the **Service Name** window, enter a name for your delivery service.
7. Click **Save**.

Result

Studio saves the delivery service on the **workday-in** transport of your integration.

Next Steps

Deploy your integration and configure the delivery service on your integration system in Workday.

Related Information
Tasks
Deploy Integrations to Workday
Set Up Integration Delivery

Create Listener Services

Prerequisites

Create an assembly project.

Context

The listener service enables your integration to receive launch messages from third-party clients through your Workday tenant.

Steps

1. Open the assembly in the Assembly Editor.
2. On the **Palette**, expand the **In Transports** tab.
3. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
4. In the Assembly Editor, click the **workday-in** transport.

©2026 Workday, Inc. All rights reserved Workday Proprietary and Confidential

5. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create listener-service**.
6. On the **Service Name** window, enter a name for your listener service.
7. Click **Save**.

Result

Studio saves the listener service on the **workday-in** transport of your integration.

Next Steps

Deploy your integration and configure the listener service for your integration system in Workday.

Note: You can only deploy integrations that use listener services if they have no launch parameters.

Related Information
Concepts
Concept: Listener Service API
Tasks
Deploy Integrations to Workday

Create Retrieval Services

Prerequisites

Create an assembly project.

Context

The retrieval service enables your integration to retrieve documents from remote clients and store them in your Workday tenant.

Steps

1. Open the assembly in the Assembly Editor.
2. On the **Palette**, expand the **In Transports** tab.
3. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
4. In the Assembly Editor, click the **workday-in** transport.
5. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create retrieval-service**.
6. On the **Service Name** window, enter a name for your retrieval service.
7. Click **Save**.

Result

Studio saves the retrieval service on the **workday-in** transport of your integration.

Next Steps

Deploy your integration and configure the retrieval service on your integration system in Workday.

Related Information
Tasks
Deploy Integrations to Workday
Set Up Integration Retrieval

©2026 Workday, Inc. All rights reserved Workday Proprietary and Confidential

Create Sequence Generator Services

Prerequisites

Create an assembly project.

Context

The sequence generator service enables you to generate unique, sequenced filenames for documents each time you run your integration. In Studio, you configure the sequence generator service on a **workday-in** transport. Before you can launch your integration, you must also configure the sequence generator service in your Workday tenant.

Steps

1. Open the assembly in the Assembly Editor.
2. In the Assembly Editor, click the **workday-in** transport.
3. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create sequence-generator-service**.
4. On the **Service Name** window, enter a name for your sequence generator service.
5. On the **Properties** view for the sequence generator service, click **Add** to add a sequencer.
6. On the **Sequencer** field, enter a name for your sequencer.
7. On the **Project Explorer** view, right-click your assembly project and select **Deploy to Workday**.
8. On the **Cloud Explorer** view, navigate to your deployed assembly project.
9. Right-click the integration node of your deployed assembly project.
10. Select **Workday: Integration System**.
11. On the **Workday** tab, consider:

Option	Description
Username	The username credentials for your Workday tenant.
Password	The password credentials for your tenant.

12. In your Workday tenant, on the **Integration Services** column of the **Custom Integration Services** section, locate the name of your sequence generator service and select **Related Actions and Preview**.
13. Select **Integration Sequence Generator Service > Configure Integration Sequence Generators**.
14. In the **Increment Sequence ID By** field, enter the value 1.
15. In the **Sequence ID Format** field, enter the value `file[Seq].csv`.
16. Click **OK**.
17. Click **Done**.

Result

Studio saves the sequence generator service on the **workday-in** transport of your integration. Workday saves your sequence generator settings on your integration system.

Next Steps

Launch your integration.

Related Information
Tasks
Launch Integrations from Studio

©2026 Workday, Inc. All rights reserved

Set Up Integration Sequence Generators

Create Transaction Log Services

Prerequisites

Create an assembly project.

Context

The transaction log service enables you to display the logs for transaction events that your integration has subscribed to in your Workday tenant.

Steps

1. Open the assembly in the Assembly Editor.
2. On the **Palette**, expand the **In Transports** tab.
3. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
4. In the Assembly Editor, click the **workday-in** transport.
5. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create transaction-log-service**.
6. On the **Service Name** window, enter a name for your transaction log service.
7. Click **Save**.

Result

Studio saves the transaction log service on the **workday-in** transport of your integration.

Next Steps

Deploy your integration and configure the transaction log service on your integration system in Workday.

Related Information
Tasks
Deploy Integrations to Workday

Create Report Services

Prerequisites

Create an assembly project.

Context

The report service specifies aliases for any custom reports that you refer to in your integration.

Steps

1. Open the assembly in the Assembly Editor.
2. On the **Palette**, expand the **In Transports** tab.
3. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
4. In the Assembly Editor, click the **workday-in** transport.
5. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create report-service**.
6. On the **Service Name** window, enter a name for your report service.

©2026 Workday, Inc. All rights reserved

7. On the **Properties** view for the report service, click **Add New Entry** to add a new alias. As you complete the task, consider:

Option	Description
Alias	The alias of your custom report.
Description	A text description of your custom report.
Report Reference	The name of your custom report as it displays in your Workday tenant.

8. Click **Select a RaaS Report**. As you complete the task, consider:

Option	Description
Environment	The Workday environment that contains your custom report.
Report name	The name of your custom report.

9. Click **Next**.
10. Select your custom report.
11. Click **Finish**.
12. Click **Save**.

Result

Studio saves the report service on the **workday-in** transport of your integration.

Next Steps

Deploy your integration and configure the report service for your integration system in Workday.

Related Information
Concepts
Concept: Custom Reports
Tasks
Deploy Integrations to Workday

Create Custom Object Services

Prerequisites

Create an assembly project.

Context

The custom object service specifies aliases for any custom objects that you refer to in your integration.

Steps

1. Open the assembly in the Assembly Editor.
2. On the **Palette**, expand the **In Transports** tab.
3. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
4. In the Assembly Editor, click the **workday-in** transport.
5. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create custom-object-service**.
6. On the **Service Name** window, enter a name for your custom object service.

©2026 Workday, Inc. All rights reserved

7. On the **Properties** view for the custom object service, click **Add New Entry** to add a new alias. As you complete the task, consider:

Option	Description
Alias	The alias of your custom object.
Description	A text description of your custom object.
Custom Object Reference	The name of your custom object as it displays in your Workday tenant.

8. Click **Select a Custom Object**.
9. On the **Select Custom Object** window, specify your Workday tenant environment and browse to your custom object.
10. Click **OK**.
11. Click **Save**.

Result

Studio saves the custom object service on the **workday-in** transport of your integration.

Next Steps

Deploy your integration and configure the custom object service on your integration system in Workday.

Related Information
Tasks
Deploy Integrations to Workday
Set Up Integration Custom Objects

Create Data Initialization Services

Prerequisites

Create an assembly project.

Context

The Data Initialization Service enables you to gain faster and more efficient access to Workday data in your Studio integrations.

Steps

1. Open the assembly in the Assembly Editor.
2. On the **Palette**, expand the **In Transports** tab.
3. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
4. In the Assembly Editor, click the **workday-in** transport.
5. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create data-initialization-service**.
6. On the **Service Name** window, enter a name for your data initialization service.
7. Click **Save**.

Result

Studio saves the data initialization service on the **workday-in** transport of your integration.

©2026 Workday, Inc. All rights reserved
Workday Proprietary and Confidential

Next Steps

Deploy your integration and configure the data initialization service for your integration system in Workday.

Create Service References

Prerequisites

Create an assembly project.

Context

Service references enable you to reference any integration service configured on another **workday-in** transport within the same collection.

Steps

1. Open the assembly in the Assembly Editor.
2. On the **Palette**, expand the **In Transports** tab.
3. Drag a **workday-in** transport from the **Palette** to the Assembly Editor.
4. In the Assembly Editor, click the **workday-in** transport.
5. On the **Services** tab of the **Properties** view for the **workday-in** transport, click **Create service-reference**.
6. On the **Service Reference** window, select a service reference.
7. Click **OK**.
8. Click **Save**.

Result

Studio saves the service reference on the **workday-in** transport of your integration.

Next Steps

Deploy your integration.

Related Information
Tasks
Deploy Integrations to Workday

Add References to Custom Reports on Workday-Out-Rest Transports

Prerequisites

Create an assembly project that contains a **workday-out-rest** transport.

Context

The **workday-out-rest** transport enables you to send REST request messages to an HTTP URL. You can add a reference to a custom report as an extra path value on a **workday-out-rest** transport.

Steps

1. Open the assembly in the Assembly Editor.
2. In the Assembly Editor, click the **workday-out-rest** transport.
3. On the **Common** tab of the **Properties** view for the **workday-out-rest** transport, click **Select a RaaS Report**.

©2026 Workday, Inc. All rights reserved

4. On the **Add Custom RaaS Report** window, consider:

Option	Description
Environment	The Workday environment that contains your custom report.
Report name	The name of your custom report.

5. On the **Select Reports** window, select your custom report.
6. On the **Report Service Configuration** window, consider:

Option	Description
Report Alias	The alias of your custom report.
Description	A text description of your custom report.

7. Click **Finish**.

Result

Studio adds a reference to your custom report on the **workday-out-rest** transport of your integration.

Next Steps

Launch your integration.

Related Information
Tasks
Launch Integrations from Studio

Add References to Custom Objects on Workday-Out-Rest Transports

Prerequisites

Create an assembly project that contains a **workday-out-rest** transport.

Context

The **workday-out-rest** transport enables you to send REST request messages to an HTTP URL. You can add a reference to a custom object as an extra path value on a **workday-out-rest** transport.

Steps

1. Open the assembly in the Assembly Editor.
2. In the Assembly Editor, click the **workday-out-rest** transport.
3. On the **Common** tab of the **Properties** view for the **workday-out-rest** transport, click **Select a Custom Object**.
4. On the **Add Custom Object** window, consider:

Option	Description
Environment	The Workday environment that contains your custom object.
HTTP Methods	Options are: • **POST** • **PUT** • **GET** • **DELETE**

©2026 Workday, Inc. All rights reserved

5. On the **Select Custom Object** window, consider:

Option	Description
Select an existing custom object alias	Displays a list of existing custom object aliases.
Create a new custom object alias	Enables you to create a new custom object alias.

   Skip to step 7 if you selected the **PUT**, **GET**, or **DELETE** HTTP methods.
6. If you selected the **POST** HTTP method, Studio displays the **POST Query Parameters** window. As you complete the task, consider:

Option	Description
updateIfExists	Updates an existing custom object.
bulk	Creates or updates a collection of custom objects.

7. If you selected the **PUT**, **GET**, or **DELETE** HTTP methods, select a custom object id reference from the **Custom Object Id Reference Selection** window.
8. Click **Finish**.

Result

Studio adds a reference to your custom object on the **workday-out-rest** transport of your integration.

Next Steps

Launch your integration.

Related Information
Tasks
Launch Integrations from Studio

CLAR Files

Generate Manifest Files

Context

Manifest files enable you to export assembly collections as CLAR files. You must save manifest files to a project folder in an assembly collection. Manifest files contain the metadata of your assembly collections.

Steps

1. On the Studio menu bar, select **Workday > Solution > Generate Workday Manifest**.
2. Double-click the target project folder for the manifest file.

Result

Studio saves the manifest file to the project folder.

Next Steps

Export the assembly collection as a CLAR file.

©2026 Workday, Inc. All rights reserved

Import CLAR Files

Prerequisites

Save a CLAR file to your file system.

Context

You can add assemblies to your workspace by importing CLAR files.

Steps

1. On the Studio menu bar, select **File > Import...**.
2. Select the **CLAR file** import wizard.
3. Click **Browse** and select a CLAR file to import.

   If your selected CLAR file contains projects with external tool builder commands, Studio informs you that those commands will be removed by default when importing the CLAR.
4. Click **Next** to display the collections and projects in the CLAR file.

   If your selected CLAR contains external tool builder commands that you want to keep, clear the **Remove all external tool builder commands from projects on import if present** checkbox.
5. Select collections and projects to import from the CLAR file.

   Studio automatically selects every collection and project in the CLAR file. Modify your selection using the check boxes. Studio displays decorators on collections and projects with name conflicts.
6. Edit all project name conflicts.
   a) Select a project with a name conflict.
   b) Click **Edit...**.
   c) In the **Project Name** field, rename the project.
7. Edit all collection name conflicts.
   a) Select a collection with a name conflict.
   b) Click **Edit...**.
   c) In the **Collection Name** field, rename the collection.
8. Click **Finish**.

Result

Studio imports the contents of the CLAR file to your workspace.

Export CLAR Files

Prerequisites

- Create an assembly project.
- Generate a manifest file.

Context

You can export assemblies from your workspace as CLAR files. You can use CLAR files to share your assemblies with other Studio integration developers. Example: by uploading them to the Contributed Solutions page on Workday Community.

Steps

1. On the Studio menu bar, select **File > Export...**.

©2026 Workday, Inc. All rights reserved

2. Select the **CLAR file** export wizard.
3. On the **CLAR File Specification** window, select a collection to export as a CLAR file.
4. On the **Workday Manifest Specification** window, select **Use existing manifest from collection**.
5. On the **Export Destination** window, specify the target destination of the CLAR file.
6. Click **Finish**.

Result

Studio exports the assemblies as a CLAR file.

Related Information
Reference
All Contributed Solutions

Concept: CLAR Files

CLAR Files

A CLAR (Cloud Archive) file is a package file that stores assembly collections, projects, and metadata. CLAR files enable you to import and export assemblies through your workspace. You can use CLAR files to share your assemblies with other Studio integration developers.

Manifest Files

Manifest files enable you to export assembly collections as CLAR files. You must save manifest files to a project folder in an assembly collection. Manifest files contain the metadata of your assembly collections.

Contributed Solutions

The Contributed Solutions page is an online repository that enables you to access and share CLAR files. You can access the Contributed Solutions page on Workday Community.

JAXB Projects

Create JAXB Projects

Prerequisites

Save an XML schema file to your file system.

Context

JAXB is a binding framework that enables you to convert data between XML and Java formats. In Studio, you can convert XML schema files into Java classes by creating a JAXB project. Once you create a JAXB project, you can implement its Java classes in an integration by adding it as a dependency in an assembly project.

Steps

1. Select **File > New > Workday Schema to Java (JAXB) Project**.
2. On the **Project Configuration Details** window, consider:

Option	Description
Project name	The name of your project.

©2026 Workday, Inc. All rights reserved

Option	Description
Target runtime	The target runtime server for your project. Automatically populates as *Workday Runtime*.
Configuration	The facet configuration of your project. Automatically populates as *Workday Schema to Java (JAXB) Project*.

3. On the **Source Folders** window, consider:

Option	Description
Generation directory	The directory folder for your JAXB generated Java classes. Automatically populates as *jaxb-gen*.
Mark generated resources as derived	Marks your JAXB generated Java classes as derived resources. Studio automatically selects this check box.

