ADOBE® VERSION CUE® CS3

ADOBE VERSION CUE CS3 CLIENT PROGRAMMER'S GUIDE

Adobe®

Contents

Adobe Version Cue CS3 Client Programmer's Guide

This document describes the Adobe® Version Cue® client from a programmer's perspective. It is intended for system integrators integrating with Version Cue, who are already writing plug-ins for Version Cue Server™.

This document describes the extensibility features enabled by Version Cue SDK at the Version Cue clients such as Adobe Bridge® CS3, and how you can extend Version Cue clients to connect Adobe Creative Suite® 3 to an external system; for instance, a digital asset management (DAM) system.

The extensibility model for Version Cue client is based on server-driven scripting. JavaScript code can be sent from a Version Cue Server plug-in to a Version Cue client, to modify the client's user interface and/or provide additional logic.

We describe the scripting interface at a conceptual level and indicate how to start writing the JavaScript needed to integrate with and extend the Version Cue client. In addition to sending scripts from the client to the Version Cue Server, you can also extend the SOAP protocol that is used to communicate between the clients and Version Cue Server to enable your own custom operations to be triggered at the server-side and we discuss how this protocol extension fits into the extensibility model.

It may also be necessary in some cases to add a client-side component that uses the Version Cue client scripting interface but that you cannot deliver through a server-driven JavaScript. For instance, if you want to add a new tabbed palette to the user interface of Adobe Bridge CS3 that was integrated with your external system through the Version Cue Server, then you would have to install the JavaScript code to create this tabbed palette as a separate client-side component. We provide some basic guidance on what you can achieve with the model of server-driven JavaScript, and when you might have to add some client-side components that cannot be delivered by server-driven scripts.

Terminology

- *Adobe Bridge*™ — Adobe Bridge CS3, a particularly privileged Version Cue client.
- *ExtendScript* — An Adobe implementation ECMA-262 JavaScript specification, used by Creative Suite applications.
- *ExtendScript Toolkit (ESTK)* — An integrated development environment (IDE) developed by Adobe for writing and debugging JavaScript code. To develop JavaScript code for Creative Suite CS3, you must use the latest version of ExtendScript Toolkit 2.0.
- *Version Cue client* — The client component of Version Cue CS3. This is a point product in Creative Suite CS3. This is a client of the *Version Cue Server* and uses the Version Cue protocol to communicate with the Version Cue Server.

- *Version Cue Server* — The server component of Version Cue CS3.
- *Version Cue Server plug-in* — A plug-in developed in Java™ using the *Version Cue SDK*, conforming to the pattern required to be loaded and called by *Version Cue Server*.

NOTE: Unless noted otherwise, references to any Adobe product always mean the CS3 product.

Introduction

Version Cue clients, such as Adobe Bridge CS3 or Adobe InDesign® CS3, are integrated with the Version Cue CS3 client libraries (Windows DLLs or Mac OS frameworks) and can be programmed through the scripting interface to these libraries, which is a JavaScript API. This scripting interface allows you to customize the user interface of the Version Cue client and have your own JavaScript code execute at the client. For details, see the *Scripting Interface Reference*, which is included in the Help system of the IDE for Version Cue SDK.

Point products in the Adobe Creative Suite CS3, like Adobe InDesign CS3 and Adobe Photoshop CS3, use the Adobe Common User Interface, such as the Adobe File > Open dialog. The dialog enables end users to open a file with a uniform user experience on both Mac OS® and Windows®. The Adobe Common User Interface is an alternative to platform-specific file-open dialogs, with divergent behaviour on Mac OS and Windows.

Another key aspect of the Adobe Common User Interface is that it integrates with Version Cue and enables end users to interact with Version Cue Server; for example, check out files for editing, see historical versions of a file, check in a version to Version Cue Server, and search for assets matching a specification on a Version Cue Server.

We recommend using Adobe Bridge CS3 to exercise Version Cue extensibility features illustrated by the Version Cue SDK sample plug-ins, as it has the deepest integration with Version Cue. However, you should also try using the Adobe File > Open dialog and other elements of the Adobe Common User Interface from point products in Creative Suite CS3, such as Adobe InDesign CS3 or Adobe Photoshop CS3, to understand how the extensibility enabled by Version Cue shows up in the other components of the Adobe user interface.

Key Operations

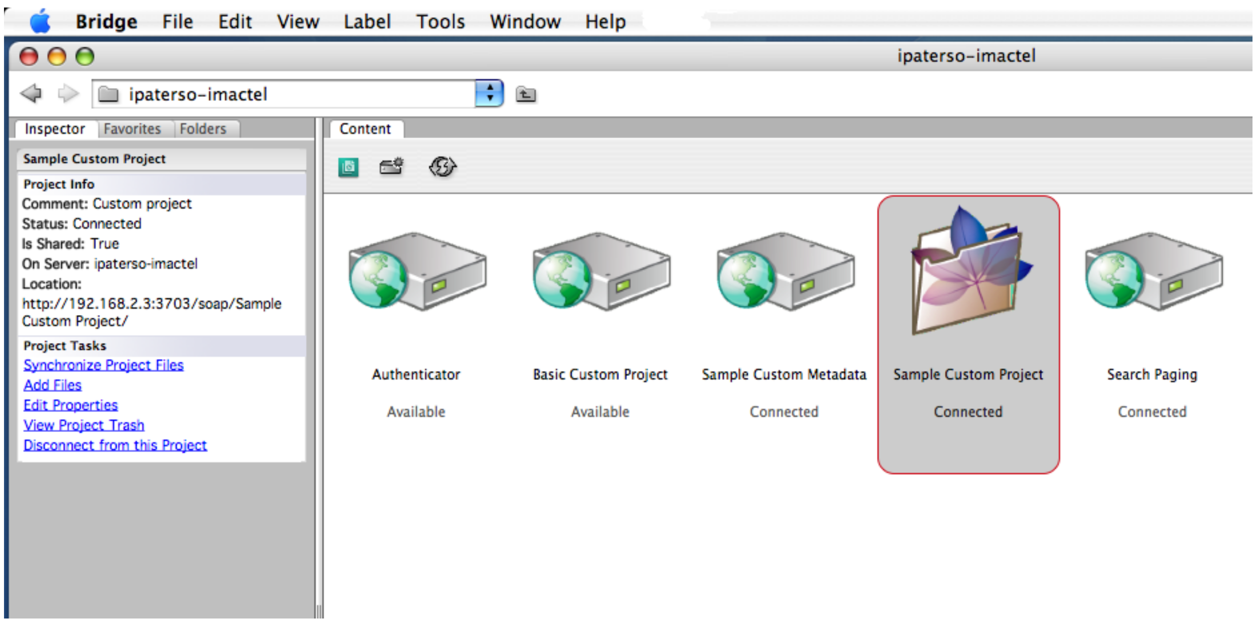
Customizing Asset Icons/ Thumbnails

A key aspect of Version Cue extensibility is customizing how assets appear in the Adobe Common User Interface; for example, providing your own choice of icons or thumbnails for projects, folders, or files. The same extensibility mechanisms that operate in the Adobe Common User Interface also let you customize how assets appear in the user interface of Adobe Bridge CS3, which is tightly integrated with Version Cue.

To see the the Adobe File > Open dialog from the Adobe Common User Interface, using a client such as Adobe InDesign CS3. If the normal operating-system file-open dialog appears instead of the Adobe dialog when you are trying this for the first time or after deleting preferences, click the button labelled "Use Adobe Dialog" in the bottom left-hand corner of the operating-system dialog. To browse Version Cue projects, select the Version Cue item in the left-hand panel of this dialog and navigate to the Version Cue Server that you just launched, the name of which should appear in bold.

If you are using Adobe Bridge CS3, then you can browse Version Cue Servers directly from the Favorites panel, which should show a root node named Version Cue. If you click on this node, then you will see the Version Cue Server instances that are visible from your client; these are displayed within the Bridge Content pane. If you navigate into a specific Version Cue Server by clicking on its icon, then the Version Cue projects on that Server can be seen in the Content pane of Bridge, as in Figure 1.

FIGURE 1 ***Custom Thumbnails for Version Cue Projects in Adobe Bridge CS3***



Sample Custom Project
Project Info
Comment: Custom project
Status: Connected
Is Shared: True
On Server: ipaterso-imactel
Location:
http://192.168.2.3:3703/soap/Sample Custom Project/
Project Tasks
Synchronize Project Files
Add Files
Edit Properties
View Project Trash
Disconnect from this Project

Figure 1 shows a customized project appearance in the user interface of Bridge for "Sample Custom Project"; note how the thumbnail differs in color and form from the other projects shown.

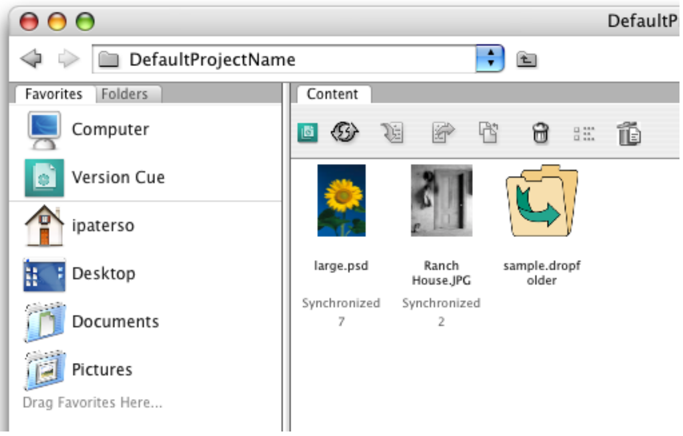
Note that this custom appearance can be viewed in both Bridge and the Adobe Common User Interface, in the context of the Adobe File > Open, File > Place and File > Save dialogs. Ideally, your custom projects— connected to your own digital asset management system, say— will have a different appearance than the standard Version Cue projects, as in this example.

The custom project icon in Figure 1 is provided by a sample plug-in for Version Cue Server named sample.customproject; this sample plug-in is the reference implementation for connecting to an external system and provides a model of how you would implement an adapter between your external system and the Adobe Creative Suite CS3 point products.

Selecting projects in the Content pane of Bridge or the Adobe File > Open dialog and double-clicking to explore them— or using the context-sensitive menu item Open— has the side effect of *mounting* the projects, at which point resources may be sent by the Server to modify the user interface to the project or contribute additional logic.

In addition to customizing the icons for projects, you can customize how folders display in the Adobe user interface. Figure 2 shows an example of a standard Version Cue project displayed in Adobe Bridge CS3, with a *drop folder* provided by the SDK sample named sample.dropfolder. The folder named sample.dropfolder displays a different icon than a standard folder.

