Adobe® LiveCycle™ Workflow 7.0
Writing a PDF file to the file system at the end of a workflow

Ron Stone

Adobe LiveCycle Workflow allows users to save forms to a file system as PDF documents at the end of a workflow, facilitating integration between LiveCycle Workflow and other enterprise systems. For example, many fax servers, print servers, document management or archiving systems, and other applications will utilize a watched collector directory for materials to be processed. By writing to that directory, you can achieve integration between LiveCycle Workflow and other enterprise processes.

Writing a PDF file to the file system at the end of a workflow can also be useful to provide a constituency of users who are not workflow participants with access to the results of a workflow. For example, a workflow may result in customer transaction summaries, which can be written to a series of directories and made available to each customer via HTTPS.

The following steps describe how to write a form to the file system as PDF at the end of a workflow.

Note: In this example we assume that you are storing the form in a process variable called Input_Doc. This variable can be created in a number of ways, including a Form Server or other Document Services Quick Process Action Components (QPAC).

1. In Workflow Designer, open the workflow that you are working with.

2. From the Components palette, open the Core category. Select and drag the Script QPAC to the end of your workflow in the workspace.

3. Double click to open the QPAC.

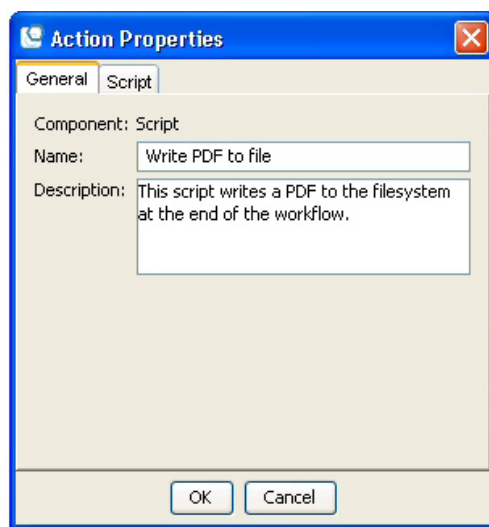4. On the General tab, enter a name and description of the script.

Figure 1: Identifying the Script QPAC

5. On the Script tab, enter Java code similar to the following example:

```java
import com.adobe.workflow.pat.service.*;
import java.io.FileInputStream;
import com.adobe.service.DataBuffer;
import com.adobe.idp.*;
import java.io.*;

//Get the Document variable Input_Doc into a format we can use
in this program

com.adobe.idp.Document myDoc = patExecContext.
getProcessDataDocumentValue("/process_data/@Input_Doc");

//Get the variable into a Byte array called attBytes

DataBuffer _buf = myDoc.getDataBuffer();
byte[] attBytes = _buf.getBytes(0, _buf.getBufLength());

// Now get the filename that we want to write to - supplied by
the user;

String file_str = patExecContext.getProcessDataStringValue("/
process_data/@Output_Filename");

// Write the byte array out to the file specified

File _tempFile = new File(file_str);
FileOutputStream _targetFile = new FileOutputStream(_
tempFile);
_targetFile.write(attBytes);
_targetFile.close();
```

Note: This sample code assumes that a user has specified a file name and location to write to, and that this information is available programmatically. In some instances it may be appropriate to code this information directly into the Script QPAC. In such cases, replace the following line:

```java
String file_str = patExecContext.getProcessDataStringValue("/
process_data/@Output_Filename");
```

with one similar to:

```java
File file_str = new File("C:\\Documents and Settings\\Default
User\\Expense Form\\outputfile.pdf");
```
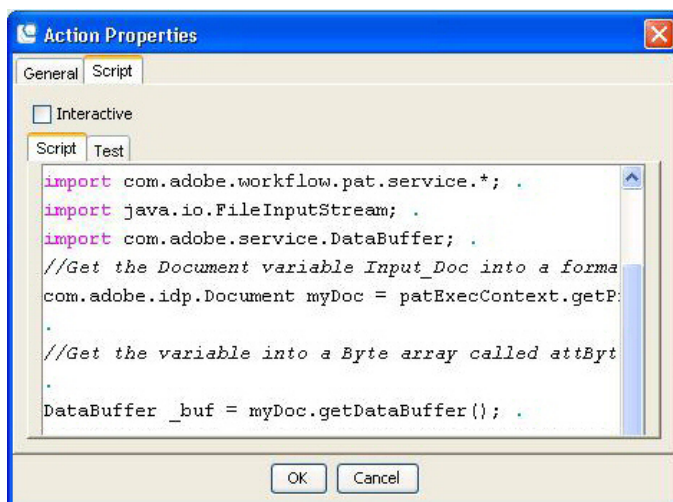

Figure 2: Entering the Java code in the Script QPAC

6. Click OK to save the QPAC.

7. Select File > Save to save the workflow.

8. Test the solution by invoking it and completing all tasks. Check for the saved PDF file.

    Note: If the workflow has not previously been deployed, you will need to do so. If it has been deployed, changes to the workflow are reflected immediately. Caution should be taken when changing deployed workflows, as some changes can cause existing process instances to stall.

ABOUT THE AUTHOR

Ron Stone is a content management consultant and technology writer based in Ottawa, Canada.

Better by Adobe.™