4. On the **JAXB Schema Selection and Generation Settings** window, consider:

Option	Description
Schema Files	The location of your XML schema files. Studio displays these options: • **Workspace** • **External** • **URL**
JAXB External Binding Files (optional)	The location of any binding files for your XML schemas. Studio displays these options: • **Workspace** • **External**
Generate default binding file for unmapped schemas	Creates generic binding files for your XML schemas.
Location within Project	The directory folder for generic binding files. Automatically populates as `jaxb-resources/bindings.xjb`.
Choose how to manage Schema and External JAXB Binding files	Select **Create links in the project to external Schema/Binding files** to create links to your schema and binding files in your new JAXB project. Studio automatically selects this option. Select **Copy Schema/Binding files to project now** to copy your schema and binding files into your new JAXB project.
Compiler Options	Studio displays these options: • **Specify target package (overrides all other package definitions)** • **Apply less strict validation to schemas** • **Suppress package level annotations** • **Automatic name conflict resolution**

©2026 Workday, Inc. All rights reserved

5. Click **Finish**.

Result

Studio displays the JAXB project on the **Project Explorer** view.

Next Steps

Add the JAXB project as a dependency in an assembly project.

Add JAXB Projects as Dependencies in Assembly Projects

Prerequisites

- Create an assembly project.
- Create a JAXB project.

Context

JAXB is a binding framework that enables you to convert data between XML and Java formats. In Studio, you can implement JAXB Java classes in your integrations by adding JAXB projects as dependencies in assembly projects.

Steps

1. On the **Project Explorer** view, right-click an assembly project.
2. Select **Build Path > Configure Build Path...**.
3. On the **Java Build Path** window, select the **Projects** tab.
4. Click **Add...**.
5. Select a JAXB project to add to your assembly project.
6. Click **OK**.
7. Click **Apply and Close**.

Result

Studio adds the JAXB project as a dependency in the assembly project. You can now implement your JAXB Java classes in integrations stored within that assembly project.

Next Steps

Deploy your integration.

Concept: JAXB Support in Studio

JAXB is a binding framework that enables you to convert data between XML and Java formats.

JAXB Projects

In Studio, you can convert XML schema files into Java classes by creating a JAXB project. Once you create a JAXB project, you can implement its Java classes in an integration by adding it as a dependency in an assembly project.

JAXB Mediation Steps

In your assembly projects, you can invoke JAXB on the **Properties** view of these mediation steps:

- **java-to-xml**

©2026 Workday, Inc. All rights reserved

- **xml-to-java**

Studio Ivy Plugin

Configure the Ivy Plugin

Context

Workday Studio supports an Apache Ivy™ plugin that enables you to manage project dependencies. You can configure the Ivy plugin as a preference in Studio.

Steps

1. Click **Window > Preferences > Workday > Ivy**.
2. On the **Configure Ivy Engine** window, consider:

Option	Description
Ivy settings URL	The URL for your `ivysettings.xml` file.
Ivy Properties	The URL to your `ivyproperties.xml` file.
Load ivy.properties file from home directory if present	Loads the `ivy.properties` file automatically from your home directory.

Result

Studio applies your Ivy plugin preferences to your assembly projects.

Add Workday Ivy Natures on Workday Studio Projects

Prerequisites

- Configure the Apache Ivy™ plugin for Workday Studio.
- Create an assembly project.

Context

Workday Studio supports an Apache Ivy plugin, which enables you to manage project dependencies by configuring projects with the Workday Ivy nature.

Steps

1. On the **Project Explorer** view, right-click an assembly project.
2. Select **Configure > Add Workday Ivy Nature**.
3. Expand the assembly project folder.
4. Open the `ivy.xml` file.
5. Expand the **info** node. As you complete the task, consider:

Option	Description
Module	The module name.
Add your content here	Insert a parent `<dependencies>` element, enclosing one or more child `<dependency>` elements.

©2026 Workday, Inc. All rights reserved

Result

Studio adds a Workday Ivy nature to the project.

Resolve Dependencies for Projects with Ivy Definitions

Prerequisites

- Configure the Apache Ivy™ plugin for Workday Studio.
- Configure an assembly project with the Workday Ivy nature.

Context

Workday Studio supports an Apache Ivy plugin, which enables you to manage project dependencies by configuring projects with the Workday Ivy nature. If your project displays a build error relating to missing dependencies, you can resolve these dependency issues.

Steps

1. On the **Project Explorer** view, right-click an assembly project with the Workday Ivy nature.
2. Select **Resolve**.

Result

The **Console** view displays a resolution report.

Add Dependent JARs to Projects with Ivy Natures

Prerequisites

- Configure the Apache Ivy™ plugin for Workday Studio.
- Configure an assembly project with the Workday Ivy nature.

Context

Workday Studio supports an Apache Ivy plugin, which enables you to manage project dependencies by configuring projects with the Workday Ivy nature. If a Studio project has an Ivy nature with dependency elements in its `ivy.xml` file, you can configure the project to include any dependent JAR files.

Steps

1. On the **Project Explorer** view, right-click an assembly project with the Workday Ivy nature.
2. Select **Properties**.
3. On the **Java Build Path** window, click **Libraries**.
4. Click **Ivy-Workday IDE**.
5. Click **Add JARs**.
6. On the **JAR Selection** window, select the JAR files to add to your project and click **OK**.
7. On the **Java Build Path** window, click **Apply and Close**.

Result

Studio adds the dependent JAR files to your project.

Next Steps

Deploy your project.

©2026 Workday, Inc. All rights reserved

Remove Workday Ivy Natures from Studio Projects

Prerequisites

- Configure the Apache Ivy™ plugin for Workday Studio.
- Configure an assembly project with the Workday Ivy nature.

Context

Workday Studio supports an Apache Ivy plugin, which enables you to manage project dependencies by configuring projects with the Workday Ivy nature. You can add and remove the Workday Ivy nature from projects as required.

Steps

1. On the **Project Explorer** view, select an assembly project folder.
2. Select **Workday > Ivy > Remove Workday Ivy Nature**.

Result

Studio removes the Workday Ivy nature from your project.

Next Steps

Deploy your project.

Integration Deployment

Add a Connection

Context

In Studio, you can create and manage connections to Workday. You can use these connections when you deploy and launch assemblies.

Steps

1. Select **Windows > Preferences > Workday > Connections**.
2. Click **Add...**.
3. On the **Add Connection** window, consider:

Option	Description
Name	The name of the connection.
URL	The URL of the connection.
Group	Adds the connection to a group.

©2026 Workday, Inc. All rights reserved

4. In the **Credentials** section, specify the **Tenant** and select an authentication method, **Basic Auth** or **OAuth 2.0**.

   Note: To use OAuth 2.0 authorization, you must first register an API client. Users of that client who are enrolled in Workday multifactor authentication are required to use a second authentication method when connecting to Workday from Studio.

Option	Description
Basic Auth	Specify the **Username** and **Password** for the connection.
OAuth 2.0	Specify these details for the connection: • **Grant Type** • **Client ID** • **Authorization Endpoint** • **Token Endpoint** • **Access Token**

5. Click **Test Connection**.

   Studio confirms whether it can access the connection.

Result

The connection displays on the **Connections** preferences page.

Next Steps

Build an assembly.

Related Information
Tasks
Enable Multifactor Authentication in Workday Studio

Add a Group of Connections

Context

In Studio, you can create and manage connections to Workday. You can organize these connections into groups. You can use these connections when you deploy and launch assemblies.

Steps

1. Select **Windows > Preferences > Workday > Connections**.
2. Click **Add Group...**.
3. On the **Add Group** window, consider:

Option	Description
Name	The name of the connection group.
Filtered	Filters the connection group out of the **Cloud Explorer** view and the **Deploy to Workday** page.

Result

The connection group displays on the **Connections** preferences page.

©2026 Workday, Inc. All rights reserved

Next Steps

Build an assembly.

Edit a Connection

Prerequisites

Add a connection to Workday.

Context

In Studio, you can create and manage connections to Workday. You can use these connections when you deploy and launch assemblies.

Steps

1. Select **Windows > Preferences > Workday > Connections**.
2. Select a connection.
3. Click **Edit Environment URL**. As you complete the task, consider:

Option	Description
Host	The host id of the connection.
URL	The URL of the connection.

Result

The connection displays on the **Connections** preferences page.

Next Steps

Build an assembly.

Edit a Group of Connections

Prerequisites

Create a group of connections to Workday.

Context

In Studio, you can create and manage connections to Workday. You can organize these connections into groups. You can use these connections when you deploy and launch assemblies.

Steps

1. Select **Windows > Preferences > Workday > Connections**.
2. Select a connection group.
3. As you complete the task, consider:

Option	Description
Name	The name of the connection group.

©2026 Workday, Inc. All rights reserved

Option	Description
Filtered	Filters the connection group out of the **Cloud Explorer** view and the **Deploy to Workday** page.

Result

The connection group displays on the **Connections** preferences page.

Next Steps

Build an assembly.

Remove Connections

Prerequisites

Add a connection to Workday.

Context

In Studio, you can create and manage connections to Workday. You can use these connections when you deploy and launch assemblies.

Steps

1. Select **Windows > Preferences > Workday > Connections**.
2. Select a connection.
3. Click **Remove...**.

Result

Studio removes the connection from the **Connections** preferences page.

Next Steps

Build an assembly.

Import Connections

Prerequisites

Access an XML file containing Workday connection details.

Context

In Studio, you can create and manage connections to Workday. You can use these connections when you deploy and launch assemblies.

Steps

1. Select **Windows > Preferences > Workday > Connections**.
2. Click **Import...**.

©2026 Workday, Inc. All rights reserved

3. Click **Browse...**.
4. Select a Workday connection XML file.

Result

Studio displays the connection on the **Connections** preferences page.

Next Steps

Build an assembly.

Export Connections

Prerequisites

Add a connection to Workday.

Context

In Studio, you can create and manage connections to Workday. You can use these connections when you deploy and launch assemblies.

Steps

1. Select **Windows > Preferences > Workday > Connections**.
2. Click **Export...**.
3. Select a connection.
4. Click **Browse...**
5. Select a target destination for the Workday connection XML file.

Result

Studio exports the connection to the target destination.

Next Steps

Build an assembly.

Connect to Workday

Prerequisites

Add a connection to Workday.

Context

In Studio, you can create and manage connections to Workday. You can use these connections when you deploy and launch assemblies.

Steps

1. From the Studio **Preferences** menu, select **Workday > Connections**.

©2026 Workday, Inc. All rights reserved

2. Select a connection. As you complete the task, consider:

Option	Description
Tenant	Specify the name of your tenant.
Username	Enter the username credentials for your tenant.
Password	Enter the password credentials for your tenant.

3. Click **Test Connection**.
   Studio confirms whether it can access your tenant.
4. Click **Apply and Close**.

Result

Studio connects to your Workday tenant.

Next Steps

Deploy an assembly to Workday.

Deploy Integrations to Workday

Prerequisites

- Create an assembly project.
- Security: *Integration Build* and *Integration Configure* domains in the Integration functional area.

Context

In Studio, you can deploy your integrations to Workday. Once you deploy an integration, you can launch it in your Workday tenant.

Steps

1. On the **Project Explorer** view, right-click an assembly project.
2. Select **Deploy to Workday**.
3. On the **Deploy to Workday** window, consider:

Option	Description
Workday Environments	Select a target Workday environment for deploying your integration.
Collection being deployed	Select a collection to deploy to Workday.

4. Click **Finish**.

Result

Studio deploys your integration to Workday.

Next Steps

Launch your integration.

©2026 Workday, Inc. All rights reserved

View Deployed Integrations in Workday

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Build* and *Integration Configure* domains in the Integration functional area.

Context

In Studio, you can deploy your integrations to Workday. You can view your deployed integrations when you sign in to your Workday tenant.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Workday: Integration System**.
5. On the **Workday** tab, sign in to your Workday tenant. As you complete the task, consider:

Option	Description
Username	Enter the username credentials for your Workday tenant.
Password	Enter the password credentials for your tenant.

Result

Your deployed integration displays on the **Workday** tab in Studio.

Next Steps

Launch your integration.

Configure Studio to Use a HTTPS Proxy Server

Context

If you need to connect Workday Studio to a training or test environment on a port other than 443 and your organization's firewall doesn't allow that by default, you might be able to connect using a proxy server in your network.

Steps

1. Select **Windows > Preferences > General > Network Connections**.
2. From the **Active Provider** drop-down menu, select *Manual*.

   Note: When you select *Manual*, the checkboxes for all 3 **Schemas** in the **Proxy entries** table, HTTP, HTTPS, and SOCKS, are selected. You can't clear these selections individually.
3. Next to the HTTPS **Schema**, enter a **Host** IP address and a **Port** number for the proxy server.
4. Click **Apply and Close**.

©2026 Workday, Inc. All rights reserved

Concept: Studio Connections

In Studio, you can create and manage connections to Workday. You can use these connections when you deploy and launch assemblies.

Note: All connection names must be unique. Studio provides **Sandbox** and **Production** as default connections. You can't rename these connections. For security reasons, Studio doesn't save username and password details to your workspace after you exit from your Studio session.

Concept: Cloud Explorer

The **Cloud Explorer** view displays your cloud server connections. You can expand each cloud server node to view its collections. You can expand each collection node to view integrations available for launch.

Note: The **Cloud Explorer** view displays a maximum of 999 collections.

The **Cloud Explorer** view uses the same date formatting settings as your local operating system.

Reference: Connection Errors

Connection Error	Notes
Your user login credentials failed to authenticate.	Check your credentials, then try again. Remember that tenant names are case-sensitive.

Your tenant is not available.	Use a web browser to check that your tenant is online. Enter the tenant URL in the form:
	`https://host-name/tenant-name/ccx/cc-cloud-repo/collections`
	Example:
	`https://wd5.myworkday.com/gms/ccx/cc-cloud-repo/collections`
	Use the same credentials you're trying to use in Studio but combine your user and tenant names with an @ character. Example:
	`lmcneil@gms`
	If the connection succeeds, the browser displays all Studio integration collections that are deployed in your tenant. Next, check your browser's connection preferences to see whether it's making a direct connection or going through a proxy server. If it's configured to use a proxy server, then you'll need to configure Studio to also use that proxy.

Permission to deploy denied.	To deploy integrations to Workday, you must be a member of a security group with *Modify* permissions in the *Integration Build* domain. Contact your Workday System Administrator.

©2026 Workday, Inc. All rights reserved

Connection Error	Notes
Permission to launch denied.	To launch integrations from Studio, you must be a member of a security group with *Put* permissions in one of these domains: • *Integration Event* • *Integration Build* • *Integration Debug* Contact your Workday System Administrator.
Permission to deploy denied (integration contains a delivery or retrieval service).	To deploy an integration that contains a delivery or retrieval service, you need an additional permission. You must be a member of a security group that has *Put* permission in one of these domains: • *Business Process Administration* • *Manage: Business Process Definitions* • *iLoad Web Services* Contact your Workday System Administrator.
Access to My Reports denied.	Workday controls access to My Reports integration documents using configurable security. The Workday account you used to access the integration document doesn't have the required permission. Contact your Workday System Administrator to request the required permission.
SOAP fault: A validation error has occurred.	Your SOAP request for the web service operation is invalid. Verify that your request contains valid data elements and values. Use the **Add Latest Workday Web Services** wizard in the **Schema Explorer** to import valid WSDLs into your Workday Studio environment. Then use the **Create and Copy SOAP Literal** or **Open SOAP wrapped request in Web Service Tester** menus to create a valid SOAP request.
Your IP address failed to authenticate.	Your IP address isn't authorized to access the Workday environment. Contact your System Administrator.