FIGURE 2 ***Customized Icons for a Folder***



You also can customize the thumbnails/icons displayed at the level of individual files. See the SDK sample named sample.customproject for an example of providing custom thumbnails for file-based assets; see the Thumbnail implementation class in that sample.

Note that there are extensibility mechanisms to enable you to customize the appearance of assets in the Adobe user interface for both assets in your own external projects and standard Version Cue projects. Note also that asset in the context of Version Cue is a generic term, which can be used to refer to a project, a folder or a file.

For More Information

See the following resources:

- VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > CustomProject
- VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > DropFolder

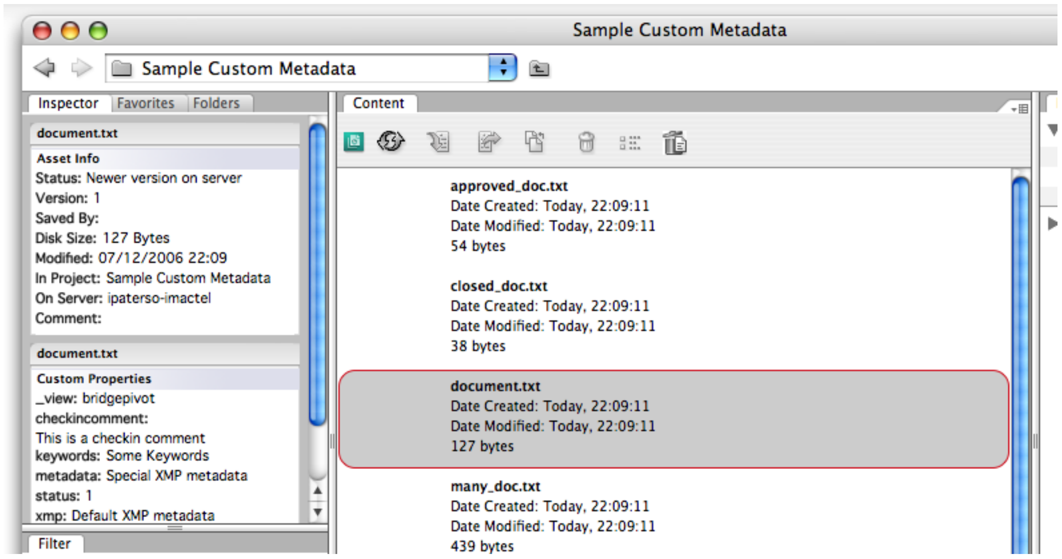
Custom Metadata

Version Cue clients can display custom metadata provided by your Version Cue Server plug-ins, in contexts such as the Inspector Panel of Adobe Bridge CS3 or the File Info area of the Adobe File > Open dialog.

The Inspector Panel in Adobe Bridge CS3 is a read-only *metadata view* component of the Bridge user interface, which lets per-asset metadata be visualized; it is entirely different from the standard Bridge Metadata panel, which is used to display and edit XMP-based metadata.

Figure 3 shows a view of custom metadata for an asset selected in the Content pane of Adobe Bridge CS3. The Inspector Panel displays custom metadata for a file selected in the Content pane.

FIGURE 3 ***Inspector Panel in Bridge showing Custom Metadata***



document.txt
Asset Info
Status: Newer version on server
Version: 1
Saved By:
Disk Size: 127 Bytes
Modified: 07/12/2006 22:09
In Project: Sample Custom Metadata
On Server: ipaterso-imactel
Comment:

document.txt
Custom Properties
_view: bridgepivot
checkincomment:
This is a checkin comment
keywords: Some Keywords
metadata: Special XMP metadata
status: 1
xmp: Default XMP metadata

approved_doc.txt
Date Created: Today, 22:09:11
Date Modified: Today, 22:09:11
54 bytes

closed_doc.txt
Date Created: Today, 22:09:11
Date Modified: Today, 22:09:11
38 bytes

document.txt
Date Created: Today, 22:09:11
Date Modified: Today, 22:09:11
127 bytes

many_doc.txt
Date Created: Today, 22:09:11
Date Modified: Today, 22:09:11
439 bytes

Note that for a Version Cue Server plug-in to display metadata in a context such as the Inspector Panel, one of its responsibilities is to advertise a particular *project capability* to the Version Cue clients; in this case, the capability to overwrite the metadata view at the Version Cue client for assets within a particular project. In "Modifying Project Capabilities" on page 9, we examine what capabilities are in a little more detail.

For More Information

See VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > Custom Metadata

Modifying Project Capabilities

In addition to advertising the additional capabilities a Version Cue Server plug-in supports for a given project, the Version Cue Server plug-in also can restrict or remove capabilities from the default set, or provide only a specific subset. For instance, this extensibility feature can be used to remove menu items from the context-sensitive menu shown when an asset is selected in one of your custom projects, if some operations are not supported.

Figure 4 shows context-sensitive menu items when a file in a standard Version Cue project is selected in Adobe Bridge CS3. Note how there are nearly twenty menu items shown for this context-sensitive menu.

FIGURE 4 ***Context-sensitive Menu for Standard Version Cue Project***

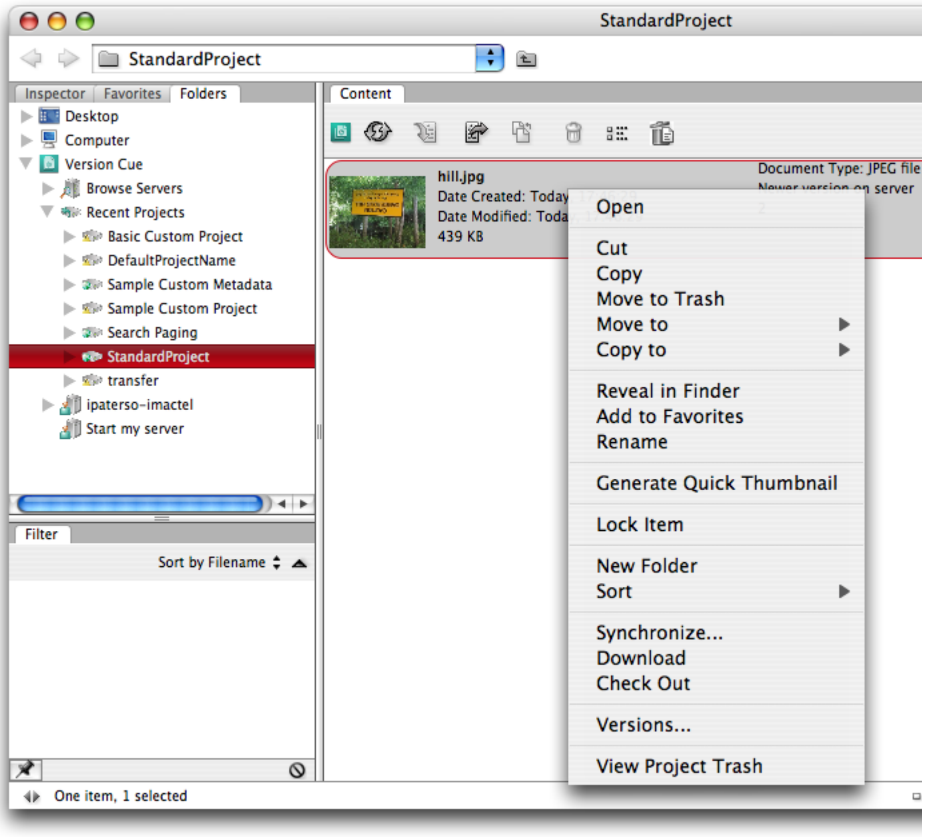
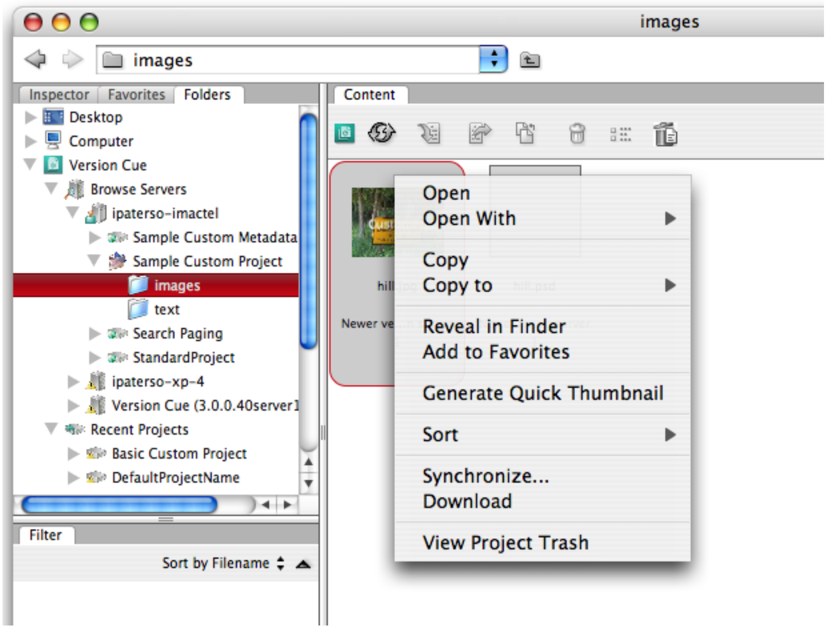


Figure 5 shows the context-sensitive menu when an asset is selected in "Sample Custom Project"; note there are only around ten context-sensitive menu items shown— around half the number compared to a standard project as shown in Figure 4. The custom project has reduced functionality compared to the standard project, and so the Version Cue Server plug-in that provides the custom project needs some way to communicate to the Version Cue client that the set of functionality has been restricted.

FIGURE 5 *Context-sensitive Menu for Custom Project*



This is achieved by restricting the *project capabilities* that sample.customproject provides. See the ProjectCapabilities class in that sample to discover why there are fewer menu items for the context-sensitive menu on the custom project compared to the standard project.

For More Information

See VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > CustomProject

To find out what capabilities exist in the API, look in the API Reference under ICapability; you will see asset, metadata, project and search capabilities. These are of interest if you want to add to or subtract from the default set of capabilities of a standard Version Cue project and you need to know how to declare these.

Downloading and Syndication

Download and *syndication* of assets between an external project and a standard Version Cue project is supported by drag and drop in the user interface of Adobe Bridge CS3 only. In this process, assets are transferred from an external project, typically with read-only access, to a standard project. Syndication is the process of transferring the asset. The destination (standard) project is referred to as the work-in-progress (WIP) project.

