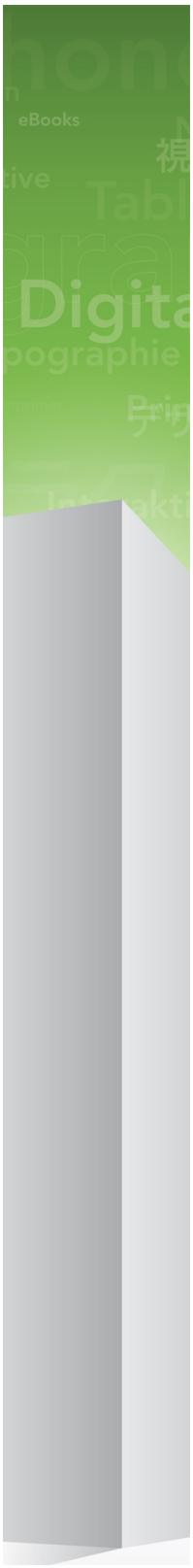


Automation Services 9.5 Workflow Reference

Contents

Introduction

This Guide explains the functionality of the workflows included with Automation Services for Quark Publishing Platform™.

Where we're coming from

This book assumes you are familiar with your computer and know how to:

- Launch an application
- Open, save, and close files
- Use menus, dialog boxes, and palettes
- Use the mouse, keyboard commands, and modifier keys

If you need help performing any of these tasks, consult the documentation resources (user or reference guides) provided with your computer.

Conventions in this book

Formatting conventions in this guide highlight information to help you quickly find what you need.

- Bold type style: The names of all dialog boxes, fields, and other controls are set in bold type. For example: "Click **Storage** in the **Administration** pane."
- References: In descriptions of features, parenthetical references guide you in accessing those features. For example: "The **System Storage** controls (**Administration** pane) let you designate asset storage."
- Arrows: You will often see arrows (>), which map the path to a feature. For example: "Choose **Administration** > **User Profiles** to add a user."
- Icons: Although many tools and buttons are referenced by name, which you can see by displaying ToolTips, in some cases icons are shown for easy identification.
- Cross-platform issues: Some labels, buttons, key combinations, and other aspects of Quark Publishing Platform client applications differ between Mac OS® and Windows® because

of user interface conventions or other factors. In such cases, both the Mac OS and Windows versions are presented, separated by a slash, with the Mac OS version presented first. For example, if the Mac OS version of a button is labeled **Select**, and the Windows version is labeled **Browse**, you are directed to "Click **Select/Browse**." More complex cross-platform differences are mentioned in notes or parenthetical statements.

➡ Notes provide helpful information about particular features and general techniques for using the software.

Understanding workflows

A workflow is a predefined sequence of actions for creating output. You can also think of a workflow as a framework on which you can build an automated task. For more information, see "Understanding workflows" in *A Guide to Automation Services*.

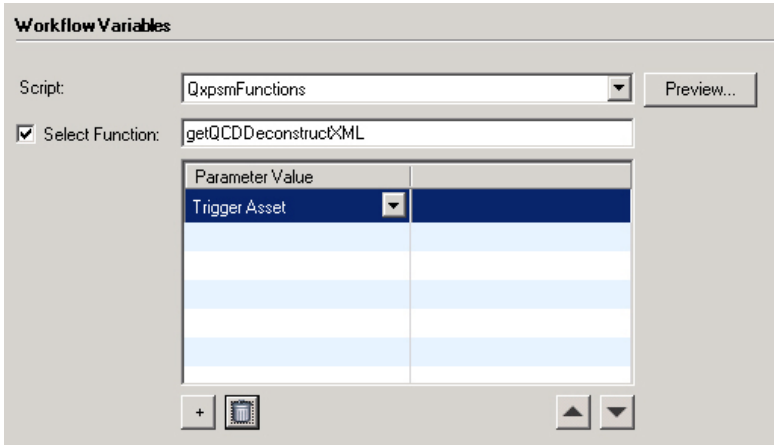
Workflows

The following topics describe the workflows currently available from Quark for Automation Services.