Multifactor Authentication in Workday Studio

Enable Multifactor Authentication in Workday Studio

Context

You can configure Workday Studio to use OAuth 2.0 authorization when it connects to a tenant. To do so, you must register an API client. Users of the client who are enrolled in Workday multifactor authentication in a tenant are required to use a second authentication method when connecting to that tenant from Studio. They can't use basic authentication.

©2026 Workday, Inc. All rights reserved

Steps

1. Set up multifactor authentication in your tenant and specify which accounts are required to use it.
   See Steps: Set Up Multifactor Authentication Using Authenticator App.
   See Steps: Set Up Multifactor Authentication Using Duo Security.
   See Steps: Set Up Multifactor Authentication Using Emailed One-Time Passcode.
   See Steps: Set Up Multifactor Authentication Using SMS One-Time Passcode.
2. Register an OAuth 2.0-enabled API client.
   Configure these settings:

Option	Value
Client Grant Type	Authorization Grant Code
Support Proof Key for Code Exchange (PKCE)	Enabled
Access Token Type	Bearer
Redirection URI	Enter the URI provided by your organization as part of the authentication process. The URI must be reachable and encrypted by HTTPS.
Scope	In the Custom Objects list, select the functional area for every web service that you'll access. In addition select these functional areas: • Implementation • Integration • System • Tenant Non-Configurable Note: Authentication fails if you omit any relevant web service.
Include Workday Owned Scope	Enabled

   Note: Don't select the **Enforce 60 Minute Access Token Expiry** check box. Doing so disables OAuth 2.0 authentication after that period. OAuth 2.0 authentication is also subject to the normal Workday session timeouts configured tenant-wide or for individual accounts.

   See Register API Clients.
3. Copy the **Client ID**, **Authorization Endpoint**, and **Token Endpoint** provided by the Register API Client task.
4. From the Workday Studio **Preferences** menu, navigate to **Workday > Connections**.
5. Add a tenant connection and select the **OAuth 2.0** credentials tab.
6. Paste in the **Client ID**, **Authorization Endpoint**, and **Token Endpoint**.
7. Click **Get Access Token**.
   If you receive an Internal Server Error message, the most likely cause is an incorrect Client ID.
8. Enter a username and password for the tenant.
   If you're enrolled in Workday Multifactor Authentication, you're prompted to provide a second authentication method.
9. Test the connection.
10. Click **Apply and Close**.

©2026 Workday, Inc. All rights reserved

Related Information
Reference
2021R2 What's New Post: Multifactor Authentication in Workday Studio

Integration Configuration

Assign an Integration System User Account to an Integration

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Security* domain in the Integration functional area.

Context

Workday Studio integrations require an integration system user (ISU) account for authentication and web service calls.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Assign Integration System User**.
5. Click **Integration System Users**.
6. Select an ISU from the right window pane.

Result

You can launch your integration as an ISU.

Next Steps

Launch your integration.

Concept: Integration System Security

In Workday, you can join security groups that enable you to perform specific tasks in your tenant. In Studio, when you connect to your Workday tenant, the **Test Connections** window displays 3 security groups:

- **Deploy**
- **Launch**
- **Business Process**

As a minimum, Studio requires you to be a member of at least 1 of these security groups before you can run an integration.

©2026 Workday, Inc. All rights reserved

Integration Launching

Launch Integrations from Studio

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Studio enables you to launch integrations in your Workday tenant.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.

Result

Your integration launches in your Workday tenant. In Studio, the **Process Monitor** tab displays the status of your integration.

Related Information
Tasks
Deploy Integrations to Workday

Copy Launch URLs

Prerequisites

Deploy an integration to Workday.

Context

Studio enables you to copy the launch URLs of integrations in your Workday tenant. You can use these URLs to launch integrations from third-party REST-based clients with basic authentication privileges.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Copy Launch URL**.
5. On the **Launch URL** window, highlight the launch URL and press Ctrl + C.

©2026 Workday, Inc. All rights reserved

Result

Studio copies the launch URL of your integration to your clipboard.

Next Steps

Use the launch URL in REST requests from third-party REST-based clients with basic authentication privileges.

Related Information
Tasks
Deploy Integrations to Workday

Create Launch Configurations

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Studio enables you to launch integrations in your Workday tenant. You can create launch configurations for your integrations.

Steps

1. Select **Run > Run Configurations**.
2. On the left navigation pane, right-click **Integration**.
3. Click **New**.
4. On the right navigation pane, consider:

Option	Description
Name	The name of your launch configuration.
Launch In	The target Workday environment for launching your integration.
Collection	The collection to launch in your integration.
Project	The project to launch in your integration.
Workday-In	The name of the **Workday-In** transport in your integration.
Launch Parameters	The launch parameters of your integration.

5. Click **Apply**.
6. Click **Run**.

Result

Studio runs your integration with your new launch configuration details.

Related Information
Tasks
Deploy Integrations to Workday

©2026 Workday, Inc. All rights reserved

Edit Launch Configurations

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Studio enables you to launch integrations in your Workday tenant. You can edit the launch configurations of your integrations.

Steps

1. Select **Run > Run Configurations**.
2. On the left navigation pane, expand the **Integration** node.
3. Select a launch configuration node.
4. On the right navigation pane, consider:

Option	Description
Name	The name of your launch configuration.
Launch In	The target Workday environment for launching your integration.
Collection	The collection to launch in your integration.
Project	The project to launch in your integration.
Workday-In	The name of the **Workday-In** transport in your integration.
Launch Parameters	The launch parameters of your integration.

5. Click **Apply**.
6. Click **Run**.

Result

Studio runs your integration with your new launch configuration details.

Related Information
Tasks
Deploy Integrations to Workday

Integration Monitoring and Troubleshooting

Process Monitor

Concept: Integration Event IDs

Workday assigns an ID to all integration events, enabling you to track integration events from launch to completion. When you launch an integration from Workday Studio, the integration event ID displays on the **Id** column of the **Process Monitor** view.

©2026 Workday, Inc. All rights reserved

Reference: Process Monitor Toolbar Buttons

Button	Description
Refresh	Refreshes the **Process Monitor** view.
Delete the Process	Deletes a process from the **Process Monitor** view.
Copy the Process ID to the Clipboard	Copies an integration event ID to your clipboard.
Cancel the Process	Cancels a process.
View Consolidated Report	Opens a consolidated report in the **Consolidated Report Viewer**.
View Log File	Displays the log file for a process in the **View Log File** window.
View Request Message	Displays the request message for a process in the **View Request Message** window.
Import Consolidated Report	Opens the **Import Consolidated Report** window.
Import Log File	Opens the **Import Log File** window.
Import Request Message	Opens the **Import Request Message** window.
Enable Auto Refresh	Enables auto-refresh for the **Process Monitor** view.
Workday: Process Monitor	Opens the Workday **Process Monitor** on the **Workday** tab.
Workday	Opens the **Workday** view and displays the integration server ESB invocation for your integration.
View Menu	Displays the toolbar options for the **Process Monitor** view.

Consolidated Report Viewer

Load Consolidated Reports for Integrations

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

In Studio, the **Consolidated Report Viewer** tab displays performance information related to the timing, memory usage, and number of calls per step in your integration. Consolidated reports enable you to analyze the process flow of your integrations.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.

©2026 Workday, Inc. All rights reserved

3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.

Result

Your consolidated report displays on the **Consolidated Report Viewer** tab.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

Save Consolidated Reports to Your Workspace

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Consolidated reports enable you to analyze the process flow of your integrations. You can save consolidated reports to your workspace.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.
8. Right-click the consolidated report on the **Consolidated Report Viewer** tab.
9. Select **Save As...**.
10. On the **Save Consolidated Report** window, select a parent folder for your consolidated report.
11. Click **Finish**.

Result

Studio saves your consolidated report to your workspace.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

©2026 Workday, Inc. All rights reserved

View Workday Integration Event Details Using Consolidated Reports

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Consolidated reports enable you to analyze the process flow of your integrations. You can view your Workday integration event details using consolidated reports in Studio.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.
8. On the **Consolidated Report Viewer** tab, select **View in Workday**.
9. On the **Workday** tab, sign in to your Workday tenant. As you complete the task, consider:

Option	Description
Username	The username credentials for your Workday tenant.
Password	The password credentials for your tenant.

Result

Your Workday integration event details display on the **Workday** tab in Studio.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

View Request Messages for Integration Events Using Consolidated Reports

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Consolidated reports enable you to analyze the process flow of your integrations. You can view the request messages for your integration events using consolidated reports in Studio.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.

©2026 Workday, Inc. All rights reserved

3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.
8. On the **Consolidated Report Viewer** tab, select **View Request Message**.

Result

The request message for your integration event displays on the **View Request Message** window.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

Monitor Integrations with the Consolidated Report Viewer Events Log

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Consolidated reports enable you to analyze the process flow of your integrations. You can monitor your integrations using the events log in the **Consolidated Report Viewer** tab in Studio.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.
8. On the **Consolidated Report Viewer** tab, select **Events Log**.

Result

The events log for your integration displays on the **Consolidated Report Viewer** tab.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

Monitor Integrations with the Consolidated Report Viewer Profile Log

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

©2026 Workday, Inc. All rights reserved

Context

Consolidated reports enable you to analyze the process flow of your integrations. You can monitor your integrations using the profile log in the **Consolidated Report Viewer** tab in Studio.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.
8. On the **Consolidated Report Viewer** tab, select **Profile Log**.

Result

The profile log for your integration displays on the **Consolidated Report Viewer** tab. The time zone used is PST.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

Replay Assembly Executions for Consolidated Reports

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Consolidated reports enable you to analyze the process flow of your integrations. You can replay assembly executions for your integrations using consolidated reports in Studio.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.
8. On the **Consolidated Report Viewer** tab, select **Replay**.

Result

The assembly execution for your integration displays on the Assembly Editor.

©2026 Workday, Inc. All rights reserved
Workday Proprietary and Confidential

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

Link Consolidated Report Audit Events to Assembly Elements

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Consolidated reports enable you to analyze the process flow of your integrations. You can link consolidated report audit events to assembly elements in Studio.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.
3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.
8. On the **Consolidated Report Viewer** tab, select **Link with editor**.

Result

Studio links your consolidated report audit events to assembly elements in your integration. When you select a node on the **Consolidated Report Viewer** tab, Studio highlights the corresponding assembly element on your assembly diagram.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

Display Consolidated Report Annotations on Assembly Diagrams

Prerequisites

- Deploy an integration to Workday.
- Security: *Integration Event* domain in the Integration functional area.

Context

Consolidated reports enable you to analyze the process flow of your integrations. You can display consolidated report annotations on your assembly diagram in Studio.

Steps

1. On the **Cloud Explorer** view, select and expand a cloud server node.
2. Select and expand a collection node.

©2026 Workday, Inc. All rights reserved

3. Right-click an integration node.
4. Select **Launch Integration**.
5. Click **Launch**.
6. On the **Process Monitor** view, right-click your integration.
7. Select **View Consolidated Report**.
8. On the **Consolidated Report Viewer** tab, select **Annotate**.

Result

Studio displays consolidated report annotations on your assembly diagram in the Assembly Editor.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

Concept: Consolidated Report Viewer

In Workday Studio, the **Consolidated Report Viewer** tab displays performance information related to the timing, memory usage, and number of calls per step in your integration. The **Consolidated Report Viewer** tab also summarizes Workday Web Service calls and highlights integration errors.

You can link consolidated reports to assemblies. You can also display consolidated report annotations on each assembly element in your diagram, enabling you to spot issues and errors in your integrations.

The **Consolidated Report Viewer** tab displays as part of the default Workday Studio perspective. If you close it, you can reopen it by selecting **Window > Show View > Consolidated Report Viewer**.

You can load consolidated reports from within the **Consolidated Report Viewer** tab, or from the **Process Monitor** view for a selected integration process.

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

Reference: Consolidated Report Error Messages

Error Message	Example	Description
java.lang.Exception: Win on E2: XSLT job failed	`com.capeclear.mediation.MediationException: Custom step id=WaitJobComplete, ref=WaitJobCompleteBean - Error while invoking bean, reason: Win on E2: XSLT job failed. The Job with referenceId <referenceid> has failed as it exceeded the maximum number of retry attempts, 1 Root Cause: java.lang.Exception:`	Displays when your job has exceeded the maximum number of retry attempts.

©2026 Workday, Inc. All rights reserved

Error Message	Example	Description
	Win on E2: XSLT job failed.	
HttpAuthenticationRequiredExce Unauthorized	com.capeclear.mediation.MediationException: RetrieveStep id=null encountered a problem sending data to endpoint <endpoint>, reason:HttpAuthenticationRequiredException: Unauthorized.	Displays when you have insufficient privileges to access an attachment.
com.workday.esb.blobitory.Blobi Tenant wdoms not active	com.capeclear.mediation.MediationException: application=bpelEnterpriseInterfaceBuilder: <version-number> - Failed to store resource in collection eib_xslt_collection in workspace wdoms.	A tenant configuration error. Contact Support and Technical Operations to ensure that the tenant is active.
java.lang.OutOfMemoryError: Java heap space.	com.capeclear.mediation.MediationException: Custom step id=WaitJobComplete, ref=WaitJobCompleteBean - Error while invoking bean, reason: Win on E2: XSLT job failed. Integration Failed. com.capeclear.mediation.MediationException: Xslt step id=transformEIB failed to apply stylesheet mctx:vars/eib.xslt.	Displays when your integration contains poorly written XSLT.
java.xml.transform.TransformCor	com.capeclear.mediation.MediationException: Custom step id=WaitJobComplete, ref=WaitJobCompleteBean - Error while invoking bean, reason: Win on E2: XSLT job failed. Integration Failed. com.capeclear.mediation.MediationException: Xslt step id=transformEIB stylesheet mctx:vars/ eib.xslt located at mctx:vars/eib.xslt is invalid or missing, reason: javax.xml.transform.TransformerConfigurationException: Failed to compile stylesheet. 1 error detected. Error reported by	Displays when your integration contains invalid XSLT.