After syndication, the user can work on the asset in the standard project exactly as on any other standard Version Cue asset and take advantage of all the Version Cue features that are available, such as versioning assets. The Version Cue Server stores a back-link or *external reference* to the original external asset, by saving some asset-level metadata along with the asset in the standard Version Cue project.

Your Version Cue Server plug-in can use this external reference to propagate any changes to a syndicated asset in the standard Version Cue project back to the external system. For example, if your external DAM system supports versioning, you may want to create a new version of the original asset in the external system when a Creative Suite end user saves a new version of the syndicated asset to the standard project. This extensibility function is demonstrated by the sample.customproject sample in the Version Cue SDK, which listens for new versions of an asset being saved and propagates the newly saved version out to the external system.

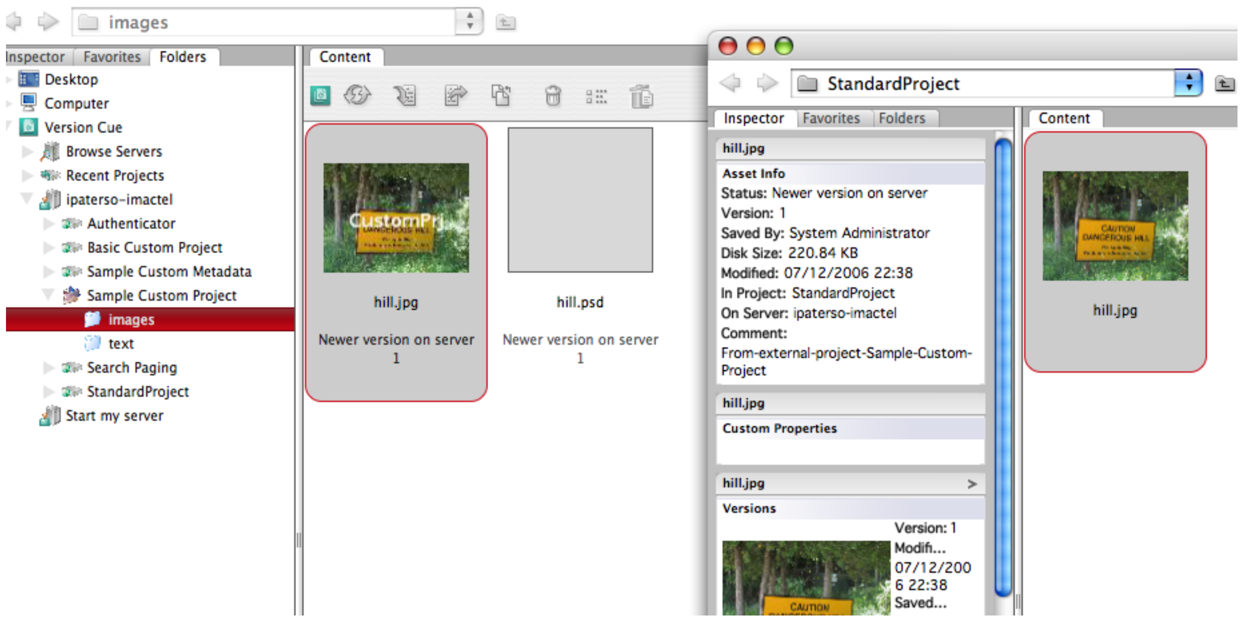
When the user finishes working on a given asset, he could break the connection between the asset in the standard project and the asset in the external system, and archive the asset to the external system. This is referred to as *de-syndication*.

For CS3, client-side support syndication is implemented by drag-and-drop or copy-paste within the Adobe Bridge CS3 user interface:

- You can drag-drop or copy/paste files/ folders from an external project to a standard Version Cue project. When the drag or paste is received by the standard Version Cue project, a check-in dialog appears, and you can supply a check-in comment.
- As a result of the gesture, the asset is shown in the standard (WIP) project. The Version Cue Server stores the back-link (external reference) with the original asset in its database as asset-level metadata, where the key is com.adobe.versioncue/externId.
- After syndication, the user should be able to work with the asset as usual; however, check-ins also will affect the original asset in the DAM project. This behavior is implemented only as part of this sample.customproject reference implementation, and developers outside Adobe probably will use different mechanisms. No further de-syndication process is implemented on the server side. For instance, end users cannot communicate with the Version Cue Server directly via a user interface gesture that they want to de-syndicate— that is, break the connection between asset in standard project and external project.

Figure 6 shows syndicating an asset from an external project named "Sample Custom Project" to a standard Version Cue project, in this case named "StandardProject". The asset on the left is in the custom project, and the asset on the right is in the standard project.

FIGURE 6 *An Asset Syndicated from a Custom Project*



hill.jpg
Asset Info
Status: Newer version on server
Version: 1
Saved By: System Administrator
Disk Size: 220.84 KB
Modified: 07/12/2006 22:38
In Project: StandardProject
On Server: ipaterso-imactel
Comment:
From-external-project-Sample-Custom-Project

hill.jpg
Custom Properties

hill.jpg
Versions
Version: 1
Modifi...
07/12/2006 22:38
Saved...

In the CustomProject sample plug-in, the external asset and the asset in the standard (WIP) project are kept in synchronization, using a storeContent listener. When a new version of the asset in the standard project is saved, the external asset in the custom project from which it was downloaded also is updated. To verify that the external asset— in the contentroot folder of this CustomProject sample— and the standard (WIP) asset are kept in synchrony, follow these steps:

1. Select the syndicated (standard) asset in the standard project in Content pane of Adobe Bridge CS3, and check out the asset from the context-sensitive menu item.
2. Use Reveal in Explorer/ Finder to locate the checked-out local replica file.
3. Edit the local replica file in an appropriate application.
4. Save the edited file locally.
5. Synchronize back to the Version Cue Server by executing the context-sensitive menu item “Synchronize” when the syndicated asset is selected.
6. Open the original external asset in the contentroot folder of the CustomProject sample in an appropriate application and check that the changes were applied to this asset.

Restrictions

The following restrictions apply to asset syndication through drag-and-drop or copy/paste in the user interface of Adobe Bridge CS3:

- Both source and target projects must be mounted before starting the drag-and-drop (or copy/paste) operation.
- Both source and target projects must have authentication switched off; that is, no login should be required for either project to complete the gesture.
- No conflict handling is implemented if an asset with the same name already exists in the target project.
- De-syndication is not supported through the user interface; there is no explicit gesture to de-syndicate an asset.

For More Information

See the following resources:

- VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > CustomProject
- VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > BasicCustom-Project

Customizing the Check-in Dialog: Metadata Capture and Enforcement

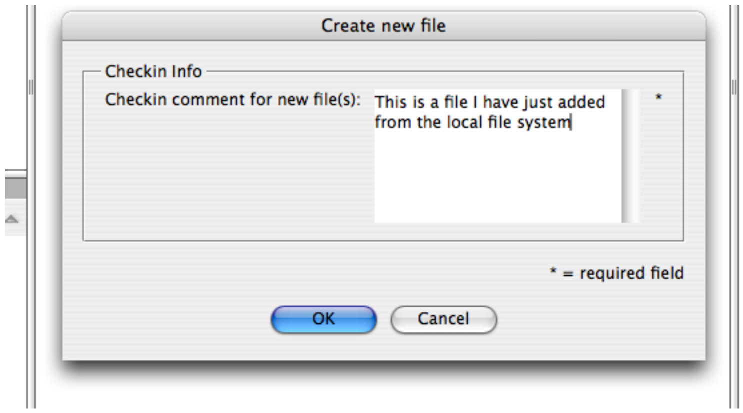
This feature provides the ability to extend and override the standard Adobe check-in dialog of a Version Cue client with a customized check-in dialog; for instance, you may want to contribute your own check-in dialog when files are being added to a Version Cue project for the first time, or every time a new version of a file is checked-in.

A customized check-in dialog can be seen if you are running Version Cue Server with the sample plug-in named sample.customdialog loaded, which is able to contribute resources to override the normal check-in dialog. General instructions on how to run samples are in *Getting Started with Adobe Version Cue CS3 Development* (getting-started.pdf).

You can add assets en masse into Version Cue, using the *ingest workflow*; for instance, you can drag-and-drop files from the desktop or from a folder in Adobe Bridge CS3 to a Version Cue project, to add the files directly to the Version Cue Server without them being copied to the local mount point of the project first. You have the opportunity to show a custom dialog when adding files in this way, if you override the normal check-in dialog of Version Cue, as well as when a new version of an existing file is being saved.

Figure 7 shows the dialog that is provided by sample.custommenu when adding files for the first time to a Version Cue project. If you examine the JavaScript code supplied with this sample, which is sent from the Version Cue Server to the client when a user visits a project for the first time, the code determines whether there is an existing version of the file on the Version Cue Server using the client scripting interface (methods on the class VCVersion) and shows one of two potential custom dialogs based on whether a file is just added or a new version of an existing file.

FIGURE 7 ***Custom Check-in Dialog on Ingest***


Create new file

Checkin Info

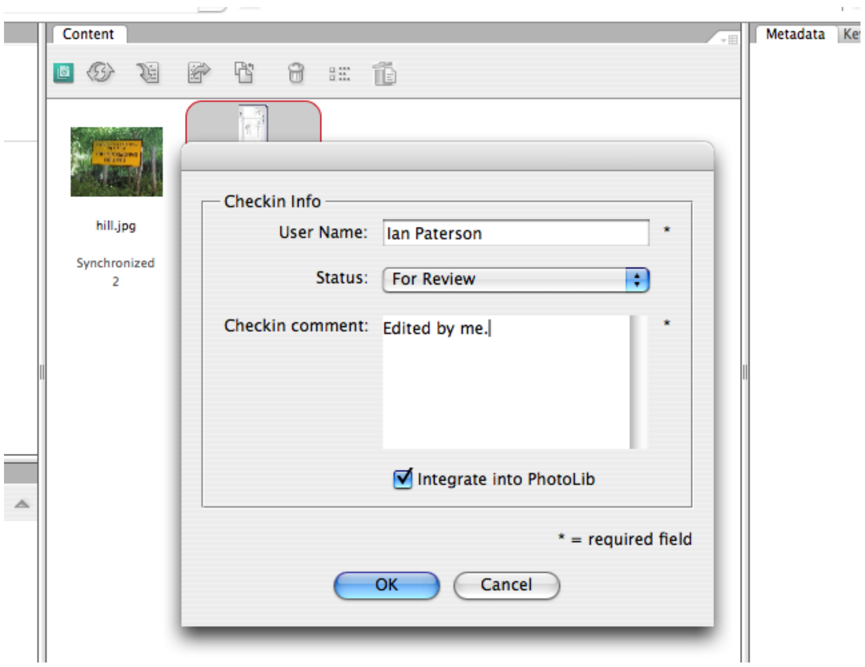
Checkin comment for new file(s): This is a file I have just added from the local file system *

* = required field

To test the metadata capture enforcement feature and make the customized check-in dialog appear, you need to open a file, modify it, and check in the modified version to a Version Cue project.

