



Upgrading Adobe® LiveCycle® Workspace ES2 for Flex 4 application deployment

This technical guide is intended for developers who want to leverage the Adobe LiveCycle Enterprise Suite 2.5 (ES2.5) LiveCycle Workspace foundation and API classes/libraries for use on Flex 4 Spark applications that run independently from Workspace but use some of its features, such as SSO authentication.

Introduction

Existing Adobe documentation covers customizing the LiveCycle Workspace ES2 user interface and creating Flex applications enabled for LiveCycle Workspace ES2. This document describes the process of converting and upgrading Workspace classes/libraries from the original Adobe Flex® SDK 3.4 to the new SDK 4.1 and modifications that need to be made in the original source to make those libraries usable in Spark applications.

Why upgrade Workspace libraries to SDK4/Spark? Why not wait for Adobe to upgrade Workspace? Skunkwerks Software needed to deploy its own Flex 4 Spark applications to LiveCycle ES2.5. Some of those applications had been developed using Flex 4 and were based on their own internal framework. Thus, downgrading those apps to SDK 3.4 was not an option.

Our initial requirement was authentication with LiveCycle ES2.5 and requiring the applications to be created as `lc:AuthenticatingApplication`. Deploying Spark applications and using non-Halo themes demanded upgrading `lc:AuthenticatingApplication` to a `spark.components.Application`, instead of an `mx.core.Application`.

The following sections outline the process to accomplish that goal.

Requirements

The requirements are basically the same as those specified in [customizing the LiveCycle Workspace ES2 user interface](#) with the addition of:

- Adobe Flash® Builder™ 4 software
- Flex SDK 4.x

You also need to copy two SWC libraries from the Flex SDK 3.4.1 to your SDK:

- Fds.swc
- Fiber.swc

You may obtain these files from Adobe LiveCycle Data Services ES2 installs, the LiveCycle ES2 Discovery plug-in, or other sources that use/install the libraries.

Installing those libraries into the SDK makes them easily available to all projects. Alternatively, you may leave them outside the SDK and reference them specifically.

Upgrade process

First, copy both projects, foundation and API, so as not to interfere with your regular Workspace development, which stays in Flex 3.

To enable Spark applications to authenticate to and use LiveCycle ES2.5, you need to modify `AuthenticatingApplication`. That class needs to extend `spark.components.Application`, which requires modifications to `AuthenticatingApplication.as` (in the API project) and global replacements throughout both projects, as described in the following table.

Locate all occurrences of:	Replace with:
<code>import mx.core.Application</code>	<code>import mx.core.FlexGlobals; import spark.components.Application;</code>
<code>Application(Application.application)</code>	<code>(FlexGlobals.topLevelApplication as Application)</code>

Table 1—Global replacements

`AuthenticatingApplication.as` needs additional work to make it Spark compatible as described in the table below. All changes are due to the use of the new Spark model and are documented extensively (see "Additional resources").

Locate all occurrences of:	Replace with:
<code>import mx.core.Application</code>	<code>import mx.core.FlexGlobals; import spark.components.Application;</code>
<code>Application(Application.application)</code>	<code>(FlexGlobals.topLevelApplication as Application)</code>
<code>StyleManager.loadStyleDeclarations</code>	<code>Application(FlexGlobals.topLevelApplication).styleManager.loadStyleDeclarations</code>
<code>addChild(UIComponent(</code>	<code>this.addElement(Container(</code>
<code>createComponentsFromDescriptors(recurse)</code>	<code>createDeferredContent()</code>
<code>createComponentsFromDescriptors</code>	<code>createDeferredContent</code>
<code>application.systemManager</code>	<code>this.systemManager</code>

Table 2—`AuthenticatingApplication.as` replacements

You also need to two new imports to `AuthenticatingApplication.as`:

- `import mx.core.Container`
- `import mx.core.ContainerCreationPolicy`

One extra modification to `AuthenticatingApplication.as` involves more than a global replacement. You have to slightly modify the for loop in `setVisibilityForChildren` that hides/removes the Login component. So, replace the first for loop in the method/function with the following code:

Old code:

```
private function setVisibilityForChildren(value:Boolean):void
{
    if (_progress)
    {
        PopUpManager.removePopUp(_progress);
    }

    for each (var child:UIComponent in getChildren())
    {
        if(child is ILoginPage || child is Reconnect)
        {
            removeChild(child);
        }
        else
        {
            child.visible = value;
        }
    }
}
.....
```

New code:

```
private function setVisibilityForChildren(value:Boolean):void
{
    if (_progress)
    {
        PopUpManager.removePopUp(_progress);
    }

    for (var k:uint=0; k < this.numElements; k++)
    {
        var child:UIComponent = this.getElementAt(k) as UIComponent;
        if(child is ILoginPage || child is Reconnect)
        {
            removeChild(child);
        }
        else
        {
            child.visible = value;
        }
    }
}
.....
```

How to use the new libraries

Once you have the two libraries compiled syntax free, you can incorporate them into your own Flex application.

Use **lc:AuthenticatingApplication** instead of **s:Application**:

```
<lc:Application xmlns:lc="http://www.adobe.com/2006/lifecycle"
xmlns:fx="http://ns.adobe.com/mxml/2009"
xmlns:mx="library://ns.adobe.com/flex/mx"
xmlns:s="library://ns.adobe.com/flex/spark"
height="100%"
width="100%"
initComplete="initApp();"/>
```

Adjust your html-templates folder to use the same files from Workspace. The standard Flex html-templates will not work, as they do not include the JavaScript, localization, and SWF files required by LiveCycle ES2.5 authentication.

Conclusion

Additional resources

Many resources detailing Flex 3 to Flex 4 integration are available.

- [Technical guide: Upgrading Flex 3 applications to Flex 4 SDK](#)
- [What's new in Flex 4](#)
- [Differences between Flex 3 and Flex 4](#)
- [Transitioning an application from Flex 3 to Flex 4](#)
- [Moving existing Flex projects from Flex Builder 3 to Flash Builder 4](#)

About the author

Julio Carneiro is a software developer with over 40 years of experience in a multitude of platforms, languages, and environments. He has been developing Flex applications for over six years, starting with Flex 2.5 and has been involved with the LiveCycle platform for about two years. He is the principal architect at Skunkwerks Software Inc., in British Columbia, Canada, an Adobe Solutions Partner and LiveCycle OEM.

For more information and additional product details:
www.adobe.com/devnet/livecycle



Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

Adobe, the Adobe logo, Flash, Flash Builder, Flex, Flex Builder, and LiveCycle are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. All other trademarks are the property of their respective owners.