©2026 Workday, Inc. All rights reserved

Error Message	Example	Description
	XML parser org.xml.sax.SAXParseException; systemId: mctx:vars/eib.xslt; lineNumber: 960; columnNumber: 5; The markup in the document following the root element must be well-formed. Root Cause: javax.xml.transform.TransformerConfigurationException: Failed to compile stylesheet. 1 error detected.	
com.workday.services.server.cap	com.capeclear.mediation.MediationException: application=bpelACHTransformOutbound-21.0.06.267 - Custom step id=aggregateElement, ref=DocumentElementProcessor - Error while invoking bean, reason: document processor step error Root Cause: com.workday.services.server.capeclear.bean.dep.ProcessingExc error transforming SOAP request element, reason:error transforming request element, reason:javax.xml.transform.TransformerException: org.w3c.dom.DOMException: HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted.	**ception** Displays when you implement an invalid XSLT processor to create a SoapElementProcessor transformer instance.
java.io.FileNotFoundException	com.capeclear.mediation.MediationException: application=<application-name> - Http Out transport id=retrieveSchemaFromOMS encountered error sending to endpoint <endpoint> Root Cause: java.io.FileNotFoundException: <keystore-file-location> (No such file or directory).	Displays when a keystore file missing from the Workday ESB server.
org.jaxen.UnresolvableException	com.capeclear.mediation.MediationException: Error eval step id=ValidatePaygroup	Displays when the namespace prefix definition is missing from your assembly.

©2026 Workday, Inc. All rights reserved

Error Message	Example	Description
	expression=props['myprop.empl'] = vars['input.picof'].xpath('/ picof:Employee/ picof:Status/ picof:Pay_Group') failed to evaluate. Reason: unable to resolve property: vars['input.picof'].xpath('/ picof:Employee/ picof:Status/ picof:Pay_Group') Root Cause: org.jaxen.UnresolvableException: Cannot resolve namespace prefix 'picof'.	
BadRequestException	com.capeclear.mediation.MediationException: Workday Out transport id=CallGetPos encountered problem sending to endpoint 'https://impl-cc.workday.com/ccx/service/mycompany/Staffing/v17' with operation 'Get_Positions_Request'. Details: Code: SOAP-ENV:Client.validationError, Reason: Validation error occurred. Invalid ID value. '?' is not a valid ID value for type = 'Position_ID' Root Cause: BadRequestException : Error code : unknown.	Displays when your integration generates an invalid SOAP request message.
java.net.SocketTimeOutExceptior	com.capeclear.mediation.MediationException: Transport error in email out transport id=emFailureRecords with endpoint=mailto: reason:com.capeclear.capeconnect.transport.TransportExceptio Transport messaging error: Unable to send message : Exception reading response Root Cause: java.net.SocketTimeoutException: Read timed out.	Displays when your integration contains invalid SMTP details.

©2026 Workday, Inc. All rights reserved

Related Information
Tasks
Deploy Integrations to Workday
Launch Integrations from Studio

AUnit Assembly Testing

Create an AUnit Test Skeleton for an Assembly Project

Context

AUnit is a Workday Studio test framework for developing and running unit tests for assemblies. AUnit test skeletons enable you to build test cases for your assemblies.

Steps

1. Select **File > New > Workday Assembly Project**.
2. On the **Project name** field, enter a name for your project.
   Project names must not contain spaces.
3. On the **Target Runtime** field, select *Workday Runtime*.
4. On the **Configurations** field, select *Assembly (with Java support)*.
5. Click **Finish**.
6. Select **File > New > Other > Class**.
7. On the **New Java Class** menu, consider:

Option	Description
Package	Enter a package name.
Name	Enter a name for the new java class.
Superclass	Click **Browse**, and select `com.workday.aunit.AssemblyTestCase`.

8. Click **Finish**.
9. On the **Project Explorer** view, right-click the **src** folder in your project.
10. Select **Add JUnit Library**.

Result

Studio displays your AUnit test skeleton on the java editor.

Next Steps

Run unit tests for your assemblies.

Concept: AUnit Tests

AUnit is a Workday Studio test framework that enables you to develop and run unit tests for assemblies. You can develop AUnit tests in Java using test skeletons that you create in your assembly projects. AUnit test skeletons support these features:

- Actions
- Annotations
- Assertions

©2026 Workday, Inc. All rights reserved

You can use these features to build test cases for your assemblies.

Example:

```
package com.workday.training.aunit;

import java.io.InputStream;

import com.workday.aunit.AssemblyTestCase;
import com.workday.aunit.actions.Action;
import com.workday.aunit.actions.StandardAction;
import com.workday.aunit.annotations.AssemblyTest;
import com.workday.aunit.annotations.AssertAfter;
import com.workday.aunit.annotations.AtComponent;
import com.workday.aunit.annotations.UnitTest;


@AssemblyTest(project = "AUnitTrainer", displayLabel = "Citizens of the
 Cloud")
    public class SimpleMediationTest extends AssemblyTestCase {

    @UnitTest(startComponent = "Simple-Mediation")
    public void testSimpleMediation() throws Exception {
        setMessagePart(0, "test/simple-input.xml", "text/xml");
    }

    // Using the @AssertAfter annotation and the assertTrue assertion.    //
    @AssertAfter(id = "Simple-Mediation", step="transform")
    public void verifySimpleMediation() throws Exception {
        assertTrue(compare(
            getTestResourceInputStream("test/simple-expected.xml"), "text/
xml",
            (InputStream) getMediationContext().getMessage().getMessagePart(0,
 InputStream.class),
            "text/xml", Comparator.dom));
    }

    @AtComponent(id="Next-Thing")
    public Action handleAfterMediation() throws Exception {
        return new StandardAction(Action.Type.terminate);
    }

}
```

Annotations

All AUnit tests must contain an `@AssemblyTest` annotation. This annotation identifies your AUnit test and enables you to associate the test with your assembly project. Example:

```
@AssemblyTest(project = "AUnitTrainer")
```

Annotations on test methods define the starting point for a unit test in the assembly processing chain. Example:

```
@UnitTest(startComponent = "Simple-Mediation")
```

Assertions

Assertions define Boolean test conditions. If an assertion fails, AUnit aborts the assembly execution and displays the test result as a failure. If an unexpected exception occurs, AUnit aborts the assembly execution and displays the test result as an error.

Example: The sample below uses an `@AssertAfter` annotation and an `assertTrue` assertion to check whether the root message part contains the expected XML after the mediation executes.

©2026 Workday, Inc. All rights reserved

```
@AssertAfter(id = "Simple-Mediation", step="transform")
public void verifySimpleMediation() throws Exception {
    assertTrue(compare(
        getTestResourceInputStream("test/simple-expected.xml"), "text/xml",
        (InputStream) getMediationContext().getMessage().getMessagePart(0,
 InputStream.class),
        "text/xml", Comparator.dom));
}
```

Actions

Actions define AUnit test behaviors when the assembly execution reaches a component or exit point. Example:

```
@AtComponent(id="Next-Thing")
public Action
    handleAfterMediation() throws Exception {
        return new StandardAction(Action.Type.terminate); }
```

Reference: AUnit Test Annotations

Annotation	Description	Example
`@AssemblyTest`	Identifies a test as being an AUnit test and associates it with an assembly.	`@AssemblyTest(project = "AUnitTrainer", displayLabel = "Citizens of the Cloud")`
`@UnitTest`	Identifies an individual test case.	`@UnitTest(startComponent = "Simple-Mediation") public void testSimpleMediation() throws Exception { setMessagePart(0, "test/simple-input.xml", "text/xml"); }`
`@AssertBefore`	Executes assertions before executing a named assembly component or step.	`@AssertBefore(id="Store-Output") public void verifySummaryDoc() throws Exception { assertEquals("10", evaluateExpression("parts[0].xpath summary/ wd:WorkerSummary)')")); }`
`@AssertAfter`	Executes assertions after executing a named assembly component or step.	`@AssertAfter(id = "Simple-Mediation", step="transform") public void verifySimpleMediation() throws Exception { assertTrue(compare(getTestResource simple-expected.xml"), "text/xml", (InputStream) getMediationContext().getMessage() InputStream.class),`

©2026 Workday, Inc. All rights reserved

Annotation	Description	Example

		"text/xml",
		Comparator.dom));
		}
		@AssertAfter(id="Process",
		step="transform")
		public void
		verifySummary() throws
		Exception {
		assertEquals("Ace
		Developer",
		evaluateExpression("parts[0].xpath
		wd:WorkerSummary/
		wd:Name')"));
		
		assertEquals("2010-07-27-07:00",
		evaluateExpression("parts[0].xpath
		wd:WorkerSummary/
		wd:HireDate')"));
		}

`@AssertAfterCompletion`	Executes assertions after the assembly has completed.	`@AssertAfterCompletion`
		`public void`
		`verifyErrorHandling()`
		`throws Exception {`
		`assertEquals(3,`
		`evaluateExpression("props['error.c`
		`}`

`@AtComponent`	Defines where the test intercepts the assembly to perform verification and defines the test behavior for the named component.	`@AtComponent(id="Get-`
		`Workers")`
		`public Action`
		`handleGetWorkersInvocation()`
		`throws Exception {`
		`setMessagePart(0,`
		`"test/fromworkdayin/`
		`workdayin_workers_response.xml",`
		`"text/xml");`
		`return new`
		`StandardAction(Action.Type.mock);`
		`}`

Reference: AUnit Test Actions

Actions	Description	Example
`continue_to_next`	Continues the AUnit test on the next element in your assembly.	`return new` `StandardAction(Action.Type.continu`
`invoke`	Invokes the endpoint specified by the out-transport of your assembly.	`return new` `StandardAction(Action.Type.invoke)`
`mock`	Simulates a response message returned by the out-transport by specifying an input file containing the response message.	`return new` `StandardAction(Action.Type.mock);`

`mock_exception`	Simulates an exception thrown by the out-transport by	`new`
		`StandardAction(Action.Type.mock_ex`
		`new`

©2026 Workday, Inc. All rights reserved

Actions	Description	Example
	specifying an instance of this class: `com.capeclear.capeconnect.transport.BadRequestException(Strin msg)`	`BadRequestException("Include message here..."));`
`terminate`	Terminates the test.	`return new StandardAction(Action.Type.termina`

Web Service Testing

Test Web Services Using a WSDL File

Context

You can use Workday Studio's Web Service Tester to send sample request messages using any operation described in a WSDL file.

Steps

1. On the Studio toolbar, click **Test Web Service in Tester**.
2. As you complete the task, consider:

Option	Description
File	Displays the location of the selected WSDL file.
Workspace	Select a WSDL file from your Workspace.
External	Select a WSDL file from an external location.
URL	Select a WSDL file from a URL.
Initialize response endpoint and authentication details for Sandbox environment	Specifies whether: • The tester's **Location** field is prepopulated with the web service endpoint. • The WS-Security Username Token is prepopulated with your Sandbox user login details. This option is selected by default.
Finish	Click when you've selected a WSDL file. Alternatively, click without specifying a file to open the Web Service Tester and enter the location and message of your choice.

3. Select a WSDL file from your workspace, an external location, or a URL. Click **Finish**.
4. Studio opens the Web Service Tester. As you complete the task, consider:

Option	Description
Location	The web service endpoint.
PortType	The WSDL operations exposed by the web service and its associated binding.

©2026 Workday, Inc. All rights reserved

Option	Description
Operation	Operations from the WSDL file. When you select one, Studio generates a sample message in the **Request** pane.
SOAP action	The SOAPAction header in the HTTP request.
Request	The request message that Studio sends to the web service.
Response	The response message that Studio receives from the web service.

5. Modify the XML in the **Request** pane as required. To add an attachment, click **Add Attachment**, then click **Edit** to specify the file location and attachment's **Name** and **Mime type**.
6. Click **Send Request** to send the request message to the Web service at the URL specified in the **Location** field. Studio displays the response in the **Response** pane.

Test Web Services Using an XSD File

Context

You can use Workday Studio's Web Service Tester to send sample request messages using any element described in an XSD file.

Steps

1. On the Studio toolbar, click **Test Web Service in Tester**.
2. As you complete the task, consider:

Option	Description
File	Displays the location of the selected XSD file.
Workspace	Select an XSD file from your Workspace.
External	Select an XSD file from an external location.
URL	Select an XSD file from a URL.
Initialize response endpoint and authentication details for Sandbox environment	Specifies whether: • The tester's **Location** field is prepopulated with the web service endpoint. • The WS-Security Username Token is prepopulated with your Sandbox user login details. This option is selected by default.
Finish	Click when you've selected an XSD file. Alternatively, click without specifying a file to open the Web Service Tester and enter the location and message of your choice.

3. Select an XSD file from your workspace, an external location, or a URL. Click **Finish**.

©2026 Workday, Inc. All rights reserved

4. Studio opens the Web Service Tester. As you complete the task, consider:

Option	Description
Location	The web service endpoint. Because the location value isn't available in an XSD, you must enter the location manually.
Elements	Element definitions from the XSD file. When you select one, Studio generates a sample message in the **Request** pane.

5. Modify the XML in the **Request** pane as required. To add an attachment, click **Add Attachment**, then click **Edit** to specify the file location and attachment's **Name** and **Mime type**.
6. Click **Send Request** to send the request message to the Web service at the URL specified in the **Location** field. Studio displays the response in the **Response** pane.

Test Individual Web Services from the Schema Explorer

Prerequisites

Display Workday Web Service WSDLs or XSDs in the **Schema Explorer**.

Context

You can test individual web services by using the Workday Studio Web Service Tester to create sample SOAP or XML request messages from the **Schema Explorer**.

Steps

1. To generate a sample request message for a specific web service, right-click a WSDL or XSD element on the **Schema Explorer** view. Select either **Open request in Web Service Tester** or **Open SOAP wrapped request in Web Service Tester**.
2. Studio opens the **WS Tester** view. The **Target** section displays the **Location** of the Web service endpoint.
3. From the **Content-Type** drop-down list, select the message's media type and subtype:

Option	Description
text/plain; charset=utf-8	Specifies plain text, containing no formatting commands or directives.
text/xml; charset=utf-8	Specifies XML text.
application/octet-stream	Specifies an unrecognized type. Example: select this option if you're attaching a binary file to the message.

4. Modify the XML in the **Request** pane as required. To add an attachment, click **Add Attachment**, then click **Edit** to specify the file location and attachment's **Name** and **Mime type**.
5. Click **Send Request** to send the request message to the Web service at the URL specified in the **Location** field. Studio displays the response in the **Response** pane.

Related Information
Tasks
Display Workday Web Service WSDLs

©2026 Workday, Inc. All rights reserved

Test Get Operations and Request Messages Using the Request Builder Wizard

Prerequisites

Display Workday Web Service WSDLs or XSDs in the **Schema Explorer**.

Context

You can use the **Workday Web Service Request Builder Wizard** to simplify testing many Get operations and request messages. Eligible operations and request messages have an arrow decorator in the upper left corner of their icon.

Steps

1. Right-click the relevant Get node in the **Schema Explorer** view, then select **Workday Web Service Builder Wizard**.
2. Select whether you want to:
    - Use a Reference ID to retrieve specific data.
    - Retrieve all available records from the web service.
3. If you selected to use a Reference ID, enter it.
4. Select the Response Groups you want to include.
5. Select whether you want to:
    - Copy the sample request to the clipboard.
    - Open the sample request in a new **WS Tester** view.
    - Replace the contents of the current **WS Tester** view.
6. Click **Finish**.

Configure Web Service Tester Authentication

Prerequisites

Create a web service test.