1. Ensure Version Cue Server is running with a plug-in like sample.customdialog loaded for the custom check-in dialog to appear.
2. Make sure you have at least one standard Version Cue project on the Version Cue Server, with at least one file.
3. Navigate to the standard Version Cue project with at least one file, so that a file can be seen in the Content pane of Bridge.
4. Select a file in the Content pane. Check-out the asset from the context-sensitive menu item.
5. With the checked-out file selected in the Content pane of Bridge, execute a right-mouse click (or ctrl-click for a single-button mouse on Mac OS) to bring up a context-sensitive menu. Use Reveal in Explorer to find out where the local replica file is located.
6. Modify the local replica file, and return to Bridge.
7. With the checked-out file selected, execute check-in from the context-sensitive menu associated with the file.
8. At this point you should see a custom check-in dialog, similar to that shown in Figure 8.
9. Note that the dialog has logic to enforce that you enter a check-in comment; if you do not do so, the OK button to dismiss the dialog is not enabled. This is a form of metadata enforcement, since it requires you to supply metadata about the file being checked in before you can successfully complete the check-in operation.

FIGURE 8 ***Custom Check-in Dialog***



Checkin Info

User Name: Ian Paterson *

Status: For Review

Checkin comment: Edited by me. *

☑ Integrate into PhotoLib

* = required field

Figure 8 shows a custom check-in dialog provided by the sample plug-in for Version Cue Server named sample.customdialog. The custom check-in dialog shows fields that include the name of the user responsible for the check-in, status, a check-in comment, and a check box indicating an operation that should be performed on check-in.

The OK button is enabled when data is entered into the name and check-in comment fields. A check-in comment must be supplied before check-in can proceed; this is enforced by JavaScript logic sent from the Server plug-in to the client— hence this is a form of metadata enforcement.

Clicking the OK button submits the data to the Version Cue Server. The Version Cue Server plug-in has a listener that gathers the data submitted through the custom dialog and creates a check-in comment, based on the field values from the custom dialog. The information gathered by the custom dialog is then visible in the version history.

The custom dialog is defined via an Adobe JavaScript library named ScriptUI and the JavaScript code behind the dialog can modified to your own requirements. The script sent by the Version Cue Server is located at the following path within the sample codebase for CustomDialog:

```
<workspace>/sample.customdialog/data/scripts/SaveAVersion.jsx
```

where <workspace> is whatever you chose as the workspace when setting up the IDE of Version Cue SDK.

If you modify the script when the Version Cue Server is running and the Version Cue client is connected, you must re-launch the client. The client loads the script only once per session.

For More Information

See Help within the IDE of VersionCueSDK: Help > Version Cue SDK Help > Samples Guide > CustomDialog

Customizing Menus

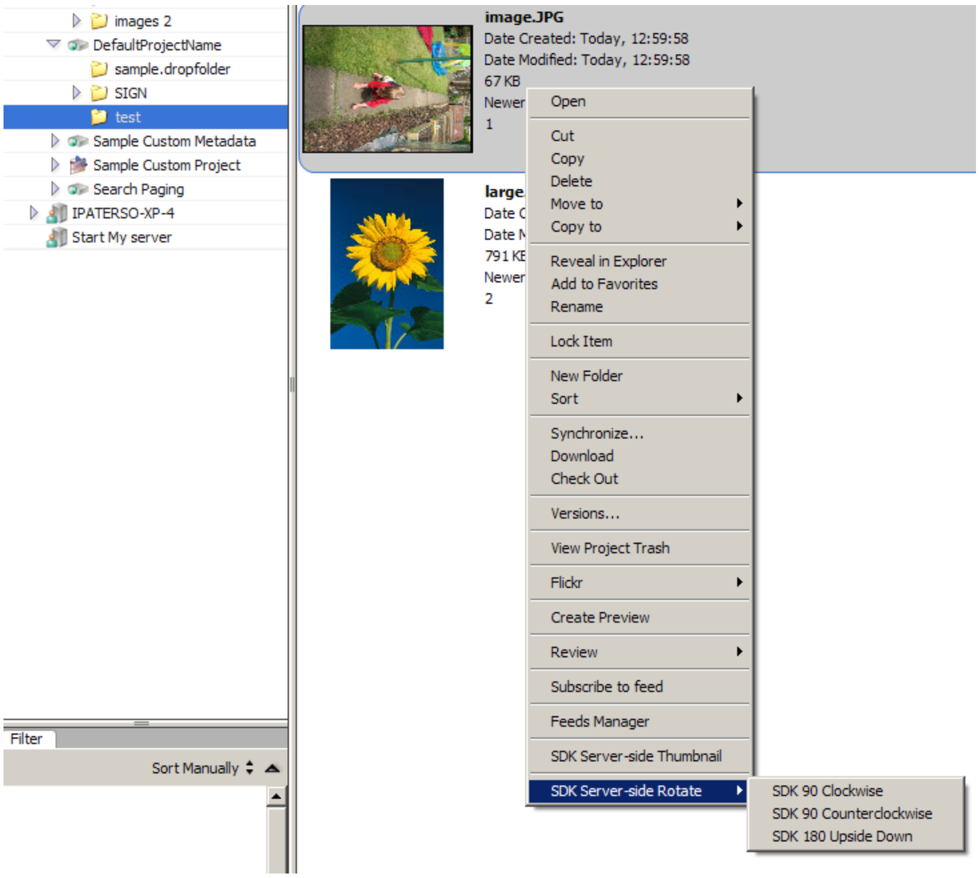
Version Cue extensibility enables you to contribute custom submenus and custom menu items in the following contexts:

- Tools menu in the Adobe File > Open dialog: Use the VersionCue.TOOLSMENU symbolic constant for the menu base path in your scripts.
- When a file is selected in contexts like the Adobe File > Open dialog or the Adobe Bridge CS3 content pane: use the VersionCue.CONTEXTMENU symbolic constant from your scripts.
- See the VersionCue.DOCUMENTMENU symbolic constant.
- See the VersionCue.MENUBAR symbolic constant.

To see the feature being exercised, you need to have started Version Cue Server and loaded the CustomMenu sample (sample.custommenu) from the Version Cue SDK.

Figure 9 shows the Bridge Content pane with a selected file, showing the context-sensitive menu item decorated with a custom submenu, "SDK Server-side Rotate", which offers options to rotate the selected image. There is also a menu item "SDK Server-side Thumbnail", which creates a custom thumbnail.

FIGURE 9 *Custom Menus within Adobe Bridge CS3 Content pane*



The Version Cue client receives JavaScript code— using a JavaScript library from Adobe named ScriptUI— that enables custom menus to be created when first visiting a Version Cue project. To see the scripts associated with creating the user interface, see the data/scripts folder of the sample.custommenu SDK sample. Note that you can add custom menu elements to both standard Version Cue projects and external projects, although the underlying mechanisms are slightly different in each case. See *Case Study: Connecting to an External System through Version Cue CS3* and examine the use cases associated with*Client User Interface Customization* for more detail.

The operations triggered by selecting these custom menu items in Bridge are executed by the sample plug-in named sample.custommenu on the Version Cue Server.

The code in sample.custommenu that executes on the Version Cue Server is invoked by receiving a custom SOAP message at the Server. The sample named sample.custommenu extends the default Version Cue protocol between client and Version Cue Server, to enable parameters to be passed from the clients like Bridge to control how the operation executes at the Server. For instance, parameters passed back to the Server identify the asset that was selected in Bridge and can specify other information like the angle of rotation or the quality and resolution of the thumbnail to create.

Restrictions

Support for custom menus has some limitations:

- The Version Cue client receives the custom menu information from the Version Cue Server when you enter the project for the first time; for instance, by double-clicking the project in the File > Open dialog. Before entering the project, the custom menu information is not available, which means that you are not able to contribute custom menu items to the context-sensitive menu shown when a project icon is selected in Adobe Bridge CS3 or the Adobe Common User Interface File > Open dialog, say.
- The menu information also is retrieved only once and cached at the Version Cue client. If you change anything on the Version Cue Server related to custom menus, you must restart the client to see the update.
- Custom menu information is not available for offline projects. This is as designed.

For More Information

See VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > CustomMenu.

See *Case Study: Connecting to an External System through Version Cue CS3*, which is available in PDF in the sdk.docs folder of the SDK and also accessible through VersionCueSDK IDE Help > Version Cue SDK Help >

Customizing the Search Interface

You can customize the search user interface of the Version Cue client. The Version Cue client can retrieve custom search criteria from Version Cue Server and send these custom criteria to a Version Cue Server plug-in, which can then trigger execution of the search in an external system, subject to the custom criteria. A search criterion contains the following key elements:

- A display name; for instance, "Processing Status."
- An operator; for instance, "contains."
- Optional predefined values, such as "approved."

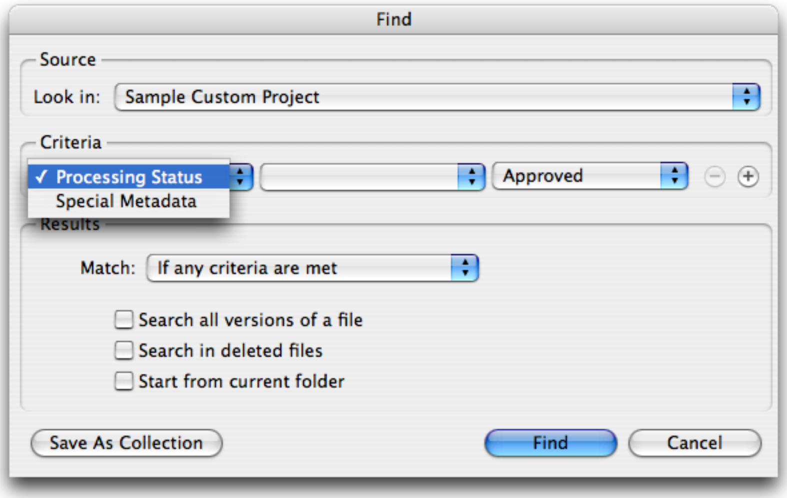
The user interface of the standard Version Cue client—that is, other than Bridge—shows only criteria of type "string" and "enum." For criteria of type "string," the user interface of the client displays a text field and silently uses the operator "contains" for the search operation. For criteria of type "enum," the text field in the user interface of the client changes to a drop-down list, which shows different predefined values; in this case, the operator "equals" is implicitly used by the search operation.

