

Calling Reader Extensions from a Java or .NET Application



This is an online bonus article for Chapter 10 of *Paperless: Real-World Solutions with Adobe Technology*. This article demonstrates how you can call Reader Extensions from a Java or .NET application. In addition to the Workbench methods you learned about in the book, you can also call Reader Extensions ES and other LiveCycle services from your Java or .NET application. You can do this in either of the following ways:

- Using the Java API
- Using the web service API

Both techniques are useful when your application and the LiveCycle service are not tightly coupled. You can use these methods to call the LiveCycle services from a different application or from a different physical machine.

Using the Java API

You can use either of the following connection modes when accessing LiveCycle ES through the Java API:

- EJB (Enterprise Java Beans)
- SOAP (Simple Object Access Protocol)

The following is the list of the jar files that are mandatory for both connection modes.

FILE	DESCRIPTION	LOCATION
adobe-livecycle-client.jar	Must always be included in a Java client application's class path.	<install directory>/Adobe/LiveCycle8.2/LiveCycle_ES_SDK/client-libs/common
adobe-usermanager-client.jar	Must always be included in a Java client application's class path.	<install directory>/Adobe/LiveCycle8.2/LiveCycle_ES_SDK/client-libs/common
adobe-utilities.jar	Must always be included in a Java client application's class path.	<install directory>/Adobe/LiveCycle8.2/LiveCycle_ES_SDK/client-libs/<app server>
adobe-reader-extensions-client.jar	Required to invoke the Reader Extensions service.	<install directory>/Adobe/LiveCycle8.2/LiveCycle_ES_SDK/client-libs/common

In order to use SOAP connection, you also need to include these Java packages:

FILE	DESCRIPTION	LOCATION
activation.jar axis.jar commons-codec-1.3.jar commons-collections-3.1.jar commons-discovery.jar commons-logging.jar dom3-xml-apis-2.5.0.jar jaxen-1.1-beta-9.jar jaxrpc.jar log4j.jar mail.jar saaj.jar wsdl4j.jar xalan.jar xbean.jar xercesimpl.jar	If LiveCycle ES is invoked using the SOAP mode, include these JAR files.	<install directory>/LiveCycle8.2/LiveCycle_ES_SDK/client-libs/thirdparty

The sample is created in Eclipse 3.3 as a JAVA Project. Follow these steps to create a similar project:

1. Create a Java Project by selecting File > New > Java Project.
2. In the New Java Project dialog box, name the project *ReaderExtension*. Click Finish.
3. Select the Project in the Navigator tab and expand it to reveal the src folder.
4. Right-click the src folder and select New > Class. The New Java Class dialog box opens (**Figure 1**).

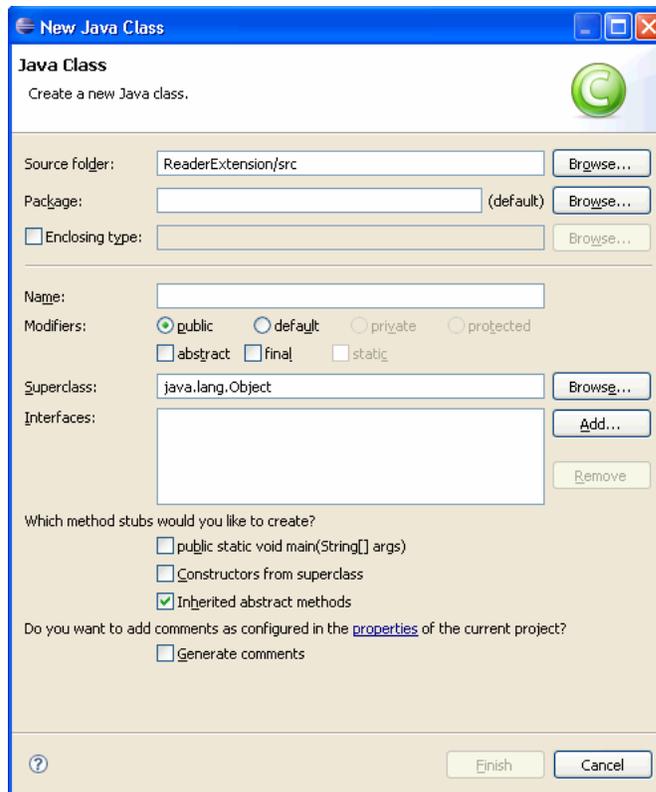


Figure 1 The New Java Class dialog box.