Context

You can configure a number of security options in Workday Studio's Web Service Tester. The **Authentication** options enable you specify whether to apply WS-Security or basic authentication to SOAP request messages, or to call the internal `cc-authenticate` service.

Steps

1. In the **WS Tester** view, click **Send Options**.
2. Select the **Authentication** check box. As you complete the task, consider:

Option	Description
WS-Security UsernameToken	Enables you to specify a username and password for use with WS-Security, without having to select a WS-Security configuration file. Studio inserts a fragment of XML containing the username and password that you specify in the SOAP Header section of request messages that it generates.
Authentication Service	The Web Service Tester calls the internal `cc-authenticate` service using the username

©2026 Workday, Inc. All rights reserved

Option	Description
	and password that you specify. If the username and password are valid on the server, Workday returns a session ticket which the Web Service Tester adds to the SOAP header when the request is sent. The server processing includes an interceptor that checks that the supplied session ticket is valid. By default, a connection to the `cc-authenticate` service is made once and reused. If you're stopping and starting servers, select **Re-connect to the cc-authenticate service before each send** to create a new connection for each request.
Basic Authentication (pre-emptive)	Enables you to specify a username and password for use with preemptive HTTP Basic Authentication.

Configure Web Service Tester X.509 Authentication

Prerequisites

Create a web service test. Configure an integration system user with a username that corresponds to a key store alias name, but without the `@tenant-name` suffix. Assign the appropriate user-based group for the web service to the integration system user.

Context

You can configure a number of security options in Workday Studio's Web Service Tester. The **X.509 Authentication** options enable you to select a key store alias that corresponds to an integration system user and the associated private and public key pair.

Steps

1. On the **WS Tester** view, click **Send Options**.
2. Select the **X.509 Authentication** check box.
3. Select a username from the **Username** drop-down list. Each username is associated with a key store alias.
4. (Optional.) Click **Manage Keys** to manage existing X.509 keys or generate a new key pair. Studio requests a key store password once per session. As you complete the task, consider:

Option	Description
Generate X.509 key pair	Enables you to generate an X.509 key pair with an associated digital certificate.
Remove key from the key store	Removes the selected key pair.
View the public key (certificate)	Displays the public key for the selected key pair.
Import key pair	Enables you to either: • Import a private key and X.509 certificate from a Java key store.

©2026 Workday, Inc. All rights reserved

Option	Description
	• Import a private key and X.509 certificate in OpenSSL PEM format.
Rename alias	Enables you to rename key store alias for the selected key pair.

Configure Web Service Tester Advanced WS-Security

Prerequisites

Create a web service test.

Context

You can configure a number of security options in Workday Studio's Web Service Tester. The **Advanced WS-Security Configuration** options enable the Web Service Tester to read a security configuration file that specifies a sequence of security steps for the request and another sequence of security steps for the response. It applies the security steps to the outgoing request message and removes security encoding from the incoming response message. The server removes security encoding as it receives, and adds security encoding as it responds.

Steps

1. On the **WS Tester** view, click **Send Options**.
2. Select the **Advanced WS-Security Configuration** check box.
3. Provide a **Configuration File URL** from either a **Workspace** or **External** location.
4. (Optional.) If the security configuration file has relative locations for values such as the key store location, provide a **Configuration Base URL**. Studio prepends this URL to the relative locations.
5. (Optional.) Select the **Re-load the security configuration information before each send** check box to reload the file for each request. This option is useful when you're making changes to the configuration file.

Concept: Web Service Testing

The Workday Studio Web Service Tester enables you to create sample SOAP or XML request messages based on the web service description (WSDL) or on a schema definition (XSD). The tool also displays any response messages that the web service returns. It supports HTTP and HTTPS.

This table summarizes the 3 main ways to launch the Web Service Tester:

Method	Notes
From the Studio menu bar.	Enables you to select a WSDL or XSD file for testing from your workspace, a URL, or an external file location. You can also use this option to open the Web Service Tester without reference to any specific file.
From the **Schema Explorer** view.	Enables you to generate XML requests for a specific WSDL operation or schema element, in either non-SOAP or SOAP envelope-wrapped format.
Using the **Workday Web Service Request Builder** wizard.	Simplifies testing of specific Get operations and Get request messages.

©2026 Workday, Inc. All rights reserved

Reference: Web Service Tester Toolbar Buttons

General

Button	Description
Send Request	Invokes the Web service. The response message is displayed in the **Response** pane.
Send Options	Opens the **Send Options** dialog box, enabling you to configure security options.
Abort Request	Aborts the current request.
Reload the WSDL	Reloads the WSDL driving the Web Service Tester. Useful if the web service has changed.
Choose the XML font via the preferences	Enables you to set font properties for the XML displayed in the **Request** and **Response** panes.
Request/Response History	Provides access to a list of previously sent request messages, enabling you to select and resend them.
Request Only	Displays the **Request** pane only.
Response Only	Displays the **Response** pane only.
Horizontal Layout	Arranges the layout horizontally.
Vertical Layout	Arranges the layout vertically.
Help	Displays context-sensitive Help for the Web Service Tester.

Request Pane

Button	Description
Save Request	Saves the request message as an XML file.
Load Request	Loads a request message.
Format Request	Applies standard indentation to the request message. Text can exceed the panel width.
Restore original or last sent request	For unsent messages, restores the text to its original unformatted and unwrapped state. Overrides any manual edits. For sent messages, returns the text to its state before the last send action.
Add Attachment	Enables you to add or remove a file attachment. To add an attachment to the request message, click **Edit**, select a file from the workspace or an external file location, and specify its **Name** and **Mime Type**. To remove an attachment from a request message, click **Remove**. To save an attachment included in a response message, click **Open** below the **Response** pane.

©2026 Workday, Inc. All rights reserved

Button	Description
Increase Font	Increases text size in the **Request** pane.
Decrease Font	Decreases text size in the **Request** pane.

Response Pane

Button	Description
Save Response	Saves the response message as an XML file.
Load Response	Loads a response message.
Format Response	Applies standard indentation to the response message. Text can exceed the panel width.
Wrap Response	Applies indentation, but also formats the text within the boundaries of the panel width.
Restore Response	For response messages, restores the text to its original unformatted and unwrapped state. For sample responses, removes the loaded file from the text pane. Undoes any manual edits.
Sample Response	Generates a sample response for a SOAP request message generated from a WSDL.
Increase Font	Increases text size in the **Response** pane.
Decrease Font	Decreases text size in the **Response** pane.

Assembly Debugging

Debug an Assembly

Prerequisites

- Create an assembly project.
- Security: *Integration Debug* domain in the Integration functional area.

Context

Workday Studio enables you to detect and debug errors in your assemblies. You can debug assemblies that you've deployed to your Workday server environment. If an assembly contains components or steps that reference a static file, ensure that there are no spaces in the referenced filepath before debugging.

Steps

1. On the **Project Explorer** view, right-click an assembly project.
2. Select **Deploy to Workday**.
3. On the **Deploy to Workday** window, consider:

Option	Description
Workday Environments	Select a target Workday environment for deploying your integration.
Collection being deployed	Select a collection.

©2026 Workday, Inc. All rights reserved

4. Click **Finish**.
5. On the **Cloud Explorer** view, select and expand the cloud server node of the target Workday environment to which you deployed your assembly project.
6. Select and expand the collection that you deployed to Workday.
7. Right-click an integration node in your collection.
8. Select **Debug Integration**. On the **Debug Integration** window, consider:

Option	Description
Launch In	The target Workday environment for launching your debug session.
Collection	The name of the collection you're debugging.
Project	The name of the project you're debugging.
Workday-In	The name of the **Workday-In** transport in the project you're debugging.
Launch Parameters	The launch parameters of the project that you're debugging. To debug with alternative launch parameters, click **Select Literal Values** and select from the Workday IDs that the CRF returns. If you know a specific Workday ID that you want to use as a launch parameter, click **Edit Literal Value** and enter it in the **Value** field.

9. Click **Debug**.

   Note: If you encounter problems when the debug runtime begins to start up, consider adjusting the **Session Reconnect Retries** and **Max Startup Wait Iterations** values.

   Session Reconnect Retries determines how many times Studio tries to connect a debug session to the debug runtime. **Max Startup Wait Iterations** determines how many times Studio checks whether the debug runtime is ready to receive requests. Try increasing them in increments of 10.

   You can adjust these values for an existing launch configuration in the**Debug Configurations**dialog.

   To adjust them for future launch configurations, select **Window > Preferences > Workday > Assembly Debug > Debug Runtime**.

Result

The results of your debug session display on the **Console** view.

Concept: Assembly Debugger

The Workday Studio Assembly Debugger is a tool for debugging assemblies in the Workday server environment. The Assembly Debugger enables you to:

- Examine the assembly message runtime context.
- Suspend assembly threads.
- Test MVEL expressions.

If your assembly contains components or steps that reference a static file, ensure that there are no spaces in the referenced filepath before debugging.

Note: You can't debug an assembly in a Production tenant. Use a non-Production environment for debugging.

©2026 Workday, Inc. All rights reserved

Setting Breakpoints

Breakpoints enable you to pause the debugging process at specific points in your assemblies. To set breakpoints on assembly elements, right-click an element and select **Toggle Assembly Breakpoint**. In the main Assembly Editor view, a blue circle displays on the upper right-hand corner of the element to indicate a new breakpoint.

Studio Management

Studio Preferences

Concept: Assembly Editor Preferences

You can specify preferences for your assembly diagrams using the **Assembly Editor** preferences page. To navigate to the preferences page, select **Window > Preferences > Workday > Assembly Editor**. Select one of these nodes to specify the related Assembly Editor preference:

- **Color**
- **Fonts**
- **Migration**
- **Modules**

Color

The **Diagram Color Preferences** page enables you to specify color preferences for components in your assembly diagram.

Setting	Description
Background Color	Specifies the background color for assembly components.
Line Color	Specifies the line color for assembly components.
Text Color	Specifies the font color for assembly components.
Step Background Color	Specifies the background color for assembly steps.
Header Background Color	Specifies the background color for assembly component headers.

Fonts

The **Diagram Font Preferences** page enables you to specify font preferences for components in your assembly diagram.

Setting	Description
Default	Specifies the standard font type, style, and size for your assembly diagram component ID values.
Step	Specifies the standard font type, style, and size for your assembly step ID values.
Conditional Tooltip Header	Specifies the standard font type, style, and size for tooltip header text displayed in your assembly diagram.

©2026 Workday, Inc. All rights reserved

Setting	Description
Conditional Tooltip Content	Specifies the standard font type, style, and size for tooltip content displayed in your assembly diagram.

Migration

On the **Migration** page, you can turn migration on or off for legacy assembly projects created using earlier versions of Workday Studio.

If migration is enabled, Workday Studio automatically migrates the assembly configuration for an assembly.xml file to the latest version of Studio when you open the file.

To turn automatic migration off, clear the **Enable legacy assembly migration in the Assembly Editor** check box before opening or saving an assembly.xml file.

Modules

Before you start developing assembly module templates, create a folder to store your assembly modules either in your workspace, or in an external location such as a folder within your Workday Studio installation.

The **Module Library** in the Assembly Editor's assembly tools palette displays only those assembly modules that are selected to load from the modules folder, as specified in the **Window > Preferences > Workday > Assembly Editor > Modules page**. Click **Browse Workspace** or **Browse File System** to locate the modules folder and specify the path to this folder. Studio displays all assembly module XMI (XML Metadata Interchange) files that are located in the specified folder in the **Modules** table. Select the check box beside each assembly module that you want to load in the **Module Library**. Studio performs template validation and loads only valid XMI files.

At design time, the **Assembly Module** wizard prompts you to save the module XMI file to a folder in your workspace. If preferred, you can later copy the assembly module XMI files from your workspace folder to the modules folder that you specify here, if different from the workspace folder.

Concept: Connections Preferences

The **Connections** preferences page enables you to set your integration cloud connection credentials. To navigate to the **Connections** preferences page, select **Window > Preferences > Workday > Connections**. You can enter credentials for both **Sandbox** and **Production** environments. You can also add connections to other environments, as well as add groups to your connections.

Concept: Deployment Preferences

On the **Deployment** preferences page, you can specify whether to include project source code in the CLAR when deploying to a Workday environment. To do so, select the **Include source code in deployed CLAR** check box.

To exclude the source code from the CLAR, clear the check box.

If you select the **Include source code in deployed CLAR** option, you can import the source to a project in your workspace. To do so, right-click the collection in the **Cloud Explorer** and select **Import Source**.

Concept: HTTPS Preferences

On the **HTTPS** preferences pane, you can specify whether to enable truststore certificate management in Studio. If you select the **Enable truststore certificate management in Studio** check box, when Studio attempts to connect to Workday over HTTPS, a message is displayed that prompts you to trust the Server certificate for the Sandbox or Production environment. The certificates that you trust are displayed in the **Truststore Certificates** pane, where you can reload, remove all, or delete selected certificates. To open the **Truststore Certificates** pane, expand the **Workday > HTTPS** link in the left navigation pane.

©2026 Workday, Inc. All rights reserved

Concept: Ivy Preferences

Workday Studio provides an Ivy plugin, which enables you to manage project dependencies with Apache Ivy. To enable the Ivy plugin to resolve resources, you can specify the settings and properties file to use in your project.

You can configure these settings on the **Workday > Ivy** preferences page:

- **Ivy settings URL:** click **Browse...**, then select the `ivysettings.xml` file or specify a URL. The settings file defines variables that the Ivy engine can load to resolve resources.
- **Ivy properties file:** click **Browse...**, then select the `.properties` file that the settings file can reference. The properties file contains properties that Ivy loads as variables, in the form:

  ```
  name=value
  ```

  Select the **Load ivy.properties file from home directory if present** check box to load the `ivy.properties` file automatically from your home directory.

Concept: Linking Preferences

The **Linking** preferences page specifies whether to use path variables with absolute paths for linked resources in your workspace. To navigate to the **Linking** preferences page, select **Window > Preferences > Workday > Linking**. The **Use PATH variables for linked resource absolute paths** setting is enabled automatically. To disable this setting, clear the check box.

Concept: MVEL Validation Preferences

The **MVEL Validation** preferences page enables you to configure validation for MVEL code. The **Enable MVEL Validation** setting is enabled automatically. To disable this setting, clear the check box.

Concept: Packaged Integrations Preferences

The **Packaged Integrations** preferences page is intended for use by internal Studio developers only for packaged integration deployment purposes. Internal Studio developers, please consult your internal Wiki for more details on support for packaged integrations.

Concept: Runtime Preferences

Workday Studio automatically checks for Workday runtime updates. If any updates are available, Studio downloads them in the background and notifies you upon installation. You can configure your download, installation, and notification preferences on the **Window > Preferences > Workday > Runtime** page.

Concept: Server Preferences

On the **Server** preferences page, you can adjust the **Undeploy Workday collections when they are removed from the Server** setting. This setting is disabled automatically, which means that if a collection is already deployed to the selected Workday environment, and you remove the collection from the **Configured** list, Studio doesn't undeploy the collection. To enable this setting, select the check box. When you click **Finish** on the **Deploy to Workday** wizard, Studio undeploys any collections that are in your workspace, but are no longer in the **Configured** list.