With Bridge, search criteria of all types (not just "string" and "enum") are shown, and end users can specify the operator to use for a particular search operation. The other feature of Bridge that is unavailable in other Version Cue clients is that Bridge can display multiple search criteria and allow you to make logical compositions of these criteria (such logical OR / logical AND) to create more complex search expressions.

Figure 10 shows custom search criteria, provided by a sample plug-in for Version Cue Server named sample.customproject. Note the custom search categories: "Processing Status" and

"Special Metadata". The former is of type "enum"; the latter, "string." You can select one of those new categories and retrieve search results from the Version Cue Server.

FIGURE 10 ***Project Search over Sample Custom Project***


Find
Source
Look in: Sample Custom Project
Criteria
✓ Processing Status
Special Metadata
Approved
Results
Match: If any criteria are met
☐ Search all versions of a file
☐ Search in deleted files
☐ Start from current folder

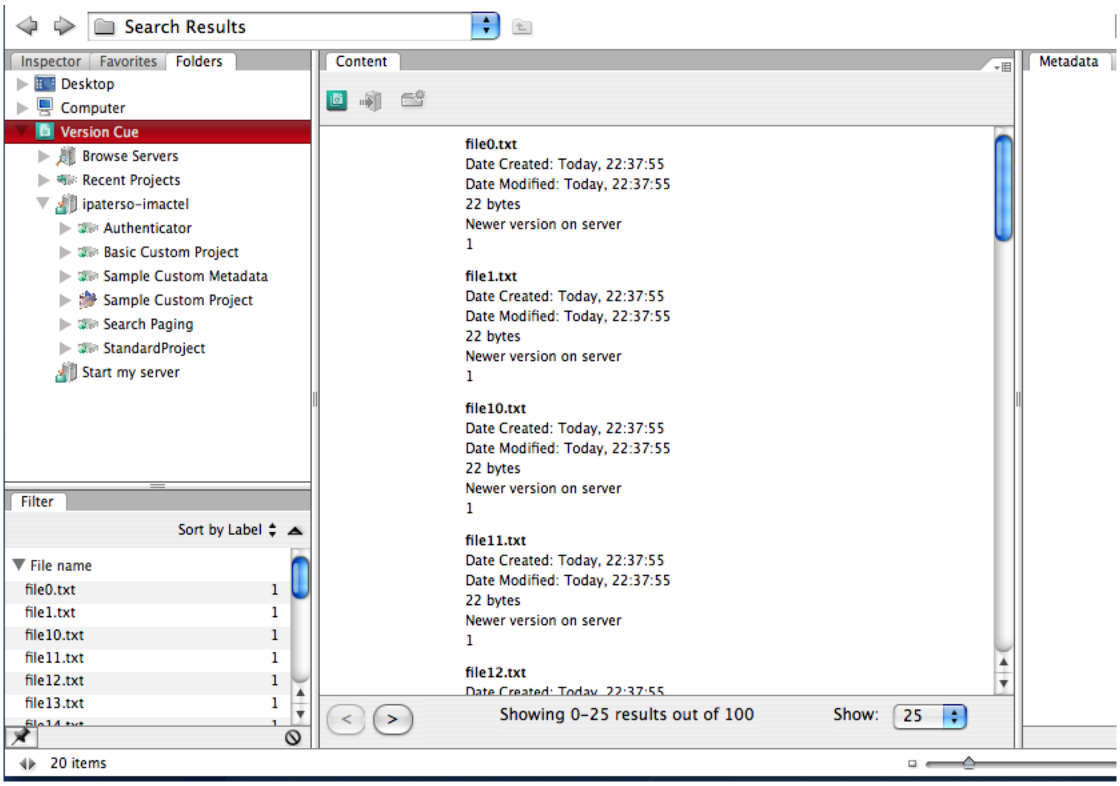
If you have the plug-in loaded, to see this dialog, navigate within the Adobe Bridge CS3 to "Sample Custom Project" and execute Edit > Find once the root folders of the project are visible in the Content pane of Bridge.

To change the default search results that are returned by sample.customproject, you must do the following:

1. Update the values within the existing .txt files in the samples.customproject/data folder, or add new .txt files to the data folder.
2. Restart the Version Cue Server.
3. Run the search query again.

Figure 11 shows an illustration of returned chunked or paged result sets to Adobe Bridge CS3; note how there are controls to allow the user to navigate through the result set pages. You can customize the size of these result sets sent from Version Cue Server to the Version Cue client—for instance, you can return sets of partial search results rather than the complete set in one message— if you support paged result sets in your custom search controller. For a sample plug-in for Version Cue Server that implemented this scheme, see sample.searchpaging.

FIGURE 11 Project Search with Paged Result Set



File name	
file0.txt	1
file1.txt	1
file10.txt	1
file11.txt	1
file12.txt	1
file13.txt	1

```
file0.txt
Date Created: Today, 22:37:55
Date Modified: Today, 22:37:55
22 bytes
Newer version on server
1

file1.txt
Date Created: Today, 22:37:55
Date Modified: Today, 22:37:55
22 bytes
Newer version on server
1

file10.txt
Date Created: Today, 22:37:55
Date Modified: Today, 22:37:55
22 bytes
Newer version on server
1

file11.txt
Date Created: Today, 22:37:55
Date Modified: Today, 22:37:55
22 bytes
Newer version on server
1

file12.txt

Showing 0-25 results out of 100
```

For More Information

See the section on custom search criteria in the plug-in sample documentation: VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > Custom Project.

See the plug-in sample documentation: VersionCueSDK IDE Help > Version Cue SDK Help > Samples Guide > Search Paging.

Version Cue Client Architecture

Version Cue Integration

Desktop applications in Creative Suite integrate with Version Cue; the depth of integration varies between applications. Some of these applications support the basic File > Check-in function. Bridge has the deepest integration with Version Cue and exposes some features not accessible through other clients.

Version Cue clients like InDesign and Photoshop are *editing* applications. Integration with Version Cue enables version-controlled assets to be placed in InDesign documents, or different versions of an image to be created in Photoshop. Integration with Version Cue enables InDesign to change the state of a link in its user interface when an asset is updated on a Version Cue Server by another user, as if it were a local file (rather than a managed asset). The Version Cue asset-management features are largely transparent to the end user of applications like InDesign and Photoshop.

On the other hand, Adobe Bridge CS3 is primarily a file browser and searching tool, rather than an editing application. It is a particularly privileged Version Cue client, and there are many features of Version Cue that can be accessed only through Bridge; for instance, if you want to compose a search based on more than one search criterion.

When using a Version Cue client, you can browse assets that are not on the Version Cue Server (only in the local file system), as well as ones that are on the Version Cue Server and, hence, version controlled. These assets appear quite similar in the Adobe Common User Interface, except some status information, which distinguishes those managed by the Version Cue Server from those solely in the local file system of the Version Cue client. This is the key benefit of Version Cue from the end-user's perspective; he does not have to learn a new idiom to navigate through an asset management system, as it can look just like he is browsing through a file system. In other words, Version Cue implements a virtual file system for its clients.

Client Libraries

Version Cue is a client-server application; see *Adobe Version Cue CS3 Server Programmer's Guide* for a discussion of its architecture. The server is Version Cue Server, and the clients are desktop applications in Creative Suite.

Applications like InDesign and Photoshop are integrated with Version Cue via the Version Cue client libraries. The libraries support features like asset management— such as check-out and check-in from Version Cue— and communication with Version Cue Server. The libraries also provide user interface dialogs, which enhance the normal operating system Open and Save dialogs; for instance, these enable end users to place a file from Version Cue, search for files, view collaboration status, and communicate with a Version Cue Server.

Client applications can use the Version Cue File > Open and File > Save dialogs in place of the normal operating system dialogs. Version Cue also provides user interface elements to display the historical versions of an asset and error conditions.

The details of the Version Cue client libraries are largely hidden from external developers, although some description of the library components is given in the *Adobe Version Cue CS3 Server Programmer's Guide*. The scripting interface to Version Cue enables you to create user interface by using the Adobe ScriptUI, as well as to perform Version Cue specific operations using the scripting interface to the VersionCueSDK client library.

Server-driven Scripting

The Version Cue client libraries expose a C++ API (the "Q-API"), which is used by the developers of Creative Suite applications to integrate with Version Cue. It would not be straightforward to use this C++ API to integrate with Version Cue as an external developer; you would have to write a separate plug-in for each desktop application in Creative Suite. Since writing a

Version Cue Server plug-in provides a single point of integration with Version Cue for adding server-side functionality, the same integration model is followed on the client side; that is, having a single point of integration to add client-side functionality.

The mechanism that makes possible a single point of integration for client-side extensibility is server-driven scripts sent to the client. This should *not* be confused with server-side scripting; the scripts run at the client, but they are sent by the Version Cue Server to the client.

You can transmit JavaScript code from a Version Cue Server plug-in to Version Cue client applications on certain events at the Version Cue Server. For instance, the first time an end user of a Version Cue client application visits a standard Version Cue project, JavaScript code can be sent from a Version Cue Server plug-in to the Version Cue client, to enable custom user interface and logic to execute on particular events at the client.