In order to work correctly, Automation Services needs to have sufficient rights to read and write to the folders it works with. For more information, see "Setting access privileges and idle time-out" in *A Guide to Automation Services*.

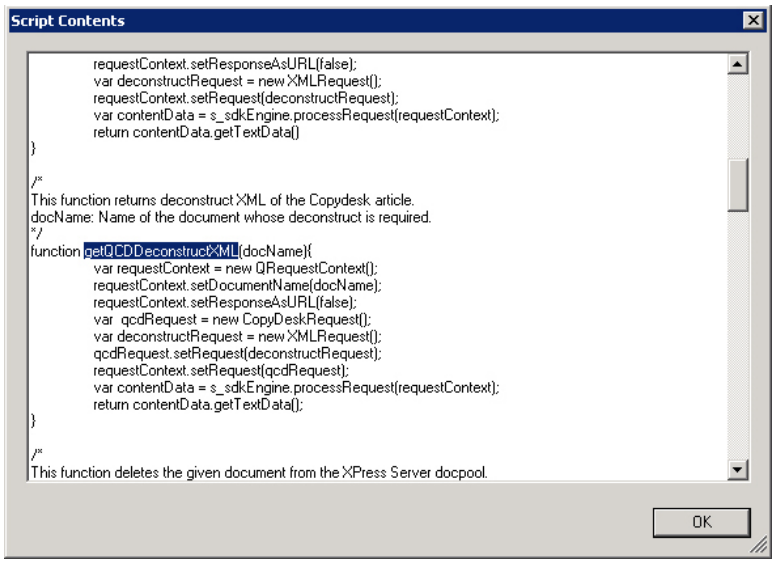
Execute QPP Script

The Execute QPP Script workflow lets you execute either a single function in a Quark Publishing Platform script or the entire script.



Workflow Variables area for Execute QPP Script workflow

- **Script**: Select the name of the target script.
- **Preview**: Displays the selected script in the **Script Contents** dialog box. If you only want to execute a single function, this can help you to locate the function you want and determine what its parameters are and what order they occur in.



Script Contents dialog box

- **Select Function**: If you want to execute a single function, check this box and enter the function name (without parameters) in the corresponding field.
- **Parameter Value** list: The rows in this list correspond to the parameters of the selected function. The first row corresponds to the functions first parameter, the second row to the function's second parameter, and so forth. To add a row, click +. To pass a the trigger asset to a parameter, choose **Trigger Asset** from the drop-down menu for that row. To pass a literal value to a parameter, choose **Other** from the drop-down menu for that row and then double-click in the corresponding field to the right and enter the value.

  If you plan to use the trigger asset, a template, or some other file or asset as a parameter value for a script, you must make sure the script has been set up to handle that file or asset. To do so, load the "ScriptWorkflowUtil.js" script and use its `createFilePathContainer()` method. For example, if you wanted to write a simple script that writes the name of the trigger asset to the console, it might look like this:

```
load("AutomationServicesUtil");
...
function displayFileName(filePathParam)
{
  // Create a filePathContainer from the file path parameter
  var filePathContainer = createFilePathContainer(filePathParam);
  if (filePathContainer.assetId != null) {

    // Get the file name and write it to the console
    var assetName = getAssetName(filePathContainer.assetId);
    print(assetName);
  }
}
```

  The createFilePathContainer function (supplied in the "AutomationServicesUtil.js" script) takes a file reference that has been passed from Automation Server and converts it to a filePathContainer object. You can then access the path to the asset with `filePathContainer.filePath` and get the ID of the asset with `filePathContainer.assetId`.

For more information, see the documentation in the "ScriptWorkflowUtil.js" script.