Concept: Splash Screen Login Preferences

At startup, Workday Studio displays an optional splash-screen where you can enter connection details to your Workday environment. If preferred, you can enter your details later in **Connection Details** from the **Cloud Explorer** toolbar. To turn off the splash-screen at startup, select the **Don't show the login on startup** check box, then click **Apply** to save your changes. You can find this check box from the **Workspace Launcher** dialog box when you launch Workday Studio. To access the page in Studio on:

- Windows: Select **Window > Preferences > Workday > Splash Screen Login**.
- Mac OS: Select **Workday Studio > Preferences > Workday > Splash Screen Login**.

©2026 Workday, Inc. All rights reserved

Concept: Status Bar Preferences

The **Status Bar** preferences page enables you to specify the maximum number of status bar items to store in the Status Bar history list, which is displayed in the bottom right-hand corner of the Studio workspace. In the **Max Status Bar history items** field, enter the maximum number of items you wish to store.

Concept: Update Preferences

Workday Studio supports updates for integrators based in low-bandwidth areas through the Workday Content Delivery Network (CDN). You can enable access to the Workday CDN on the **Window > Preferences > Workday > Install/Update** page.

Note: CDN usage requires unrestricted outbound access on port 443.

Concept: XPath Explorer Preferences

The **XPath Explorer** preferences page enables you to specify namespace-prefix mappings to use when generating XML fragments for the **Evaluate XPath to XML fragment** option.

To add a mapping, click **New...**. The **Add Namespace Prefix Mapping** field is displayed, which enables you to enter **Prefix** and **Namespace** values for the mappings that you require.

To remove an unwanted mapping, select it and click **Remove**. To rearrange the order of the mappings, select a mapping and click **Up** or **Down**, as required.

Reference: Assembly Debug Runtime Preferences

Setting	Description
Hostname	The hostname for launching debug sessions.
HTTP Port	The port number that connects debug sessions to Workday.
Session Connect Retries	The number of times Studio attempts to connect debug sessions to Workday.
Max Startup Wait Iterations	The number of times Studio checks if Workday is ready to receive requests.
Accept All Certificates	Specifies whether Studio accepts unsigned certificates.
Enable HTTPS Proxy Settings for Debugging	Enables HTTPS proxy server settings for debug sessions.
Copy Proxy Settings from Eclipse	Specifies whether to copy the proxy settings for debug sessions from Eclipse.
Proxy Host	The proxy server hostname.
Proxy Port	The port on which the proxy server listens.
Proxy User Name	The username for the proxy server.
Proxy Password	The password for the proxy server.
Wait for preceding BP steps timeout, e.g., Retrieval (seconds)	The timeout period that Studio allocates to related business process integrations before resuming debug sessions.

©2026 Workday, Inc. All rights reserved

Reference: Assembly Debug View Preferences

Setting	Description
Auto format root part of XML	Check this box if you want Studio to automatically format the displayed XML for the root part of the message.
Bring the Assembly Debug View to the front when a thread suspends	Check this box if you want Studio to bring the **Assembly Debug** view to the foreground when a thread suspends during the debug process.
Bring the Exception tab to the front when a thread suspends on an error handler	Check this box if you want Studio to bring the **Exception** tab in the **Assembly Debug** view to the foreground when a thread suspends during the debug process.
Automatically retrieve root parts with lengths less than (KB)	Specifies the automatic download threshold, which is the maximum file size for message root parts that the **Assembly Debug** view can display automatically. To view message root parts that exceed the automatic download threshold, you must download the message or save it as a document in the workspace or file system. To do so, use the Assembly Debug view toolbar buttons.
Truncate messages downloaded to the Assembly Debug View when lengths exceed (KB)	Truncates messages that exceed the specified file size when displaying in the **Assembly Debug** view after download.

Reference: Cloud Explorer Preferences

Setting	Description
Ask for confirmation when deleting a collection	If this feature is enabled, you're prompted to confirm your choice before you can delete a collection from the Cloud.
Show annotations in the Cloud Explorer	If this feature is enabled, annotations are displayed on nodes in the **Cloud Explorer**.
Show Workday connector content	If this feature is enabled, the **Cloud Explorer** displays content for the `workday-in` transport such as launch parameters, attributes, and maps.

Reference: Consolidated Report Viewer Preferences

Profile Log Column	Description
Calls	The number of calls the component receives during the integration event.
Avg Time	The average processing time of the component.
Max Time	The maximum processing time of the component.
Min Time	The minimum processing time of the component.

©2026 Workday, Inc. All rights reserved

Profile Log Column	Description
Std Deviation Time	The standard deviation from the average processing time of the component.
Total Time	The total processing time of the component.
Avg Req Size	The average request size of the component.
Max Req Size	The maximum request size of the component.
Min Req Size	The minimum request size of the component.
Std Deviation Req Size	The standard deviation from the average request size of the component.
Total Req Size	The total request size of the component.
Avg Resp Size	The average response size of the component.
Max Resp Size	The maximum response size of the component.
Min Resp Size	The minimum response size of the component.
Std Deviation Resp Size	The standard deviation from the average response size of the component.
Total Resp Size	The total response size of the component.
Max Memory	The maximum memory usage of the component.
Min Memory	The minimum memory usage of the component.
Avg Memory	The average memory usage of the component.
Available Memory	The available memory usage of the component.

Reference: Java Project Settings Preferences

Setting	Description
Enforce compatibility with only the current workbench default JRE	By default, any Java code that Studio generates is compatible with any installed JRE. However, by selecting this check box, you can enforce Java web service projects to be compatible with only a specific JRE, that is, the default Workbench JRE.
Respect export settings for classpath entries in referenced projects	Select this check box to include only exported resources from referenced projects when exporting a WSAR or deploying a web service project.

Reference: Notification Preferences

Setting	Description
Enable notifications	Studio automatically enables this setting. Clear this check box to disable the display of notifications on the Studio screen.
Fade out delay	Enter a value in milliseconds to specify the length of time that notifications remain displayed before fading out.

©2026 Workday, Inc. All rights reserved

Reference: Process Monitor Preferences

Setting	Description
Enable auto refresh	Automatic refreshing of the **Process Monitor** view is enabled by default. To disable this feature, clear the check box.
Refresh interval in seconds	Enter a value (in seconds) to specify the length of time between automatic refreshes. Use the scroll buttons on the right to increase or decrease the value.
Number of last active processes to refresh	Enter the number of active processes you wish to refresh. The processes refreshed are the current and previous processes in reverse chronological order.
Show error dialog when auto refresh fails	Enable the display of an error message when the automatic refresh feature fails. Studio automatically enables this feature. To disable this feature, clear the check box.
Show view after launching an integration	Displays the **Process Monitor** in the foreground when you launch an integration. To disable this feature, clear the check box.

Reference: Web Service Tester Preferences

Setting	Description
Request and response XML display	Specifies the font size and style used to display the request and response XML data. The open editor updates immediately when you change the font details.
Do not display messages larger than:	Specifies the maximum size of messages to be displayed in the Web Service Tester's **Response** panel. The standard value is 1024 KB. Any message larger than the value specified is saved as an attachment. You can save the attachment locally.

Studio Updates

Update Studio

Context

Just as there's only 1 version of the Workday application that's used by all customers, there's only ever 1 valid release of Workday Studio. Your Studio installation must be up-to-date to interact with your Sandbox or Production environment.

If your update fails, Workday recommends that you uninstall the current version of Studio before installing the latest version.

©2026 Workday, Inc. All rights reserved

Steps

1. If an update is available, Studio notifies you when you enter your tenant credentials, either at startup or later. Confirm that you would like to update, then follow the steps in the **Update Workday Studio** wizard.

   Note: Don't update the underlying Eclipse platform independently of the Studio update schedule. Doing so can cause your integrations to fail.
2. If you choose not to update immediately, Studio displays a red arrow in the status bar to act as a reminder. You can still develop assemblies locally while an update remains uninstalled. However, you're prevented from connecting to your Sandbox or Production tenants. To update later, select **Workday > Update Workday Studio**.

   Note: Studio includes a lightweight version of the Workday Runtime to enable you to test and debug your integrations. To ensure that you always have access to the same Runtime that's available in your production tenant, Studio checks for a Runtime update on start-up. If one is available, it's downloaded it in the background. To adjust this behavior, select **Window > Preferences > Runtime > Automatic Updates**.

Workday Assembly Runtime Changes

Concept: Assembly Versions

Assembly version-management allows the Workday Enterprise Service Bus (ESB) server and Studio development teams to evolve assemblies within the Workday Runtime programming model over time without affecting existing users. For example, assembly version-management enables Workday to:

- Deprecate features
- Introduce new features
- Modify the semantics of existing features

All assemblies have a `version` attribute. By default, when you create a new assembly project, Workday Studio sets the `version` attribute value for the `assembly` element in the `assembly.xml` file to the most current version of the Workday Runtime. The format is `year.drop-number`, where drop-number corresponds to the week number. Example: `2016.11`.

Studio displays this number next to the **Assembly** folder in the **Project Explorer**.

When changes are introduced to existing runtime features, Workday ensures that they are backwards compatible. This means that all existing assemblies operate as before, according to their version number. However, by default, new assemblies have the behavior that corresponds to the current version number. Typically, when Workday introduces a change, we define a new attribute on an assembly step or component. Older assemblies do not have this attribute, so we introduce a rule based on the version number to default the value for older assemblies. We also introduce a rule to default a different value for newer assemblies. As a developer, if you want to uptake new features in older assemblies, you have several options. You can:

- Keep the older version and accept the default that the Workday Runtime assigns.
- Keep the older version and set an explicit value for the new attribute, depending on your requirements.
- Migrate to a newer version and accept the new default.
- Migrate to a newer version and set an explicit value for the new attribute.

Reference: Version Changes at the Schema Level

This table summarizes schema-level changes that have occurred for each version update.

©2026 Workday, Inc. All rights reserved

Version	Component	Attribute/Property	Change Description
8.0	async-mediation and sync-mediation components	handle-downstream-errors	Default changed from `true` to `false` A value of `false` means local error handlers in this component only deal with errors occurring in this component's steps. A value of `true` means local error handlers in this component can deal with downstream errors.
9.0	write	output-mime-type	Default changed from `text/plain` to `text/xml`, in line with other steps.
11.0	xmldiff step	copy-to-output	Default changed to `true`. A value of `true` means the step copies the output to the mediation message. A value of `false` means the step copies the output to a variable.
12.0	http-in transport	secure	Default changed from `false` to `null`, meaning the endpoint is available over HTTP and HTTPS unless otherwise specified.
	http-out transport	error-as-response	Default changed from `false` to `true`. A value of `true` means that when the runtime encounters an error in the transport, it sets the error in the current message. Previously, the error was available only through a `BadRequestException` on the `MediationContext`.
	local-out, http-out, ftp-out, ftps-out, sftp-out, xmpp-out, custom-out, email-out transports	ignore-dynamic-endpoints	Default changed from `false` to `true`.

©2026 Workday, Inc. All rights reserved

Version	Component	Attribute/Property	Change Description
			A value of `true` means the transport ignores the dynamic WS-Addressing (WSA) value in the incoming message's header. A value of `false` means the transport uses the WSA value to override the value in its **Endpoint** property.
13.0	store	schema	Default changed from `urn:com.workday/bsvc/blob` to `[http://www.w3.org/2005/Atom]`. The **PutIntegrationMessage** subassembly can directly consume the Atom when adding documents to the current integration event.
	async-mediation and sync-mediation components	continue-after-error	Default changed from `recover` to `rewind`. A value of `rewind` means that processing continues from the previous component in the assembly when an error is cleared. The previous default, `recover`, is now deprecated.
	local-out transport	unset-properties	Default changed from `false` to `true`. A value of `true` means the transport's properties reset after it calls a subassembly.
	on-error error handler component	rethrow-error	Default changed from `true` to `false`. A value of `false` means the runtime clears the error when the error handler fires.

©2026 Workday, Inc. All rights reserved

Version	Component	Attribute/Property	Change Description
			A value of `true` means the error handler must clear the error itself.
14.0	workday-out-soap	replace-with-soap-fault	Default changed from `false` to `true`. A value of `true` means the transport replaces the current message with the SOAP fault when one is returned.
	local-in	access	Default changed from `public` to `private`. A value of `private` means only a `local-out` in the same assembly can call the transport. A value of `public` means any assembly can call the transport.
	The aggregator component's message-content-collater strategy.	output-mimetype	Default changed from `application/octet-stream` to `text/plain`. Custom collaters that implement the `AggregationCollater` interface still default to `application/octet-stream`.
18.0	store	immutable	New property that specifies whether the binary data that the blob references can be modified directly. Defaults to `true`. A value of `true` means the blob is immutable and can't be modified. A value of `false`means the blob is mutable and can be modified. Note: In Workday 18, the server runtime migrates all **store** steps on existing assemblies with old version numbers to

©2026 Workday, Inc. All rights reserved

Version	Component	Attribute/Property	Change Description
			use this new default setting, which makes blobs immutable. If your assembly needs to update a blob, make sure that you set immutable to `false`.
2016.45 Note: After version 27, Workday introduced a new naming scheme in the form `year.week`	PagedGet	is.paged.get.version	Default changed from `util.assemblyVersionAsWWSVer` to `null`.
2019.30	PagedGet	As_Of_Effective_Date As_Of_Entry_DateTime	If no values are provided for these properties, they default to the current date and time (PST).
2020.09	ftp-out, ftps-out, sftp-out		Workday now uses JSCAPE version 9.3, which supports a wider set of encryption ciphers.
2020.09	PrismAnalytics	wd.prism.component.api.hostname	Studio now calls v2 of the Prism Analytics REST API. The component name has changed from PrismConnector to PrismAnalytics.
2021.10	http-out	User-Agent	Sets the default header User-Agent to Workday-Studio.
2023.37	http-out	transport.http.auth.handler	Removes the HTTP header transport.http.auth.handler.
2024.38	http-out	Authorization	Removes the HTTP header Authorization.

Related Information
Reference
FAQ: Encryption, Certificates, and Ciphers for Integrations

Reference: Version Changes at the XML Level

This table below summarizes XML-level changes that have occurred for each version update.