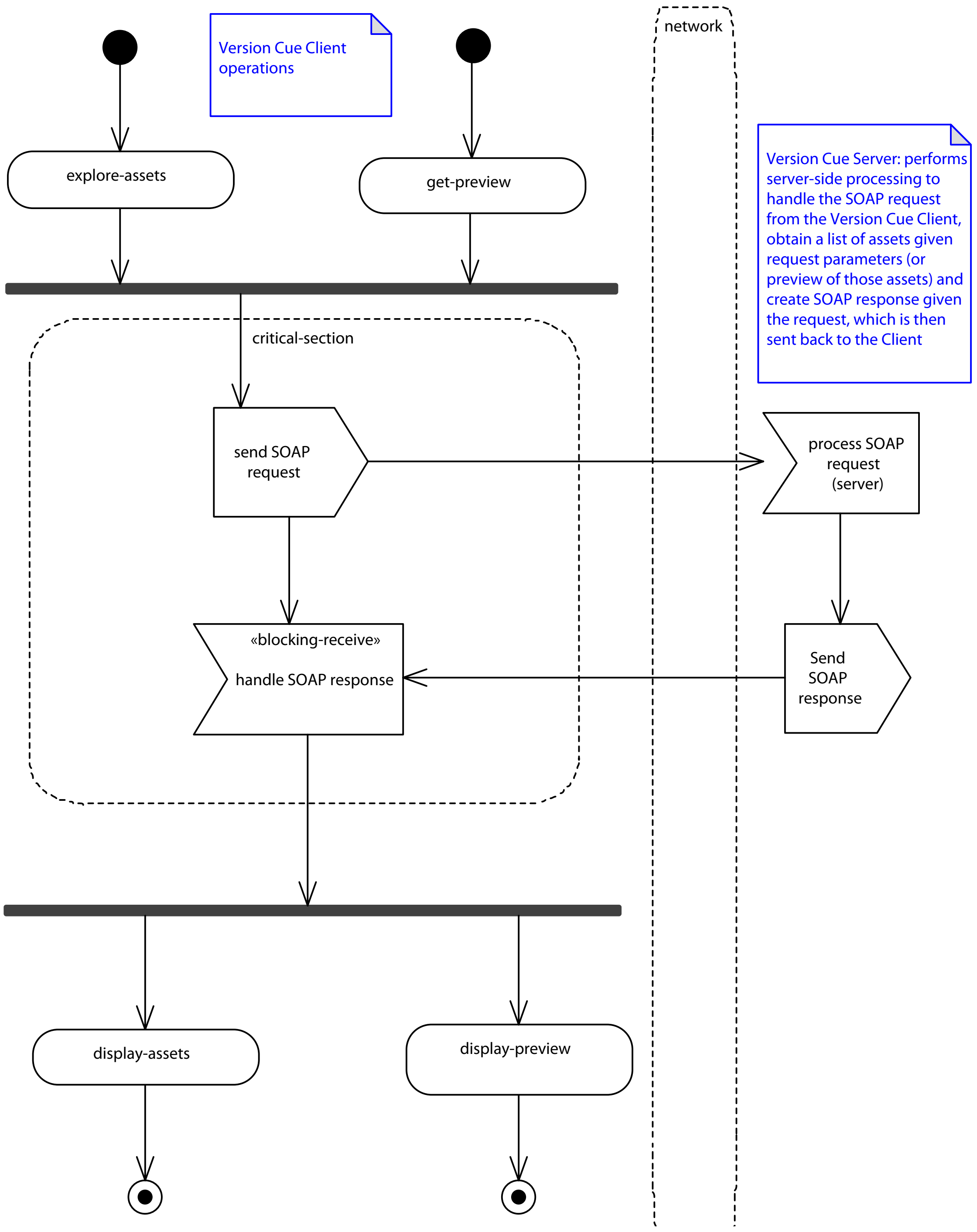
To learn more about the architecture that supports this server-driven scripting integration model, see *Adobe Version Cue CS3 Server Programmer's Guide*.

Multi-threading

Version Cue clients are Adobe applications that are integrated with the Version Cue client libraries. The Version Cue client library has a multi-threaded core, but sending and receiving SOAP messages is serialized: although a single Version Cue client can have multiple threads executing the library code, only one of its thread is sending/receiving at any one time.

The critical section in the client is protected by a synchronization object, a "baton" or *mutex*. In a relay race, even if there are four runners in the team, only one at a time can hold the baton, and similarly with running threads and the mutex. In Figure 12, two hypothetical requests are shown, both of which try to acquire the mutex synchronization object, protecting access to the critical section. The first to acquire the mutex proceeds to execute in the critical section, while the other thread is blocked.

FIGURE 12 *Conceptual Threading Model for Version Cue Client*


Version Cue Client operations
network
explore-assets
get-preview
Version Cue Server: performs server-side processing to handle the SOAP request from the Version Cue Client, obtain a list of assets given request parameters (or preview of those assets) and create SOAP response given the request, which is then sent back to the Client
critical-section
send SOAP request
process SOAP request (server)
«blocking-receive»
handle SOAP response
Send SOAP response
display-assets
display-preview

The thread executing in the critical section sends a SOAP request, receives a SOAP response (or a time-out period is exceeded, if there is network congestion or a Version Cue Server is unavailable), and leaves the critical section, releasing the mutex. The next thread acquires the mutex, enters the critical section, sends the SOAP request, waits for a SOAP response (or time-out period to be exceeded), and so on.

A single Version Cue client issues only one SOAP request at a time; however, if you have multiple Version Cue clients (such as Bridge and Photoshop) on one machine, they can issue concurrent requests to the Version Cue Server. There also can be multiple machines with Version Cue clients issuing concurrent requests to a single Version Cue Server. To ensure responsiveness at each client, the Version Cue Server must support a high degree of request concurrency.

Scripting Environment

Before you Begin

To execute, test and debug Adobe ExtendScript and ScriptUI scripts as well as JavaScript code that uses the Version Cue client scripting interface, you *must* install and use Adobe ExtendScript Toolkit 2. Note that ExtendScript Toolkit 1, which was installed by Creative Suite CS2, is not compatible with developing JavaScript code for Version Cue CS3 clients.

The ExtendScript Toolkit is a JavaScript integrated development environment (IDE); it is described in more detail in the *JavaScript Tools Guide* in the Adobe Bridge CS3 SDK, which describes developing JavaScript code with the ExtendScript Toolkit.

ExtendScript Toolkit is a source-level debugger for JavaScript; you can use it to execute and debug JavaScript code written against the Version Cue client scripting interface— for instance, you can set breakpoints and single-step through the JavaScript code. JavaScript code can be opened and edited directly in the ExtendScript Toolkit or pasted into an open editor window in the Toolkit from another editor if you prefer.

ExtendScript: a JavaScript implementation

ExtendScript is an implementation of the ECMA-262 JavaScript specification, developed by Adobe and used by Creative Suite applications with a scripting interface. To support developing and debugging, ExtendScript provides features and utilities above and beyond the ECMA-262 specification. ExtendScript also supports integration of XML data into JavaScript, according to the ECMA-357 specification— it supports a subset of that implementation we call Mini-E4X.

For help in developing, debugging, and testing scripts, ExtendScript provides the following:

- The ExtendScript Toolkit itself, which has a full-featured source-level debugger and development environment for writing JavaScript code.
- A global debugging object, the dollar ($) object, which can emit trace to the JavaScript console in the ExtendScript Toolkit. See "Debugging, Logging, and Profiling with the Toolkit" on page 26.
- A reporting utility for ExtendScript elements, the ExtendScript Reflection Interface. For instance, this enables you to dump properties for any class.

To identify specific Creative Suite applications, scripts must use the correct application and namespace specifiers, described in "Application and Version Specifiers" on page 27. These can target not only an application at a specific version level, but also a particular localized version.

ExtendScript has a localization utility for providing user interface string values in different languages. ExtendScript supports preprocessor directives, to allow scripts to be combined using #include and other directives; see "Preprocessor Directives in ExtendScript" on page 27. The BridgeTalk component allows inter-application communication through scripting; for instance, one Creative Suite application can send JavaScript code to another through this BridgeTalk mechanism. This is described in the Bridge JavaScript documentation.

To find out more about ExtendScript, see the *JavaScript Tools Guide* in the Adobe Bridge CS3 SDK.

Debugging, Logging, and Profiling with the Toolkit

The ExtendScript Toolkit has a comprehensive set of features typical of an interactive source-level debugger, including the following:

- Setting normal and conditional breakpoints within a script.
- A JavaScript console window, which shows trace output; for instance, emitted using the Dollar object.
- Inspecting the state of variables in a script halted within the script debugger.
- Single-stepping through lines of script code: stepping into, over, and out of functions.
- Displaying the call stack and letting you switch scope to different functions in the stack.

NOTE: You can't just set a breakpoint through the IDE of ExtendScript Toolkit to break at specific lines of code in your scripts when they are sent from the server. To debug server-driven scripts for Version Cue, add "$.bp() ;" to set a breakpoint in your script sent from the Version Cue Server to the Version Cue client. You also add "VersionCue.debugLevel = 1;" to enable the debugger. When the script executes, this will halt ExtendScript Toolkit at the line you added. See the JavaScript Tools Guide in the Adobe Bridge CS3 SDK for more information on using the dollar object methods and properties.

NOTE: The Version Cue client downloads the script when the project is first entered; thereafter, the client needs to be restarted. This means if you change the script, you must restart the client to have it execute the new script correctly.

The JavaScript console is a key feature of the Toolkit. In addition to seeing trace output, you can use its command line to execute lines of JavaScript, in the current execution context, and also see directly the results of executing JavaScript in the output field.

A key feature of ExtendScript for debugging is the built-in dollar object, used as follows:

```
$.writeln(...);
```

A profiling tool is built into the ExtendScript toolkit, letting you identify hot spots in executing your JavaScript code.

For details about the features offered by the Toolkit to develop, debug, and test JavaScript code, see the *JavaScript Tools Guide* in the Adobe Bridge CS3 SDK.

Preprocessor Directives in ExtendScript

By default, ExtendScript supports the use of certain preprocessor directives and import directives, to enable you to compose a script by including script fragments. This is particularly useful to write object-oriented JavaScript, and perhaps keep one class per file, as in the Java model.

For details about this ExtendScript feature, see the *JavaScript Tools Guide* in the Adobe Bridge CS3 SDK.

This feature is not likely to work in a server-driven scripting environment.

Application and Version Specifiers

If you have scripting code you want to run only in a particular client application—for example, if you only want some JavaScript code to execute in the context of Adobe Bridge CS3— the recommended way to write this is as follows:

```
if(BridgeTalk.appName == 'bridge') {
     // Continue with the Bridge-specific logic here
}
```

You also can use another static property of the BridgeTalk class to determine which version of an application is installed:

```
if(BridgeTalk.getSpecifier == 'photoshop', 10) {
     // Continue with logic that depends on Photoshop CS3 here
}
```

Adobe BridgeTalk is an inter-application communication framework that can be used for scripting; it is described in the *JavaScript Tools Guide* in the Adobe Bridge CS3 SDK.

Writing Client-side Scripts for the First Time

Using the Version Cue Client Scripting Engine

The recommended way to develop JavaScript code for CS3 clients is to use the ExtendScript Toolkit 2, which is bundled with Creative Suite CS3 or can be separately installed. In this section, we examine two different ways to develop and test JavaScript code using the ExtendScript Toolkit.