5. Name your Java class *ReaderExtension*. Click Finish.
6. Right click on the project's name and select Properties to bring up the Properties for ReaderExtension dialog box.
7. Select Java Build Path in the left panel and choose the Libraries tab.
8. Click the Add External JARs button and the JAR Selection dialog box will open.
9. Browse to the LiveCycle installation folders and select all of the required JAR files mentioned above. Click Open. Your dialog box should now look like **Figure 2**. If LiveCycle ES is not installed on your machine, ask your administrator to provide the JAR files.

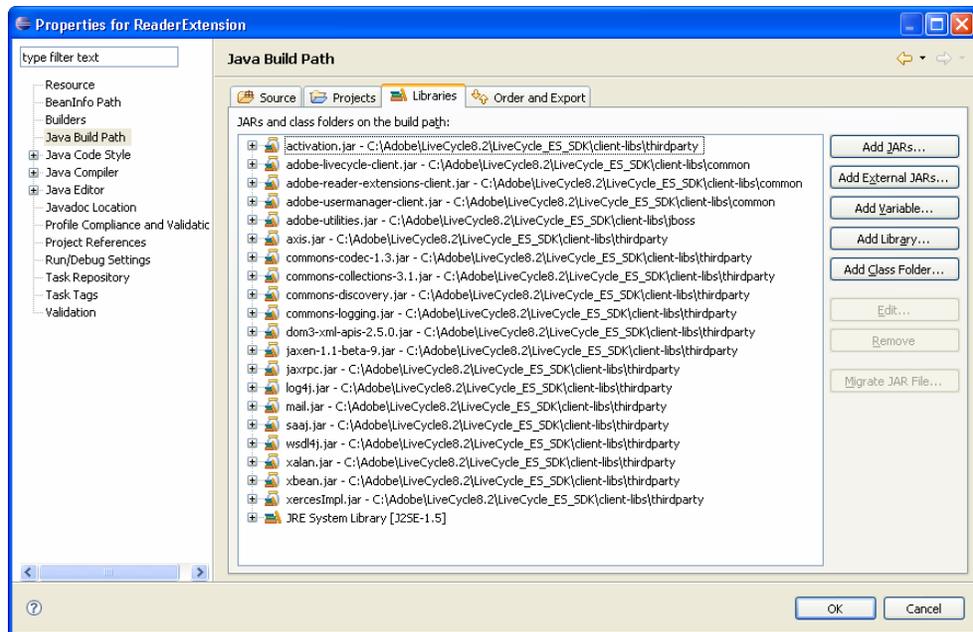


Figure 2 The Properties for ReaderExtension dialog box showing the JAR files.