©2026 Workday, Inc. All rights reserved

Version	Previous XML element/ attribute	Replaced by	Meaning/Reason for change
11.0 (These changes all concern changes to integration-system definitions in the workday-in transport component.)	`simple-type-reference`	`simple-type`	The item's type is being declared, not referred to.
	`type-reference`	`class-report-field`	The new element name better represents the meaning.
	`service`	`service-reference`	The item is not being defined, it is being referred to. Removed the `defined-by` attribute, which is no longer used.
	`integration-system/@template`	Removed	No longer applies.
	`param/@required`	`param/@option`	Changed the boolean `required` attribute to an `option` attribute with a value of `required`, where true.
	`cloud/name`	`cloud/@name`	Less verbose.
	`integration/name`	`in-connector/@id`	Less verbose.
	`in-connector/id`	`private-key`	`private-key`
12.0	`key-file`	`private-key`	Affects `sftp-out` and `ftps-out` transports. Renamed the attribute to better reflect the meaning.
	`public-key-properties`	`private-key-properties`	Renamed to better reflect the meaning.
	`aggregator/@spawn-aggregated-processing`	Removed	This feature allowed an aggregator to create a new task or thread to handle an aggregated message. This functionality was not used in Workday integrations and was not suitable for public Workday Studio, so it was removed.
	`store/@compress` `store/@encrypt`	Removed	Previously allowed fine-grained control over the store step. However, Workday

©2026 Workday, Inc. All rights reserved

Version	Previous XML element/ attribute	Replaced by	Meaning/Reason for change
			policy on integrations is to always compress and encrypt, so they were removed.

	`ftp-out/@tmp-directory`
	`sftp-out/@tmp-directory`
	`ftps-out/@tmp-directory`
		Removed	Previously these transports managed their own temp files, but were refactored to use common FileBackedManagedData (FBDM) capability, so these options were removed.

14.0	`delivery-service/child::*`	Removed	Sub-elements of the delivery service defined the allowed transports. Such constraints should not be defined at design time. This is a runtime configuration task, so they were removed from the design time development stage.

Reference: Saxon XSLT Processor Changes

The following table summarizes Workday support for various versions of Saxon and XSLT. Note that after version 27, Workday introduced a new naming scheme in the form `year.week`.

Workday Version	Supported Saxon Version	Supported XSLT Version	Notes
<20	9.1	1.0, (2.0)	Limited support for XSLT version 2.0.
20-2017.05	9.4	2.0, (3.0)	Limited support for XSLT version 3.0.
2017.05-	9.7	3.0	Support for XSLT streaming.

Note: Studio displays a warning message if you create a collection containing assemblies with different version numbers. This is because differently versioned assemblies can require conflicting versions of Saxon, leading to class loading errors at runtime. Best practice is to use the latest assembly versions for all of your integrations.

Note the following:

- The XPath methods that are available in MVEL for accessing message parts and variables support XPath 3.0.

©2026 Workday, Inc. All rights reserved

- The `mctx` URL protocol allows XSL developers to write to message parts and variables, for example:

```
<xsl:result-document href="mctx:vars/var1">...</xsl:result-document>
```

- Although there is no direct assembly support for XQuery, you can add custom steps to call XQuery scripts.
- In assemblies versioned 2022.46 or earlier, Saxon is set as the default transformer only when the `use.saxon.transformer` property is set to `true` in the mediation context. In assemblies versioned 2022.47 or later, Saxon is set as the default if that property is set to `true` or if it's not present.

Note: XSLT 2.0 and XSLT 3.0 are strongly typed. For example, they do not allow `xs:string` to be evaluated for equality in XPath expressions. The following expression, in which `stringParam` is an XSL parameter, results in a Saxon runtime error:

```
$stringParam = true()
```

The XSL compiler will not encounter the issue at deploy time, because it cannot predict what type the parameter will have in advance.

Your existing XSL documents and Workday Studio integrations will not be impacted by this strong typing issue, unless you choose to enable XSLT 3.0 by changing the version attribute value in your XSL documents to `3.0`. If you encounter these runtime errors, which result from the XSLT that you deploy with your integration, you must take action to fix them.

Note: The following error relating to the Saxon upgrade has been observed in legacy XSL for business forms printing:

```
Failed to compile stylesheet.Xerrors detected. Invalid value for @case-
order. Value must be one of (lower-first|upper-first).
```

If you encounter this error, check whether your XSL contains the following attribute setting on a sort element:

```
case-order="#default"
```

This optional `case-order` attribute determines whether the sort lists upper or lower case letters first in the sort. The default is to list upper case first. The Saxon parser accepts only these values for case-order:

- `upper-first`
- `lower-first`

To fix this error, do one of the following:

- Remove the existing case-order attribute from the sort element, because it is an optional setting.
- Set the case-order value explicitly to `upper-first`.
- Set the case-order value explicitly to `lower-first`.

©2026 Workday, Inc. All rights reserved

Sample Assemblies

Example: Run the Hello Cloud Sample Assembly

Context

Workday Studio enables you to deploy and launch integrations in Workday. This example demonstrates how to run a basic sample integration.

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > Hello Cloud**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **HelloCloud** folder and double-click the **Assembly** folder.
4. On the Assembly Editor toolbar, click **Show annotations**.
5. On the **Cloud Explorer** view, click **Connection Details**.
6. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

7. Click **Apply and Close**.
8. On the **Project Explorer** view, right-click the **HelloCloud** folder and select **Deploy to Workday**.
9. On the **Deploy to Workday** window, select the **Sandbox** environment. Click **Finish**.
10. On the **Cloud Explorer** view, expand these nodes: **Sandbox > HelloCloudCollection > HelloCloud**.
11. Right-click the **StartHere** integration node and select **Launch Integration**.
12. Click **Launch**.

Result

The integration launches in your Workday tenant. The **Process Monitor** view in Studio displays the status of the integration.

Next Steps

Sign in to Workday and access the **Process Monitor** task to display the integration event details for the sample assembly.

©2026 Workday, Inc. All rights reserved

Example: Run the Hello Workday Web Services Sample Assembly

Context

Workday Studio enables you to connect to Workday Web Services. This example demonstrates the SOAP web service request message `Get_Workers_Request`. It uses XSLT 3.0 streaming to process the request.

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > Hello Workday Web Services**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **HelloWWS** folder and double-click the **Assembly** folder.
4. On the Assembly Editor toolbar, click **Show annotations**.
5. On the **Cloud Explorer** view, click **Connection Details**.
6. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

7. Click **Apply and Close**.
8. On the **Project Explorer** view, right-click the **HelloWWS** folder and select **Deploy to Workday**.
9. On the **Deploy to Workday** window, select the **Sandbox** environment and click **Finish**.
10. On the **Cloud Explorer** view, expand these nodes: **Sandbox > HelloWWSCollection > HelloWWS**.
11. Right-click the **HelloWWS_EntryPoint** integration node and select **Launch Integration**.
12. Click **Launch**.

Result

The integration launches in your Workday tenant. The **Process Monitor** view in Studio displays the status of the integration.

Next Steps

Sign in to Workday and access the **Process Monitor** task to display the integration event details for the sample assembly.

©2026 Workday, Inc. All rights reserved

Example: Run the Common Usage Patterns Sample Assembly

Context

This example demonstrates some common integration design patterns. It contains multiple separate processing chains, each of which illustrates a different pattern.

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > Common Usage Patterns**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **CommonUsagePatterns** folder and double-click the **Assembly** folder.
4. On the Assembly Editor toolbar, click **Show annotations**.
5. On the **Cloud Explorer** view, click **Connection Details**.
6. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

7. Click **Apply and Close**.
8. On the **Project Explorer** view, right-click the **CommonUsagePatterns** folder and select **Deploy to Workday**.
9. On the **Deploy to Workday** window, select the **Sandbox** environment and click **Finish**.
10. On the **Cloud Explorer** view, expand these nodes: **Sandbox > CommonUsagePatterns > CommonUsagePatterns**.
11. Right-click the **Conditional-Processing** integration node and select **Launch Integration**.
12. Click **Launch**.

©2026 Workday, Inc. All rights reserved

13. Launch each of these integration nodes:
    - **Enriching**
    - **Global-Error-Handling**
    - **Local-Error-Handling**
    - **Looping**
    - **Request-Response-Processing-Chain**
    - **Response-Chaining**
    - **Routing**
    - **Scalable-Message-Processing**
    - **Simple-Processing-Chain**
    - **SubRoutines**
    - **Variables**

    The **Request-Response-Processing-Chain** and **Response-Chaining** integrations have an additional launch parameter: `Employee_ID`. Click **Select literal value** to configure it.

Result

The integration launches in your Workday tenant. The **Process Monitor** view in Studio displays the status of the integration.

Next Steps

Sign in to Workday and access the **Process Monitor** task to display the integration event details for the sample assembly.

Example: Run the Debugging Sample Assembly

Context

Workday Studio enables you to detect and debug errors in your integrations. This example demonstrates how to debug a sample integration.

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Debug*
- *Integration Event*

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > Assembly Debugger**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **AssemblyDebug** folder and double-click the **Assembly** folder.
4. On the Assembly Editor, right-click each of these assembly elements and select **Toggle Assembly Breakpoint**:
    - **create-get-workers-request**
    - **store-summary-doc**
5. On the **Cloud Explorer** view, click **Connection Details**.

©2026 Workday, Inc. All rights reserved

6. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

7. Click **Apply and Close**.
8. On the **Project Explorer** view, right-click the **AssemblyDebug** folder and select **Deploy to Workday**.
9. On the **Deploy to Workday** window, select the **Sandbox** environment and click **Finish**.
10. On the **Cloud Explorer** view, expand these nodes: **Sandbox > AssemblyDebug > AssemblyDebug**.
11. Right-click the **Start-Here** integration node and select **Debug Integration**.
12. Click **Debug**.
13. On the **Confirm Perspective Switch** window, click **Yes**.
14. On the **Assembly Debug** view, select the **Message Root Part** tab and click **Save the message or attachment to a document in the workspace**.
15. On the **Save** window, enter these values:

Option	Description
Parent Folder	AssemblyDebug
File name	rootpart.xml

16. On the **Debug** view, right-click the **Create-Request - create-get-workers-request** breakpoint and select **Resume**.
17. On the **Assembly Debug** view, select the **Message Root Part** tab and click **Save the message or attachment to a document in the workspace**.
18. On the **Save** window, enter these values:

Option	Description
Parent Folder	AssemblyDebug
File name	rootpart1.xml

19. On the **Debug** view, right-click the **Store - store-summary-doc** breakpoint and select **Terminate**.

Result

Studio saves the XML breakpoint files to the **AssemblyDebug** project folder. You can use these files to troubleshoot bugs in your assembly.

Next Steps

Open the XML breakpoint files for your assembly.

Example: Run the Workday-In Features Sample Assembly

Context

The **workday-in** transport component is represented in Workday as an integration system. This sample demonstrates how you can use MVEL expressions in a Workday Studio assembly to reference the integration system's services and launch parameters. It contains 3 separate processing chains.

©2026 Workday, Inc. All rights reserved

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > Workday-In Features**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **WorkdayInFeatures** folder and double-click the **Assembly** folder.
4. On the Assembly Editor toolbar, click **Show annotations**.
5. On the **Cloud Explorer** view, click **Connection Details**.
6. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

7. Click **Apply and Close**.
8. On the **Project Explorer** view, right-click the **WorkdayInFeatures** folder and select **Deploy to Workday**.
9. On the **Deploy to Workday** window, select the **Sandbox** environment and click **Finish**.
10. For each of the individual integrations, consider:

Integration	Note
AttributesAndMaps	After deployment, you can adjust the attribute and map values in Workday. a. Access the **View Integration System** report and select *WorkdayInFeatures-attr-maps*. b. From the related actions menu of the **ExampleAttrMapService** integration service, select **Integration Generic Service > Configure Integration Attributes** or **Integration Generic Service > Configure Integration Maps**. c. Adjust the attribute and map values as required. .
LaunchParameters	On the **Cloud Explorer** view in Studio, expand these nodes: **Sandbox > WorkdayInFeatures > WorkdayInFeatures**. a. On the **referenceParam** field, click **Select literal values** and select an employee.

©2026 Workday, Inc. All rights reserved

Integration	Note
	b. Enter `password` in the **passwordParam** field.
SequenceGenerator	After deployment, you can configure the Sequence Generator service in Workday: a. Access the **View Integration System** report and select *WorkdayInFeatures-sequences*. b. From the related actions menu of the **ExampleSeqGenService** integration service, select **Integration Sequence Generator Service > Configure Integration Sequence Generators**. c. In the **Increment by** field, enter `1`. d. In the **Format/Syntax** field, enter `file[Seq].csv`.

11. On the **Cloud Explorer** view, expand these nodes: **Sandbox > WorkdayInFeatures > WorkdayInFeatures**.
12. Right-click a workday-in transport node and select **Launch Integration**. Example: **AttributesAndMaps**.
13. Click **Launch**.

Result

The integration launches in your Workday tenant. The **Process Monitor** view in Studio displays the status of the integration.

Next Steps

Sign in to Workday and access the **Process Monitor** task to display the integration event details for the sample assembly.

Example: Run the Error Handling Sample Assembly

Context

This example demonstrates several approaches to error handling in Workday Studio. It contains 6 different processing chains, each of which illustrates a different pattern.

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > Error Handling**.
2. On the **Project Names** window, click **Finish**.

©2026 Workday, Inc. All rights reserved

3. On the **Project Explorer** view, expand the **ErrorHandling101** folder and double-click the **Assembly** folder.
4. On the Assembly Editor toolbar, click **Show annotations**.
5. On the **Cloud Explorer** view, click **Connection Details**.
6. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

7. Click **Apply and Close**.
8. On the **Project Explorer** view, right-click the **ErrorHandling** folder and select **Deploy to Workday**.
9. On the **Deploy to Workday** window, select the **Sandbox** environment and click **Finish**.
10. On the **Cloud Explorer** view, expand these nodes: **Sandbox > ErrorHandling > ErrorHandling**.
11. Right-click a workday-in transport node and select **Launch Integration**. Example: **All-Or-Nothing**.
12. Click **Launch**.

Result

The integration launches in your Workday tenant. The **Process Monitor** view in Studio displays the status of the integration.

Next Steps

Sign in to Workday and access the **Process Monitor** task to display the integration event details for the sample assembly.

Example: Run the AUnit Sample Tests

Context

AUnit is a Workday Studio test framework that enables you to perform unit tests on your assemblies. This example demonstrates how to run AUnit tests. It contains 8 separate processing chains.

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > AUnit Tests**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **AUnit 101** folder and double-click the **Assembly** folder.
4. On the Assembly Editor toolbar, click **Show annotations**.
5. Run the **AUnit 101** sample tests.
    a) On the **Project Explorer** view, expand the **src** folder.
    b) Right-click the **com.workday.example.aunit** folder.
    c) Select **Run As > AUnit Test**.

Result

Studio displays the results of the sample tests on the **Console** view.

©2026 Workday, Inc. All rights reserved

Next Steps

Navigate to the **JUnit** view to display a detailed summary of the results.

Example: Run the FOPStep Sample Assembly

Context

Workday Studio enables you to generate PDF files in the output of your integrations. This example demonstrates how to generate basic PDF documents. It contains 3 separate processing chains.

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > FOP Step**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **FOPStep** folder and double-click the **Assembly** folder.
4. On the **Cloud Explorer** view, click **Connection Details**.
5. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

6. Click **Apply and Close**.
7. On the **Project Explorer** view, right-click the **FOPStep** folder and select **Deploy to Workday**.
8. On the **Deploy to Workday** window, select the **Sandbox** environment. Click **Finish**.
9. On the **Cloud Explorer** view, expand these nodes: **Sandbox > FOPStep > FOPStep**.
10. Right-click the **Simple-Start** integration node and select **Launch Integration**.
11. Click **Launch**.
12. On the **Process Monitor** view, right-click the integration and select **Workday: Integration ESB Invocation**.
13. On the **Workday** view, enter these values:

Option	Description
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

14. On the **View Background Process** window, select the **Output Files** tab.
15. On the **File** column of the **Reports and Other Output Files** table, double-click the PDF file.
16. Launch the **Invoice-Start** and **Kanji-Start** integration nodes.