To exercise Version Cue features, use Adobe Bridge CS3 or other Version Cue clients such as Adobe Photoshop CS3 and Adobe InDesign CS3.

Enabling the Version Cue Client Scripting Interface from other Scripting Contexts

There is one contract that you need to be aware of before using the client scripting interface to Version Cue from a context such as a JavaScript running against the Adobe Bridge CS3 scripting engine. If you want to access the client scripting, you must explicitly load (and unload) the shared library that contains the scripting wrapper layer. Note that when you navigate into a custom project— for instance, "Sample Custom Project", supported by the plug-in named sample.customproject— this loading and unloading of the shared library containing the client scripting interface is done for you.

VersionCueSDKLoader.jsx is a startup script that point products in Creative Suite (such as InDesign, Bridge, Illustrator) load on startup. This file VersionCueSDKLoader.jsx contains two methods, loadVCSDK and unLoadVCSDK, which you can use can use to load and unload the VersionCue client SDK shared libraries (versioncuesdk.dll/framework) from your own Javascript code using the ExternalObject mechanism. After loading these shared libraries, you can use the Version Cue client scripting interface from your own JavaScript code. The reason you may be doing this is because you want to write some code that runs within the Bridge scripting engine, but still makes use of the Version Cue client scripting interface.

For instance, if you wanted to use the classes in the Version Cue client scripting interface from Adobe Bridge CS3, you can load and use the Version Cue classes with something like this fragment of JavaScript:

```
if(VersioncueSDKLoader.loadVCSDK() != null)
{
     // Use Versioncue scripting interface classes here
 ...
 VersionCueSDKLoader.unLoadVCSDK();
}
else
{
     alert ( "Unable to load versioncueSDK");
}
```

Example: Hello World

One of the most elementary scripts you might write is to display an alert with "Hello World":

```
Window.alert("Hello World.");
```

To run this script within the Version Cue client scripting engine, follow these steps:

1. Launch a Version Cue client application.
2. Launch the installed ExtendScript Toolkit 2.0 if you have not already done so.
3. Set the target application in the Toolkit to be "Adobe Bridge."
4. Within the Toolkit, select File > New JavaScript to get an empty editor window.

5. Enter the code below into the editor window:

   ```
   Window.alert("Hello World.");
   ```

6. Within the Toolkit, run the script you just created, by selecting Debug > Run or using the Run icon button on the toolbar (a rightward-pointing triangle).

7. Verify that you can see an alert popped from the context of the Version Cue client.

There is an even terser "Hello World" script we can write, using the dollar ($) object, which is available as a debugging aid within the ExtendScript implementation of JavaScript and emits output to the JavaScript console of the ESTK:

```
$.writeln("Hello World");
```

Follow these steps:

1. Launch the Version Cue client if it is not already running.

2. Launch the installed ExtendScript Toolkit 2, if you have not already done so.

3. Within the Toolkit, enter the script below directly into the command line of the JavaScript Console (an edit-box at the top of the Console panel):

   ```
   $.writeln("Hello World");
   ```

4. Press Return (Enter) to have the code execute and display its results to the JavaScript Console.

5. Verify that you see a message printed out, followed by "Result: undefined." The writeln function of the dollar class returns "undefined."

Example: List Version Cue Servers

In the next example, we execute code to list the Version Cue Servers known to the CS3 client. Note that this and also Example 2 you will execute directly against the Bridge main scripting engine as if the script were installed on the client machine, rather than sent by a Version Cue Server. This means you will be able to set break-points in them directly from the ExtendScript Toolkit IDE, which you would not be able to do with a server-driven script— in the latter case, you need to use $.bp().

NOTE: To make sure that the global objects added by the Version Cue client scripting interface are available, you have to make sure that the Version Cue client SDK is loaded, if you want to run this script in a context such as the main Adobe Bridge CS3 scripting engine. See "Enabling the Version Cue Client Scripting Interface from other Scripting Contexts" on page 28.

NOTE: Note that the shared library exposing the Version Cue client scripting interface is loaded automatically when you navigate into a sample like Sample CustomProject. Alternatively, you can call VersionCueSDKLoader.loadSDK() and then once you are finished running client scripts, you can call VersionCueSDKLoader.unLoadSDK().

Follow these steps:

1. If you have not already done so, launch the ExtendScript Toolkit 2.0.
2. To ensure you have at least one Version Cue Server available to the client, launch Version Cue Server in the usual fashion, as described in *Getting Started with Adobe Version Cue CS3 Development*.
3. Enter the code shown in Example 1. Make sure “Adobe Bridge” is the target application in the Toolkit.
4. Execute the script. You should see a list of Version Cue Servers accessible from the client.

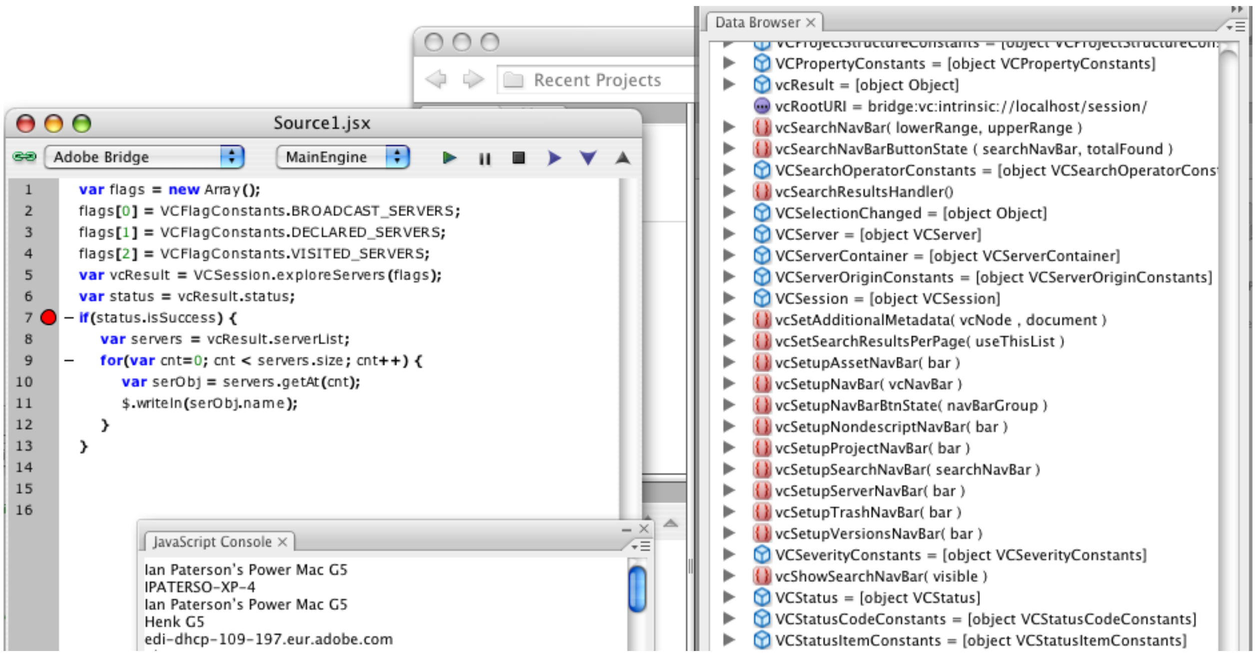
The VCSession class used in Example 1 has only class methods and class-level properties; there are no associated instance methods or instance properties on VCSession.

EXAMPLE 1

```
var flags = new Array();
flags[0] = VCFlagConstants.BROADCAST_SERVERS;
flags[1] = VCFlagConstants.DECLARED_SERVERS;
flags [2] = VCFlagConstants.VISITED_SERVERS;
var vcResult = VCSession.exploreServers(flags);
var status = vcResult.status;
if(status.isSuccess)
{
 //serverList is the container to hold Servers
 var servers = vcResult.serverList;
 for(var cnt = 0; cnt < servers.size; cnt++)
 {
 var serObj = servers.getAt(cnt);
 $.writeln(serObj.name);
 }
}
else
{
 Window.alert("ExploreServer call failed\n");
 }
```

Figure 13 shows the result of running Example 1 in the ExtendScript Toolkit.

FIGURE 13 ***Running Example 1 in ExtendScript Toolkit***



```
var flags = new Array();
flags[0] = VCFlagConstants.BROADCAST_SERVERS;
flags[1] = VCFlagConstants.DECLARED_SERVERS;
flags[2] = VCFlagConstants.VISITED_SERVERS;
var vcResult = VCSession.exploreServers(flags);
var status = vcResult.status;
if(status.isSuccess) {
    var servers = vcResult.serverList;
    for(var cnt=0; cnt < servers.size; cnt++) {
        var serObj = servers.getAt(cnt);
        $.writeln(serObj.name);
    }
}
```

```
Ian Paterson's Power Mac G5
IPATERSO-XP-4
Ian Paterson's Power Mac G5
Henk G5
edi-dhcp-109-197.eur.adobe.com
```

```
VCProjectStructureConstants = [object VCProjectStructureCon
VCPropertyConstants = [object VCPropertyConstants]
vcResult = [object Object]
vcRootURI = bridge:vc:intrinsic://localhost/session/
vcSearchNavBar( lowerRange, upperRange )
vcSearchNavBarButtonState ( searchNavBar, totalFound )
VCSearchOperatorConstants = [object VCSearchOperatorCons
vcSearchResultsHandler()
VCSelectionChanged = [object Object]
VCServer = [object VCServer]
VCServerContainer = [object VCServerContainer]
VCServerOriginConstants = [object VCServerOriginConstants]
VCSession = [object VCSession]
vcSetAdditionalMetadata( vcNode , document )
vcSetSearchResultsPerPage( useThisList )
vcSetupAssetNavBar( bar )
vcSetupNavBar( vcNavBar )
vcSetupNavBarBtnState( navBarGroup )
vcSetupNondescriptNavBar( bar )
vcSetupProjectNavBar( bar )
vcSetupSearchNavBar( searchNavBar )
vcSetupServerNavBar( bar )
vcSetupTrashNavBar( bar )
vcSetupVersionsNavBar( bar )
VCSeverityConstants = [object VCSeverityConstants]
vcShowSearchNavBar( visible )
VCStatus = [object VCStatus]
VCStatusCodeConstants = [object VCStatusCodeConstants]
VCStatusItemConstants = [object VCStatusItemConstants]
```

Example: List Projects in a Version Cue Server

In the next example, we execute code to list the names of projects in a Version Cue Server. We assume that the client is Adobe Bridge CS3. Follow these steps:

1. Launch Version Cue Server, if you have not already done so.
2. Make sure you have at least one project in the Version Cue Server.
3. If you have not already done so, launch Adobe Bridge CS3.
4. Launch the installed ExtendScript Toolkit 2.0.
5. Enter the code shown in Example 2. Make sure "Adobe Bridge" is the target application in the Toolkit.
6. Execute the script. You should see a list projects on the Version Cue Server.