10. Copy this sample Java code into the Java Class file in Eclipse.

```
import java.io.File;
import java.io.FileInputStream;
import java.util.Properties;

import com.adobe.idp.Document;
import com.adobe.idp.dsc.clientsdk.ServiceClientFactory;
import com.adobe.idp.dsc.clientsdk.ServiceClientFactoryProperties;
import com.adobe.livecycle.readerextensions.client.ReaderExtensionsOptionSpec;
import com.adobe.livecycle.readerextensions.client.ReaderExtensionsServiceClient;
import com.adobe.livecycle.readerextensions.client.UsageRights;

public class ReaderExtension {
    public static void main(String[] args) {
        try{
            //Set connection properties required to invoke LiveCycle ES
            Properties connectionProps = new Properties();
            connectionProps.setProperty(ServiceClientFactoryProperties.DSC_DEFAULT_SOAP_ENDPOINT, "http://localhost:8080");
            connectionProps.setProperty(ServiceClientFactoryProperties.DSC_TRANSPORT_PROTOCOL, ServiceClientFactoryProperties.DSC_SOAP_PROTOCOL);
            connectionProps.setProperty(ServiceClientFactoryProperties.DSC_SERVER_TYPE, ServiceClientFactoryProperties.DSC_<SAMPLE>_SERVER_TYPE);
            connectionProps.setProperty(ServiceClientFactoryProperties.DSC_CREDENTIAL_USERNAME, "username");
            connectionProps.setProperty(ServiceClientFactoryProperties.DSC_CREDENTIAL_PASSWORD, "password");
        }
    }
}
```

```

        //Create a ServiceClientFactory object
        ServiceClientFactory myFactory =
ServiceClientFactory.createInstance(connectionProps);

        //Create a ReaderExtensionsServiceClient object
        ReaderExtensionsServiceClient reClient = new
ReaderExtensionsServiceClient(myFactory);

        //Retrieve the PDF document to which to apply usage rights
        FileInputStream fileInputStream = new FileInputStream("c:\\SampleForm.pdf");
        Document inputPDF = new Document(fileInputStream);

        //Create a UsageRight object and specify specific usage rights
        UsageRights useRight = new UsageRights();
        useRight.setEnabledDynamicFormFields(true);
        useRight.setEnabledComments(true);
        useRight.setEnabledFormFillIn(true);
        useRight.setEnabledDigitalSignatures(true);

        //Create a ReaderExtensionsOptions object
        ReaderExtensionsOptionSpec reOptions = new ReaderExtensionsOptionSpec();

        //Set the usage rights
        reOptions.setUsageRights(useRight);
        reOptions.setMessage("This is a Rights-Enabled PDF Document");

        //Apply usage rights to a PDF document
        Document rightsEnabledPDF = reClient.applyUsageRights(
                                                    inputPDF,
                                                    "SMARTDOC",
                                                    null,
                                                    reOptions);

        //Create a new PDF file that represents the rights-enabled PDF document
        File resultFile = new File("c:\\UsageRightsSampleForm.pdf");
        rightsEnabledPDF.copyToFile(resultFile);
    }catch (Exception e) {
        e.printStackTrace();
    }
}
}
}

```

Make these changes to adapt the code to your environment:

1. Change the *http://localhost:8080* to be your LiveCycle ES server name and port number.
2. Change the *DSC_<SAMPLE>_SERVER_TYPE* to your server type.
3. Change the "username" and "password" to your preferred Username and Password.
4. Change the *c:\\SampleForm.pdf* to the pathname of your PDF file.
5. Change the *SMARTDOC* string to your Reader Extension Credential Alias.

6. Change the `c:\UsageRightsSampleForm.pdf` to the folder and the file name that you want for your Reader extended file.

Using the web service API

Microsoft .NET developers can invoke LiveCycle services through the web service API. This sample was created using C# in Visual Studio 2005. Follow these steps to create a similar project:

1. Create a new Web Site by selecting File > New > Web Site.
2. In the New Web Site dialog box, highlight the ASP.NET Web Site (**Figure 3**). In the Location drop-down, choose File System and click Browse. The Choose Location dialog box will open.