©2026 Workday, Inc. All rights reserved

Result

Workday displays the newly generated PDF files.

Next Steps

Download and save the PDF files.

Example: Run the Extract Organization Data for a Worker Sample Assembly

Context

Workday enables you to categorize workers into groups using organization types. This example demonstrates how to run an integration that extracts the organization type and employee id values of a worker from Workday.

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > Extract Organization Data for a Worker**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **OrgDataForWorkerForOrgType** folder and double-click the **Assembly** folder.
4. On the Assembly Editor toolbar, click **Show annotations**.
5. On the **Cloud Explorer** view, click **Connection Details**.
6. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

7. Click **Apply and Close**.
8. On the **Project Explorer** view, right-click the **OrgDataForWorkerForOrgType** folder and select **Deploy to Workday**.
9. On the **Deploy to Workday** window, select the **Sandbox** environment. Click **Finish**.
10. On the **Cloud Explorer** view, expand these nodes: **Sandbox > OrgDataForWorker > OrgDataForWorkerForOrgType**.
11. Right-click the **StartGetOrgForOrgType** integration node and select **Launch Integration**.
12. On the **Launch Integration** window, navigate to the **Launch Parameters** section.
13. On the **OrgType** field, click **Select literal values**.
14. On the **Select Class Report Field** window, click **Next**.

©2026 Workday, Inc. All rights reserved

15. On the **CRF** window, select an organization type. As you complete the task, consider:

Option	Description
Name	The name of the organization type.
WID	The Workday ID of the organization type.

16. Click **Finish**.
17. On the **EmployeeID** field, click **Select literal values**.
18. On the **Select Class Report Field** window, click **Next**.
19. On the **CRF** window, select an employee. As you complete the task, consider:

Option	Description
Name	The name of the employee.
WID	The Workday ID of the employee.

20. Click **Finish**.
21. Click **Launch**.
22. On the **Process Monitor** view, right-click your integration.
23. Select **Show Details View**.
24. On the **Process Details** view, expand the **OrgType** and **EmployeeID** launch parameter fields.

Result

Studio displays the organization type and employee id values of the worker on the **Process Details** view.

Example: Run the PrismAnalytics Sample Assembly

Context

This example demonstrates how to use the **PrismAnalytics** component to send CSV data to Workday Prism Analytics. Source CSV files are provided in the assembly project. For the purposes of this example, copy these files to an SFTP server for retrieval by the sample assembly.

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Steps

1. Register a Prism API client and copy values that the **PrismAnalytics** component requires.
    a) In Workday, access the **Register API Client for Integrations** task.
    b) Register an API client with **Prism Analytics** as its **Scope (Functional Area)**. Select the **Include Workday Owned Scope** check box.
    c) Copy the **Client ID** and **Client Secret** values.
    d) From the related actions menu, select **API Client > Manage Refresh Tokens for Integrations**.
    e) Select a **Workday Account** to associate with the refresh token. The account must have permission to create Prism tables in Workday.
    f) Generate a refresh token and copy the value.

©2026 Workday, Inc. All rights reserved

2. Obtain Workday REST API Endpoint.
    a) Access the **View API Clients** report.
    b) Copy the **Workday REST API Endpoint** value. Delete the `https://` at the beginning and the `/ccx/api/v1/<tenant_name>` at the end.
3. On the Studio toolbar, select **Help > Welcome > Samples > PrismAnalytics**.
4. On the **Project Names** window, click **Finish**.
5. On the **Project Explorer** view, expand the **PrismAnalytics** folder and double-click the **Assembly** folder.
6. Copy the provided sample CSV files to your SFTP server.
    a) In the **PrismAnalytics** project folder, expand **ws > WSAR-INF > PrismAnalytics Sample CSVs**.
    b) Copy these files to your SFTP server:
        - states.csv
        - states2.csv
        - states3.csv
        - states4.csv
        - states5.csv
7. On the Assembly Editor toolbar, click **Show annotations**.
8. On the **Cloud Explorer** view, click **Connection Details**.
9. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

10. Click **Apply and Close**.
11. On the **Project Explorer** view, right-click the **PrismAnalytics** folder and select **Deploy to Workday**.
12. On the **Deploy to Workday** window, select the **Sandbox** environment and click **Finish**.
13. On the **Cloud Explorer** view, launch 1 of the integrations in this sample.
    a) Expand the **Sandbox** node.
    b) Expand the **PrismAnalyticsCollection** collection node.
    c) Expand the **PrismAnalytics** project node.
    d) Right-click a workday-in transport node and select **Launch Integration**. Example: **SingleCSV**.
14. Click **Launch**.

Result

Workday Prism Analytics creates a table using the CSV data you supplied.

©2026 Workday, Inc. All rights reserved

Third-Party Payroll Interface Sample

Steps: Run the Payroll Interface Sample

Context

In this example, an integration called **PayData Extract** extracts data relating to one-time payments for a pay group. A second Studio integration cycles through the integration event documents and uses XSLT 3.0 streaming to process them.

Steps

1. Create the Payroll Interface Security Group .

   Create a security group with access to the domains required by the payroll interface sample.
2. Access the **Create Integration System User** task.

   Create an integration system user for the **PayData Extract** integration. Example: PayData Extract ISU.

   Select **Security Profile > Assign Integration System Security Groups** from the related actions menu of the integration system user. On the **Integration System Security Group to Assign** prompt, select *Payroll Interface Security*.

   Security: *Integration Security* domain in the Integration functional area.
3. Create the PayData Extract Integration System .

   Create an integration in Workday that extracts one-time payment data for a pay group.
4. Test the PayData Extract Integration System .
5. Deploy the Studio PayData Integration .

   Deploy a Studio integration that cycles through the **PayData Extract** integration event documents and uses XSLT 3.0 streaming to process them.
6. Access the **Create Integration System User** task.

   Create an integration system user for the **PayData** integration. Example: PayData ISU.

   Security: *Integration Security* domain in the Integration functional area.
7. Assign an Integration System User to the PayData Integration .
8. Link the PayData Extract and PayData Integrations .
9. Launch the Linked Integrations .

Create the Payroll Interface Security Group

Prerequisites

Security: *Security Configuration* domain in the System functional area.

Context

Before you can run the payroll interface sample in Studio, you need to create a security group in Workday.

Steps

1. Sign in to your Workday test environment.

©2026 Workday, Inc. All rights reserved

2. Access the **Create Security Group** task. As you complete the task, consider:

Option	Description
Type of Tenanted Security Group	Integration System Security Group (Unconstrained)
Name	Payroll Interface Security

3. Access the **View Security Group** report.
4. Select *Payroll Interface Security* from the **Security Group** prompt.
5. From the related actions menu of the security group, select **Security Group > Maintain Domain Permissions for Security Group**.
6. Select these domains from the **Domain Security Policies permitting Get access** prompt:
   - *Payroll Interface*
   - *Manage: Organization Integration*
   - *Worker Data: Public Worker Reports*
   - *Worker Data: Compensation by Organization*

   Workday alerts you that you must activate your security policy changes.
7. Access the **Activate Pending Security Policy Changes** task and enter a comment.
8. Select the **Confirm** check box.

Result

Workday adds your new security group to the domain security policies of each relevant functional area.

Next Steps

Create an integration system user to run the **PayData Extract** integration.

Create the PayData Extract Integration System

Prerequisites

Security: *Integration Build* and *Integration Configure* domains in the Integration functional area.

Context

Create an integration in Workday that extracts one-time payment data.

Steps

1. Sign in to your Workday test environment.
2. Access the **Create Integration System** task.
3. As you complete the **Create Integration System** task, consider:

Option	Description
System Name	PayData Extract
New Using Template	Payroll Interface

©2026 Workday, Inc. All rights reserved

4. On the **Configure Integration Services** page, select the **Enabled** check box for these integration services:
   - **Payroll Interface/PI Change Detection Launch Parameters**
   - **Payroll Interface/PI Pay Data Section Fields**
   - **Payroll Interface/PI Filename**

   Workday alerts you that you must assign values to some Integration Attributes and configure a Sequence Generator for the integration system.
5. On the related actions menu for the **PayData Extract** integration system, select **Workday Account > Edit**.
6. On the **Workday Account** prompt, enter the name of the integration system user for the **PayData Extract** integration system.
7. On the **Edit Account for Integration System** page, click **OK**.
8. On the related actions menu for the **PayData Extract** integration system, select **Integration System > Configure Integration Attributes**. As you complete the task consider:

Option	Description
Version	Add a row and select the value *25*.
Output Document Tags	Add a row, select **Create Document Tag**, and enter the **Document Tag Name** *PICOF*.
Primary Payroll Integration	Add a row.
Payroll Vendor	Add a row and select any vendor.

9. On the related actions menu for the **PayData Extract** integration system, select **Integration System > Configure Integration Field Attributes**.
10. In the **Field Configuration** section, select the **Include in Output** check box for these fields:
    - **Code**
    - **Date**
    - **Earning or Deduction**
    - **Amount**
11. On the related actions menu for the **PayData Extract** integration system, select **Integration System > Configure Integration Sequence Generators**.
12. On the **Configure Integration Sequence Generators** page, consider:

Option	Description
Increment by	1
Format/Syntax	Payrollinterface-paydata-[Seq].xml

Result

Workday configures the **PayData Extract** integration system.

Next Steps

Test the **PayData Extract** integration system.

Test the PayData Extract Integration System

Prerequisites

Security: These domains in the Integration functional area:

©2026 Workday, Inc. All rights reserved

- *Integration Build*
- *Integration Configure*
- *Integration Event*

Context

Before you can run the payroll interface sample in Studio, you need to test the integration system for it in Workday.

Steps

1. Sign in to your Workday test environment.
2. Access the **View Integration System** task.
3. On the **Integration System** prompt, enter **PayData Extract**.
4. On the related actions menu for the **PayData Extract** integration system, select **Integration > Launch/Schedule**.
5. On the **Schedule an Integration** page, enter these values in the **Value** column:

Option	Description
Pay Group	Select any pay group that has one-time payments.
Pay Period Selection Option	Use Pay Period for Current Date.
Last Successful Run	Select a date after which there have been new one-time payments.

6. Launch the integration and refresh until it completes.

Result

The **PayData Extract** integration system launches in Workday.

Next Steps

Deploy the **PayData** integration to Workday from Studio.

Deploy the Studio PayData Integration

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*

Context

The Studio **PayData** integration cycles through the **PayData Extract** integration event documents and uses XSLT 3.0 streaming to process them. Before you can launch it, you must deploy it to Workday.

Steps

1. On the Studio toolbar, select **Help > Welcome > Samples > Payroll Interface**.
2. On the **Project Names** window, click **Finish**.
3. On the **Project Explorer** view, expand the **PayData** folder and double-click the **Assembly** folder.
4. On the Assembly Editor toolbar, click **Show annotations**.

©2026 Workday, Inc. All rights reserved
Workday Proprietary and Confidential

5. On the **Cloud Explorer** view, click **Connection Details**.
6. Select the **Sandbox** environment and enter these values on the **Connections** window:

Option	Description
Tenant	The name of your tenant.
Username	The username credentials for your tenant.
Password	The password credentials for your tenant.

7. Click **Apply and Close**.
8. On the **Project Explorer** view, right-click the **PayData** folder and select **Deploy to Workday**.
9. On the **Deploy to Workday** window, select the **Sandbox** environment. Click **Finish**.

Result

Studio deploys the **PayData** sample integration to Workday.

Next Steps

Create an integration system user to run the **PayData** integration.

Assign an Integration System User to the PayData Integration

Prerequisites

Security: *Integration Configure* domain in the Integration functional area.

Context

Assign an integration system user to the **PayData** integration system in Workday.

Steps

1. On the **Cloud Explorer** view, expand these nodes: **Sandbox > PayData > PayData**.
2. Right-click the **StartHere** integration node.
3. Select **Workday: Integration System**.
4. On the **Workday** view, consider:

Option	Description
Username	The username credentials for your Workday tenant.
Password	The password credentials for your tenant.

5. On the related actions menu for the **PayData** integration system, select **Workday Account > Edit**.
6. On the **Workday Account** prompt, enter the name of the integration system user for the **PayData** integration system.
7. On the **Edit Account for Integration System** page, click **OK**.

Result

Workday assigns an integration system user to the **PayData** integration system.

Next Steps

Link the **PayData Extract** and **PayData** integrations.

©2026 Workday, Inc. All rights reserved

Link the PayData Extract and PayData Integrations

Prerequisites

Security: *Integration Configure* domain in the Integration functional area.

Context

Link the PayData Extract and PayData integrations in Workday.

Steps

1. Sign in to your Workday test environment.
2. Access the **View Integration System** task.
3. On the **Integration System** prompt, enter *PayData Extract*.
4. On the related actions menu for the **PayData Extract** integration system, select **Business Process > Create, Copy, or Link Definition**.
5. Click **OK** to create a new business process definition.
6. On the **Business Process Steps** tab of the **Edit Business Process Integration Process Event for PayData Extract (TOP LEVEL)** page, click **Add Row**. On the new row, consider:

Option	Description
Order	c
Type	Integration

   Workday warns you that configuration isn't yet complete.
7. Click **Configure Integration System**.
8. On the **Integration** prompt, select **PayData**.
9. On the **Integration Attributes** tab of the **Configure Integration Step** page, consider:

Option	Description
Value Type	*Determine Value at Runtime*
Value	*Deliverable Documents*

Result

Workday links the **PayData Extract** and **PayData** integrations together to deliver the required output.

Next Steps

Launch the linked **PayData Extract** and **PayData** integrations in Workday.

Launch the Linked Integrations

Prerequisites

Security: These domains in the Integration functional area:

- *Integration Build*
- *Integration Configure*
- *Integration Event*

©2026 Workday, Inc. All rights reserved

Context

Launch the **PayData Extract** integration that is now linked to the **PayData** integration.

Steps

1. Sign in to your Workday test environment.
2. Access the **View Integration System** task.
3. On the **Integration System** prompt, enter **PayData Extract**.
4. On the related actions menu for the **PayData Extract** integration system, select **Integration > Launch/Schedule**.
5. On the **Schedule an Integration** page, enter these values in the **Value** column:

Option	Description
Pay Group	Select any pay group that has one-time payments.
Pay Period Selection Option	Use Pay Period for Current Date.
Last Successful Run	Select a date after which there have been new one-time payments.

6. Launch the integration and refresh until it completes.
7. On the **Event Documents** section, note the CSV file that Workday produced. Example: **payrollinterface-paydata-1.csv**.

©2026 Workday, Inc. All rights reserved
Workday Proprietary and Confidential