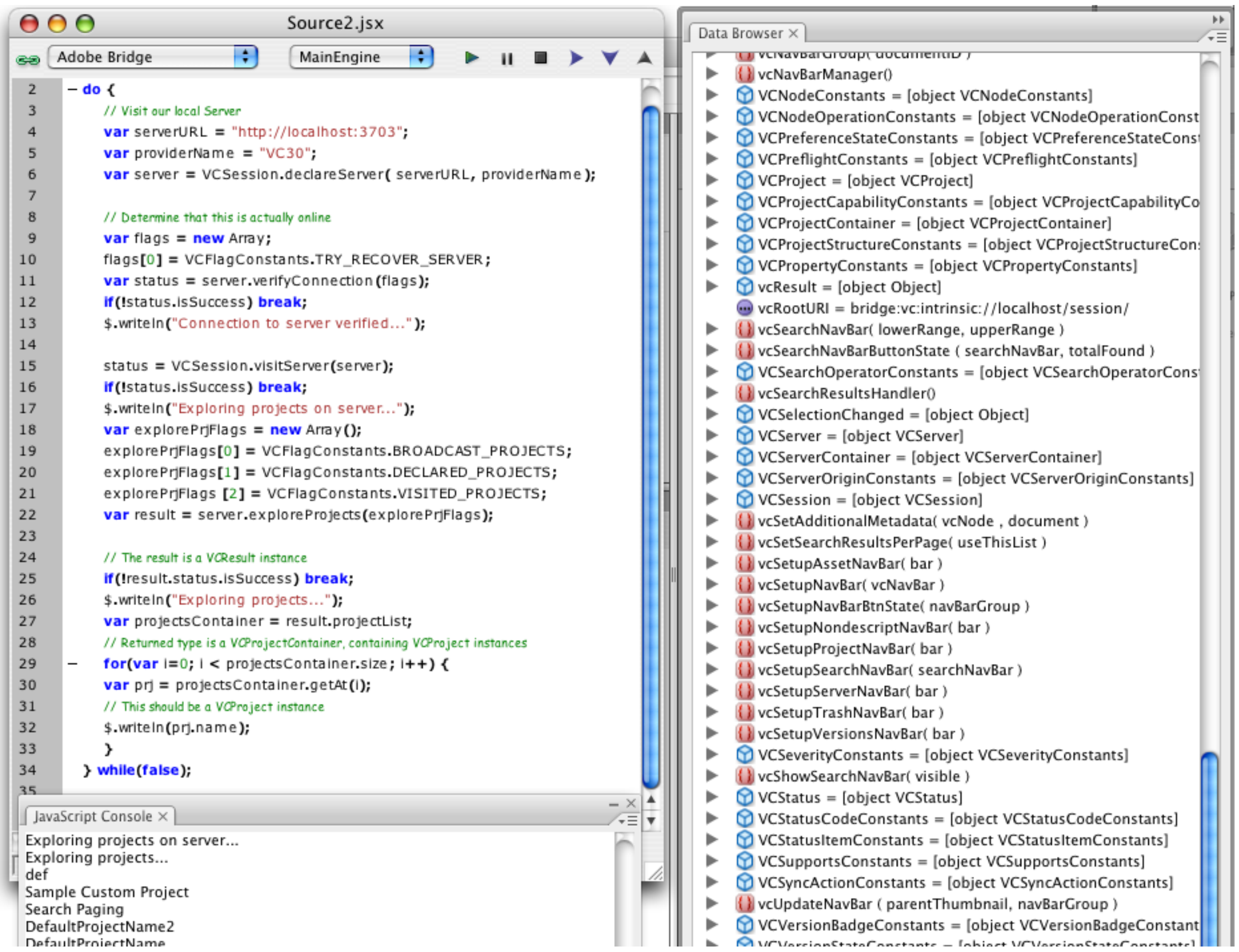
Ideally, you should be handling error conditions in a more graceful way than in the code fragment in Example 2, where detailed error handling is omitted in the interest of brevity. Note that we declare the server as "VC20", because assuming a CS3 server when it is really CS2 will cause problems and in general you should always assume CS2 (and use "VC20") until it is proven otherwise.

EXAMPLE 2

```
do {
      // Visit our local Server
      var serverURL = "http://localhost:3703";
      var providerName = "VC20";
      var server = VCSession.declareServer( serverURL, providerName);

      // Determine that this is actually online
      var flags = new Array;
      flags[0] = VCFlagConstants.TRY_RECOVER_SERVER;
      var status = server.verifyConnection(flags);
      if(!status.isSuccess) break;
      $.writeln("Connection to server verified...");

      status = VCSession.visitServer(server);
      if(!status.isSuccess) break;
      $.writeln("Exploring projects on server...");
      var explorePrjFlags = new Array();
      explorePrjFlags[0] = VCFlagConstants.BROADCAST_PROJECTS;
      explorePrjFlags[1] = VCFlagConstants.DECLARED_PROJECTS;
      var result = server.exploreProjects(explorePrjFlags);

      // The result is a VCResult instance
      if(!result.status.isSuccess) break;
      $.writeln("Exploring projects...");
      var projectsContainer = result.projectList;
      // Returned type is a VCProjectContainer, containing VCProject instances
      for(var i=0; i < projectsContainer.size; i++) {
      var prj = projectsContainer.getAt(i);
      // This should be a VCProject instance
      $.writeln(prj.name);
      }
} while(false);
```

Figure 14 shows the result of running Example 2 in the ExtendScript Toolkit.

FIGURE 14 ***Running Example 2 in ExtendScript Toolkit***



```
- do {
      // Visit our local Server
      var serverURL = "http://localhost:3703";
      var providerName = "VC30";
      var server = VCSession.declareServer( serverURL, providerName );

      // Determine that this is actually online
      var flags = new Array;
      flags[0] = VCFlagConstants.TRY_RECOVER_SERVER;
      var status = server.verifyConnection(flags);
      if(!status.isSuccess) break;
      $.writeln("Connection to server verified...");

      status = VCSession.visitServer(server);
      if(!status.isSuccess) break;
      $.writeln("Exploring projects on server...");
      var explorePrjFlags = new Array();
      explorePrjFlags[0] = VCFlagConstants.BROADCAST_PROJECTS;
      explorePrjFlags[1] = VCFlagConstants.DECLARED_PROJECTS;
      explorePrjFlags [2] = VCFlagConstants.VISITED_PROJECTS;
      var result = server.exploreProjects(explorePrjFlags);

      // The result is a VCResult instance
      if(!result.status.isSuccess) break;
      $.writeln("Exploring projects...");
      var projectsContainer = result.projectList;
      // Returned type is a VCProjectContainer, containing VCProject instances
-     for(var i=0; i < projectsContainer.size; i++) {
      var prj = projectsContainer.getAt(i);
      // This should be a VCProject instance
      $.writeln(prj.name);
      }
  } while(false);
```

```
Exploring projects on server...
Exploring projects...
def
Sample Custom Project
Search Paging
DefaultProjectName2
DefaultProjectName
```

```
vcNavBarGroup( documentID )
vcNavBarManager()
VCNodeConstants = [object VCNodeConstants]
VCNodeOperationConstants = [object VCNodeOperationConst
VCPreferenceStateConstants = [object VCPreferenceStateConst
VCPreflightConstants = [object VCPreflightConstants]
VCProject = [object VCProject]
VCProjectCapabilityConstants = [object VCProjectCapabilityCo
VCProjectContainer = [object VCProjectContainer]
VCProjectStructureConstants = [object VCProjectStructureCon
VCPropertyConstants = [object VCPropertyConstants]
vcResult = [object Object]
vcRootURI = bridge:vc:intrinsic://localhost/session/
vcSearchNavBar( lowerRange, upperRange )
vcSearchNavBarButtonState ( searchNavBar, totalFound )
VCSearchOperatorConstants = [object VCSearchOperatorCons
vcSearchResultsHandler()
VCSelectionChanged = [object Object]
VCServer = [object VCServer]
VCServerContainer = [object VCServerContainer]
VCServerOriginConstants = [object VCServerOriginConstants]
VCSession = [object VCSession]
vcSetAdditionalMetadata( vcNode , document )
vcSetSearchResultsPerPage( useThisList )
vcSetupAssetNavBar( bar )
vcSetupNavBar( vcNavBar )
vcSetupNavBarBtnState( navBarGroup )
vcSetupNondescriptNavBar( bar )
vcSetupProjectNavBar( bar )
vcSetupSearchNavBar( searchNavBar )
vcSetupServerNavBar( bar )
vcSetupTrashNavBar( bar )
vcSetupVersionsNavBar( bar )
VCSeverityConstants = [object VCSeverityConstants]
vcShowSearchNavBar( visible )
VCStatus = [object VCStatus]
VCStatusCodeConstants = [object VCStatusCodeConstants]
VCStatusItemConstants = [object VCStatusItemConstants]
VCSupportsConstants = [object VCSupportsConstants]
VCSyncActionConstants = [object VCSyncActionConstants]
vcUpdateNavBar ( parentThumbnail, navBarGroup )
VCVersionBadgeConstants = [object VCVersionBadgeConstant
VCVersionStateConstants = [object VCVersionStateConstants]
```

For More Information

See the following resources:

- The *Scripting Interface Reference* in the Version Cue SDK, for information on working with the scripting interface to the Version Cue client libraries.
- The *JavaScript Tools Guide CS3* in the Bridge CS3 SDK, for information on the ExtendScript Toolkit and the Adobe JavaScript library for creating user interfaces named ScriptUI. Note the ExtendScript File and Folder objects mentioned there are not implemented in the Version Cue scripting interface.

References

- ECMA-262: ECMAScript Language Specification

  `http://www.ecma-international.org/publications/standards/Ecma-262.htm`
- ECMA-357: ECMAScript for XML (E4X) Specification

  `http://www.ecma-international.org/publications/standards/Ecma-357.htm`
- *Adobe Bridge CS3 SDK*. Adobe Systems Inc., 2007.
- *Version Cue Client Scripting Interface Reference*. Integrated with Help in the IDE of Adobe Version Cue CS3 SDK.