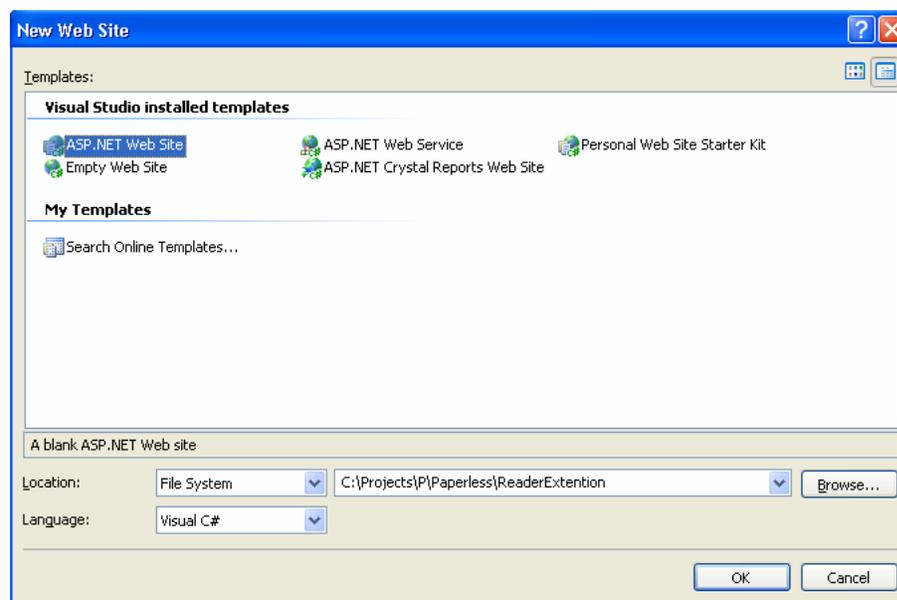


Figure 3 The New Web Site dialog box with all the settings from this example.

3. Select File System and select a directory or type in a new directory for your website. Click Open.
4. In the New Web Site dialog box, select Visual C# as the Language.
5. Click OK.
6. Right-click on your directory name in the Solution Explorer window and select Add Web Reference. The Add Web Reference dialog box will open (**Figure 4**).

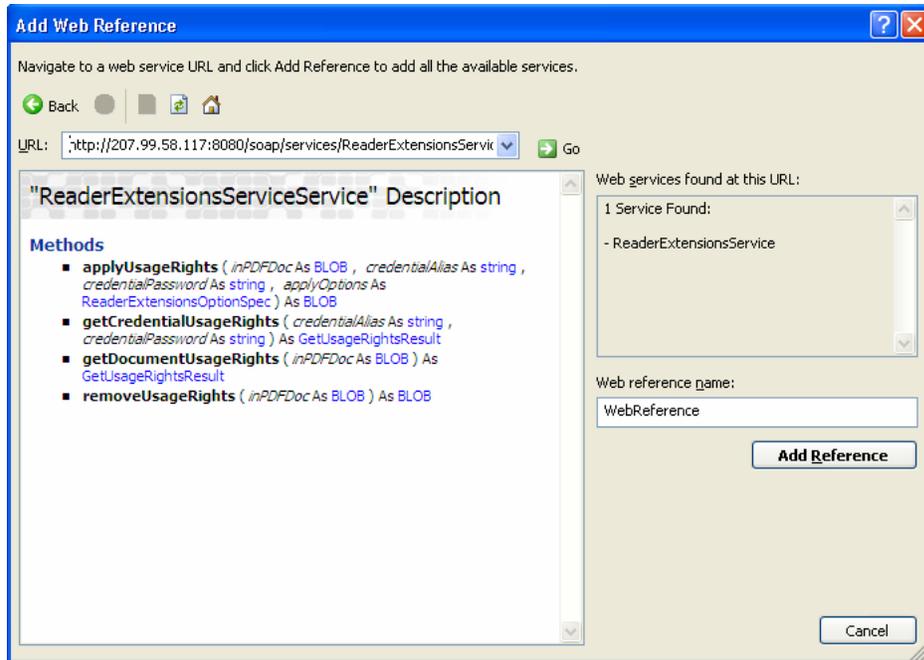


Figure 4 The Add Web Reference dialog box.

7. Enter `http://localhost:8080/soap/services/ReaderExtensionsService?wsdl` in the URL field (change the `http://localhost:8080` to be your LiveCycle ES server name and port number) and click Go. When your screen looks like Figure 4, your Reader Extension service has been added.
8. Click the Add Reference button. You should now see a reference in your Solution Explorer (**Figure 5**).

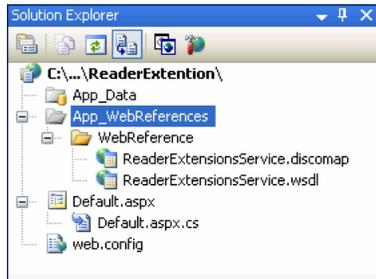


Figure 5 The Solution Explorer with your new web reference.