Export from QuarkXPress Project

The Export from QuarkXPress workflow lets you export layouts from QuarkXPress projects in various formats. You can export Print layouts in PDF and AVE format, individual App Studio layouts in PDF format, and Interactive layouts in SWF format.

- **Image Folder Paths**: Specify the path(s) where any images required by the workflow can be found. This parameter is used only if the QuarkXPress project originates from the file system. If the project is in Quark Publishing Platform, its picture attachments (if any) are used. If the project is in the QuarkXPress Server document pool, Automation Services assumes that the pictures are also in the document pool. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has read access. Check **Use Fallback Image** to substitute the default fallback picture for any pictures that are missing at output. If you uncheck this box, the low-resolution preview image stored in the project file is used.
- **Rendition**: Choose the name of the output style you want to use for PDF rendering. If you do not choose a value, Automation Services uses the default QuarkXPress Server render type.
- **Save Location**: Specify where you want the final file to be put. Note that if you specify a path in the file system, you must use the valid UNC path of a directory which the Automation Services server has write access. Automation Services auto-generates output file names using the formula `[ProjectName]_[LayoutName].[FormatExtension]`.

Publish as AVE

The Publish as AVE workflow lets you transform a QuarkXPress® layout that has been tagged with App Studio XTensions software into an App Studio issue (a .zave file) that can be viewed in an App Studio reader app.

➡ You can control various aspects of the App Studio issues that this workflow creates with an App Studio output style. For more information, see "Job Jackets and resources" in *A Guide to QuarkXPress Server*.

➡ This workflow can be used with trigger assets in Quark Publishing Platform, the file system, or the document pool. The input file can either be a QuarkXPress file or a `.issue` file (for more information, see "*Merging multiple layouts*").

- **Media Folder Paths**: Specify the path(s) where any media files required by the workflow can be found. When the automation profile executes, the software searches these paths for any files referenced in interactive items. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has read access. If Automation Services cannot locate a picture file, it substitutes a default picture

with the same name as the missing picture. The default picture can be found at `[application folder]\AutomationServices_Data\Resources\ProfileManagement\FallbackImage.[suffix]`. Check **Use Fallback Image** to substitute the default fallback picture for any pictures that are missing at output. If you uncheck this box, the low-resolution preview image stored in the project file is used.

- **Export Format**: Specify the format you want to export the issue in (AVE-Mag or AVE-Doc). To use an output style, check **Select output style**, then choose from a list of the AVE output styles available to QuarkXPress Server. If you don't specify an output style, the default AVE output style is used.
- **Output**: Click **Save to Web Feed URL** to post the exported file to a Web server, or **Save to File System** to save it to the file system. If you click **Save to Web Feed URL**, make sure the HTTP PUT method is enabled for the Web server.

➡ If the input to this workflow is a QuarkXPress file that contains multiple layout families (in other words, layouts for multiple issues), the output will include multiple .zave files with different names.

➡ You cannot export an App Studio layout in AVE-Doc format.

➡ This workflow produces a .zip file which, when uncompressed, provides a .zave file (an App Studio issue file) and a .zavem file (an App Studio issue manifest) for each App Studio issue. For more information, see *A Guide to App Studio*.