9. After your web reference is added, open the Default.aspx.cs file and paste this C# code to your .NET project. This example applies usage rights to a PDF document named Form.pdf and saves the rights-enabled PDF document as UsageRightsForm.pdf. There are comments at the beginning of each section that explain the subsequent lines.

```
private void readerExtend()
{
    //Create a ReaderExtensionsServiceService client object
    //The WebReference is generated with the WSDL:
    http://localhost:8080/soap/services/ReaderExtensionsService?WSDL
```

```

WebReference.ReaderExtensionsServiceService readerExClient = new
WebReference.ReaderExtensionsServiceService();

//The following line of code is needed if you want to populate a byte array with the
BLOB data after the call
readerExClient.Url =
"http://localhost:8080/soap/services/ReaderExtensionsService?blob=base64";
readerExClient.Credentials = new System.Net.NetworkCredential("administrator",
"password");

//Create a BLOB to store the input PDF document, Form.pdf
WebReference.BLOB inDoc = new WebReference.BLOB();
string pdfFileName = "c: \\Form.pdf";
FileStream fs1 = new FileStream(pdfFileName, FileMode.Open);

//Get the lengths of the file streams and create byte arrays
int fsLen = (int)fs1.Length;
byte[] fsByteArray = new byte[fsLen];

//Populate the byte arrays with the contents of the file streams
fs1.Read(fsByteArray, 0, fsLen);
fs1.Close();

//Populate the BLOB objects
inDoc.binaryData = fsByteArray;

//Create a UsageRight object and specify usage rights. These are the rights that will
be extended on the form.
WebReference.UsageRights useRight = new WebReference.UsageRights();
useRight.enabledComments = true;
useRight.enabledFormFillIn = true;
useRight.enabledDigitalSignatures = true;
useRight.enabledFormDataImportExport = true;
useRight.enabledSubmitStandalone = true;
useRight.enabledDynamicFormFields = true;
useRight.enabledDynamicFormPages = true;
useRight.enabledEmbeddedFiles = true;

//Create a ReaderExtensionsOptions object
WebReference.ReaderExtensionsOptionSpec reOptions = new
WebReference.ReaderExtensionsOptionSpec();
reOptions.usageRights = useRight;
reOptions.message = "This is a Rights-Enabled PDF Document";

//Apply usage rights to a PDF document, SMARTDOC is the literal value of our
credential alias
WebReference.BLOB outDoc = readerExClient.applyUsageRights(
    inDoc,
    "SMARTDOC",
    null,
    reOptions);

//Populate a byte array with BLOB data
byte[] outByteArray = outDoc.binaryData;

```

```
//Create a new file named UsageRightsLoan.pdf
string fileName = "c: \\UsageRightsForm.pdf";
FileStream fs2 = new FileStream(fileName, FileMode.OpenOrCreate);
BinaryWriter w = new BinaryWriter(fs2);
w.Write(outByteArray);
w.Close();
fs2.Close();
}
```

Make these changes to adapt the code to your environment:

1. Change the `http://localhost:8080/soap/services/ReaderExtensionsService?blob=base64` to be your LiveCycle ES server name and port number.
2. Change the `c:\\ Form.pdf` to the pathname of your PDF file.
3. Change the *SMARTDOC* string to your Reader Extension Credential Alias.
4. Change the `c: \\UsageRightsForm.pdf` to the folder and the file name that you want for your Reader extended file.

Adobe provides Quick Start examples for calling various LiveCycle services in the online SDK Help. If you are looking for examples of other services, review the SDK Help at the following URL:

<http://livedocs.adobe.com/livecycle/es/sdkHelp/programmer/sdkHelp/index.html>

Back to the Book

Now that you understand some alternative ways to call Reader Extensions ES, make sure to review chapters 11 and 12 for more information on LiveCycle. Chapter 11 will introduce you to Digital Signatures ES and Rights Management ES and chapter 12 will show you what's new in ES2.