Merging multiple layouts

If you want to combine multiple layouts into a single .zave file, you can do so by triggering the Publish to App Studio workflow with a `.issue` file. A `.issue` file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<Issue author="text" title="text" freeText="text" keyWords="text" bookId="text"
 creator="text" publisher="text">
  <Projects>
    <QpsSearch>SearchName</QpsSearch>
    <ProjectPath layoutName="Vertical">qpp:Home/Inputs/P3.qxp</ProjectPath>
    <ProjectPath>qpp:Home/Inputs/P4.qxp</ProjectPath>
  </Projects>
</Issue>
```

The `<Issue>` element has a number of optional attributes, which you can use to configure various aspects of an issue:

- `title`: The title of the issue.
- `author`: The issue's author.
- `bookId`: The ID of the issue.
- `publisher`: The name of the issue's publisher.
- `creator`: The creator of the issue
- `freetext`: Text that describes the issue.

- `keyWords`: The keywords pertaining to the issue.

The `<Projects>` attribute defines which layouts should be stitched together. It can include the following element types:

- A `<QpsSearch>` element provides the name of a search. When the workflow executes, Quark Publishing Platform executes the search and returns the list of QuarkXPress projects that match that search, in the order specified by that search's sorting. (Any other types of assets returned by the search are ignored.) The workflow stitches together the first valid layout in each returned project. A `<Projects>` attribute can contain zero or one `<QpsSearch>` elements. Note that **Ask** searches are not supported.
- A `<ProjectPath>` element lets you specify the path to a particular QuarkXPress project in Quark Publishing Platform. You can specify a particular layout by adding the `layoutName="Layout name"` attribute. If you do not specify a particular layout, the first valid layout in the project is used. A `<Projects>` attribute can contain zero or more `<ProjectPath>` elements.

You can include both `<QpsSearch>` and `<ProjectPath>` elements in a .issue file. Automation Services stitches the layouts together in the order in which these elements occur. For example, assume you have a `<Projects>` element that looks like this:

```
<ProjectPath>qpp:Home/MyMag/Issue 1/FrontMatter.qxp</ProjectPath>
<QpsSearch>Guts</QpsSearch>
<ProjectPath layoutName="Alternate">qpp:Home/MyMag/Issue
1/BackMatter.qxp</ProjectPath>
```

In this case, the workflow would stitch together the following layouts in the following order:

1. The first layout in "FrontMatter.qxp"
2. The first layout in each project returned by the search named "Guts"
3. The layout named "Alternate" in "BackMatter.qxp"

To use this feature, create a file named "[Name].issue," as described above. The file can be in Quark Publishing Platform, in the file system, or in the document pool. When you're done, execute an automation profile that uses the Publish to App Studio workflow and specify your "[Name].issue" file as the trigger asset. (You can trigger the automation profile any way you like, as long as the .issue file is the trigger asset.)

➡ You can also execute this workout with a QuarkXPress project as the trigger asset. The .issue method is required only if you need to combine multiple layouts.

Publish as ePUB

The Publish to ePUB workflow lets you transform a reflow article in a QuarkXPress® project into an ePUB file that can be viewed in an ePUB reader application.

In order for this workflow to work, you must supply it with a QuarkXPress project that contains a reflow layout. For more information about reflow layouts, see "Working with Reflow view" in *A Guide to QuarkXPress*.

- **Image Folder Paths**: Specify the path(s) where any image files required by the workflow can be found. When the automation profile executes, the software searches these paths for any image files referenced in interactive items. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has read access. If Automation Services cannot locate a picture file, it substitutes a default picture with the same name as the missing picture. The default picture can be found at `[application folder]\AutomationServices_Data\Resources\ProfileManagement\FallbackImage.[suffix]`. Check **Use Fallback Image** to substitute the default fallback picture for any pictures that are missing at output. If you uncheck this box, the low-resolution preview image stored in the project file is used.
- **Rendition**: This option lets you choose an ePUB output style to be used during export.
- **Save Location**: Click **Save to Web Feed URL** to post the exported file to a Web server, or **Save to File System** to save it to the file system. If you click **Save to Web Feed URL**, make sure the HTTP PUT method is enabled for the Web server.

XML to QuarkCopyDesk

The XML to QuarkCopyDesk® workflow lets you transform content from QuarkXPress projects, QuarkCopyDesk articles, and XML sources into QuarkCopyDesk articles without using an article template.

If the output of the transformation for this workflow includes a QPPEntityModel mapping component, Automation Services uses the default Job Jackets file (if any) for the target main category/subcategory. Otherwise, Automation Services uses the default collection-level Job Jackets file for Quark Publishing Platform.

- **Transformation**: Choose a transformation that has QuarkCopyDesk as an output format, such as **DITA to QuarkCopyDesk Article with Template**.
- **Image Folder Paths**: Specify the path(s) where any images required by the workflow can be found. When the automation profile executes, the software searches these paths for any images referenced in the transformation's XML output, attaches them to the article, and checks them into Quark Publishing Platform (if necessary). Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has read access. If Automation Services cannot locate a picture file, it substitutes a default picture with the same name as the missing picture. The default picture can be found at `[application folder]\AutomationServices_Data\Resources\ProfileManagement\FallbackImage.jpg`.
- **Save Location**: Specify where you want the final file to be put. You can save the file to the file system or check it into Quark Publishing Platform. If you check the file into Quark Publishing Platform, the transformation must supply the corresponding metadata in a

QPPEntityModel output. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has write access, and the path must include the name of the output article. Pictures are stored in a "Pictures" folder at the same level as the output article.

XML to QuarkCopyDesk From Template

The XML to QuarkCopyDesk From Template workflow lets you transform content from QuarkXPress projects, QuarkCopyDesk articles, and XML sources into QuarkCopyDesk articles with an article template.

- **Template**: Choose a QuarkCopyDesk article template.
- **Transformation**: Choose a transformation that accepts a QuarkCopyDesk template and has QuarkCopyDesk as an output format, such as **NewsML to QuarkCopyDesk Article with Template**.
- **Image Folder Paths**: Specify the path(s) where any images required by the workflow can be found. When the automation profile executes, the software searches these paths for any images referenced in the transformation's XML output, attaches them to the article, and checks them into Quark Publishing Platform (if necessary). Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has read access. If Automation Services cannot locate a picture file, it substitutes a default picture with the same name as the missing picture. The default picture can be found at `[application folder]\AutomationServices_Data\Resources\ProfileManagement\FallbackImage.jpg`.
- **Save Location**: Specify where you want the final file to be put. If you check the file into Quark Publishing Platform, the transformation must supply the corresponding metadata in a **QPPEntityModel** output. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has write access, and the path must include the name of the output article. Pictures are stored in a "Pictures" folder at the same level as the output article.

XML to QuarkXPress

The XML to QuarkXPress workflow lets you transform content from QuarkXPress projects, QuarkCopyDesk articles, and XML sources into QuarkXPress layouts with a QuarkXPress template.

- **Template**: Choose a QuarkXPress article template.
- **Transformation**: Choose a transformation that accepts a QuarkXPress template and has QuarkXPress as an output format, such as **DocBook to QuarkXPress Template to PDF**.
- **Image Folder Paths**: Specify the path(s) where any images required by the workflow can be found. When the automation profile executes, the software searches these paths for any images referenced in the transformation's XML output, attaches them to the layout,

and checks them into Quark Publishing Platform (if necessary). Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has read access. If Automation Services cannot locate a picture file, it substitutes a default picture with the same name as the missing picture. The default picture can be found at `[deployment directory]\AutomationServices_Data\Resources\ProfileManagement\FallbackImage.jpg`.

- **Save Location**: Specify where you want the final file to be put. If you check the file into Quark Publishing Platform, the transformation must supply the corresponding metadata in a **QPPEntityModel** output. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has write access, and the path must include the name of the output project. Pictures are stored in a "Pictures" folder at the same level as the output project.

If a transformation in an automation profile makes changes to an existing project and then checks it in as a new revision, you need to be aware of how Automation Services deals with attachments in such projects. If the automation profile makes changes to a box that has an attachment, the attachment is detached in the new revision, unless the Automation Profile includes a **QPPEntityModel** mapping component for the attached article. If the automation profile produces a new project rather than a new revision of an existing project, *all* attached articles are detached unless they have corresponding **QPPEntityModel** mapping components.

XML to WebFeed

The XML to WebFeed workflow lets you transform content from QuarkXPress projects, QuarkCopyDesk articles, and XML sources into formats such as RSS and Atom and post them to a Web feed location via HTTP or the file system.

- **Web Feed**:
  - **Save to Web Feed URL**: Use this option to post the updated Web feed via HTTP. Enter the URL of the input/output file in the **URL** field, then enter a user name and password in the **Web Server Username** and **Web Server Password** field. If you use a proxy server, enter values in the **Proxy Server Username** and **Proxy Server Password** fields. Make sure the HTTP PUT method is enabled for the Web server.
  - **Save to File System**: Use this option to save the updated Web feed to a file system location. Specify where you want the final file to be put. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has write access, and the path must include the name of the output project. Pictures are stored in a "Pictures" folder at the same level as the output project.

- **Transformation**: Choose a transformation that accepts an XML input and has a Web feed format as an output format, such as **XML to RSS**.

XML to XHTML

The XML to XHTML workflow lets you transform content from QuarkXPress projects, QuarkCopyDesk articles, and XML sources into XHTML format. You can then automatically upload the resulting XHTML content into Drupal.

- **Transformation**: Choose a transformation that accepts an XML input and has XHTML as an output format, such as **DITA Topic to XHTML to Drupal** or **QuarkCopyDesk to XHTML to Drupal**.
- **Image Folder Paths**: Specify the path(s) where any images required by the workflow can be found. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has sufficient access rights to read files.
- **Image Format**: Specify the format and (if applicable) quality level for pictures in the workflow.
- **Save Location**: Specify where you want the final files to be put. You can save the output to the file system or submit it as a "Page" to a running instance of the Drupal Web content management system.

  If you save the output to the file system, you must use a valid UNC path to which the Automation Services server has write access, and the path must include the name of the output XHTML file. Pictures are stored in an "Images" folder at the same level as the output XHTML file.

  If you submit the output to Drupal, you must specify the node location of the file that you are uploading. All nodes created by this workflow will be placed at that location, in a node with the same name as the trigger asset. For example, if you specify the location `news/sports` and the name of the trigger asset is "Tennis.xml," the content is checked in at `news/sports/Tennis`. The title of each page is determined by the `<title>` tag in the XHTML output.

➡ If an automation profile creates a new version of a page that already exists in Drupal, Drupal will create a new version of only the page itself; any pictures used in the page will simply be replaced.

➡ Drupal locations should not start with a slash (/).

  If you have trouble submitting content to Drupal, a good place to start troubleshooting is the "AutomationServices.Server.config" file (located at `[application folder]\AutomationServices_Data\Resources\Configuration`). Locate the "scriptRelativeUri" `<entry>` element and verify that the "Quark Automation Services Content Uploader" page can be found in this directory (for more information, see "Installing Drupal for Automation Services" in the Automation Services ReadMe file). Locate the "imageFolderPath" `<entry>` element and make sure this path points to the directory where you want picture files to be uploaded.

  Another setting to check is the `imagefolderpath` entry in the "AutomationServices.Server.config" file (located at `[application`

`folder]\Server\AutomationServices_Data\Resources\Configuration`). This entry should be set to the same value as the `IMAGES_ROOT` value in the "Drupal Content Uploader.php" file.

XML to XML

The XML to XML workflow lets you transform content from QuarkXPress projects, QuarkCopyDesk articles, and XML sources into any valid XML output format.

- **Transformation**: Choose a transformation that accepts an XML input and has XML as an output format.
- **Save Location**: Specify where you want the final file to be put. If you check the file into Quark Publishing Platform Server, the transformation must supply the corresponding metadata in a **QPPEntityModel** output. Note that if you specify a path in the file system, you must use a valid UNC path to which the Automation Services server has write access, and the path must include the name of the output XML file.

  If you save the output to the file system, you must use a valid UNC path to which the Automation Services server has write access, and the path must include the name of the output XML file.

Legal notices

Index