PureWeb® STK 3.1

Quick Start Guide: Flex

About Calgary Scientific

Calgary Scientific Inc., is dedicated to providing advanced web-enablement, mobility enhancement and advanced visualization solutions to industries looking for secure access and use of their data or graphics intensive applications, while using their existing systems. Visit www.calgaryscientific.com for more information.

Notice

Although reasonable effort is made to ensure that the information in this document is complete and accurate at the time of release, Calgary Scientific Inc., cannot assume responsibility for any existing errors. Changes and/or corrections to the information contained in this document may be incorporated in future versions.

Your Responsibility for Your System's Security

You are responsible for the security of your system. Product administration to prevent unauthorized use is your responsibility. Your system administrator should read all documents provided with this product to fully understand the features available that reduce your risk of incurring charges for unlicensed use of Calgary Scientific products.

Trademarks

Technical Support from Calgary Scientific

Please contact Calgary Scientific Support at the regional numbers provided on page 7.

Ordering and Licensing Information

The PureWeb license is an ASCII text file with a .lic extension. The license is not part of the PureWeb Server installation, and must be acquired from Calgary Scientific Inc. When you receive your license, copy the .lic file in to the C:\CSI\PureWeb\Server\conf\ directory and (re)start the server.

Released by

Calgary Scientific Inc. www.calgaryscientific.com.

Document Version: PW3.1_Flex_QSG_06-2012_v3.1000.00

Table of Contents

List of Tasks 4

List of Procedures 5

Preface 6

Intended Audience 6
Reading Recommendations 6
Making Comments on This Document 7
Contacting Calgary Scientific Support 7
Common Development Tasks 7

Chapter 1 **Introducing PureWeb® STK 9**

About PureWeb® STK 9
Basic Architecture 10
Prerequisites For Flex Development 12
About the Sample Application 12

Chapter 2 **Introducing Scribble 13**

About Scribble 13
ScribbleApp Code Description 14
ScribbleClient Code Description 14
Build Scribble 17
Run Scribble 23
Sample Functionality 29
Debug ScribbleApp 35

Supplements **Related Documentation Resources 38**

Document Conventions 39

Index 41

List of Tasks

Development Tasks 7
Scribble Application Tasks 8

List of Procedures

Building Scribble Service Application 17
Building the Scribble Client Application with Flash Builder. 18
Setting the FLEX_HOME Environment Variable 21
Building the Scribble Client Application with Apache Ant. 22
Configuring the Scribble Project Run/Debug Settings in Adobe Flash Builder 23
Running Scribble in Adobe Flash Builder. 26
Running Scribble using the TestPage.html File 27
Accessing the Diagnostics Panel 29
Sharing the Scribble Client Application 34
Debugging the ScribbleApp Client Application in Flash Builder 35
Debugging the ScribbleApp Service Application 37

Preface

Welcome to the *PureWeb STK 3.1 Quick Start Guide: Flex*. Although every effort has been made to make this document clear and easy to understand, it is still very technical in nature.

Wherever possible, cross-referencing among chapters and other documents within the PureWeb® Software Transformation Kit (STK) documentation suite has been used.

This document is valid for all 3.x release(s) of this product.

This preface contains the following sections:

- Intended Audience, page 6
- Reading Recommendations, page 6
- Making Comments on This Document, page 7
- Contacting Calgary Scientific Support, page 7
- Common Development Tasks, page 7

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 38.

Intended Audience

This document is primarily intended for software developers who plan to install and use the PureWeb® STK. It has been written with the assumption that you have:

- A working knowledge of Flash Builder.
- An intermediate knowledge of Flex.

Reading Recommendations

This preface includes task-based tables that contain all of the tasks and procedures needed to successfully build, run, and debug the PureWeb® sample applications. See Task Summary: Development Tasks and Task Summary: Scribble Application Tasks.

Making Comments on This Document

If you especially like or dislike anything about this document, feel free to e-mail your comments to techpubs@calgaryscientific.com.

You can comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this document. Please limit your comments to the scope of this document only and to the way in which the information is presented.

When you send us comments, you grant Calgary Scientific Inc. a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

Contacting Calgary Scientific Support

Contact Calgary Scientific Support during the hours of 9:00AM - 5:00PM, Monday - Friday (Mountain Standard Time).

Web Site	E-Mail
http://support.getpureweb.com	support@getpureweb.com

Common Development Tasks

Common development tasks are listed in the table below.

Task Summary: Development Tasks

Task	Procedure
Familiarize yourself with PureWeb®.	“About PureWeb® STK” on page 9. “Basic Architecture” on page 10.
Ensure that you have the required development prerequisites.	“Prerequisites For Flex Development” on page 12.

Common Scribble Application tasks are listed in the table below.

Task Summary: Scribble Application Tasks

Task	Procedure
Familiarize yourself with the ScribbleApp classes and code.	“About Scribble” on page 13.
Build ScribbleApp.	“Build Scribble” on page 17.
Run ScribbleApp.	“Run Scribble” on page 23.
Exercise the ScribbleApp functionality.	“Sample Functionality” on page 29.
You may find it helpful to debug ScribbleApp client application.	“Debugging the ScribbleApp Client Application in Flash Builder” on page 35.
You may find it helpful to debug ScribbleApp service application.	“Debugging the ScribbleApp Service Application” on page 37.

Chapter

1 Introducing PureWeb® STK

This chapter contains detailed information about the PureWeb® Software Transformation Kit (STK) solution and its architecture. It also contains information about prerequisites needed for Flex development using the PureWeb® STK.

This chapter contains the following sections:

- About PureWeb® STK, page 9
- Basic Architecture, page 10
- Prerequisites For Flex Development, page 12
- About the Sample Application, page 12

About PureWeb® STK

PureWeb® is a platform that enables applications to be centrally hosted and delivered to the end user, from workstations to hand-held devices, through standard web technologies.

PureWeb® allows you to migrate your existing desktop applications to the web without requiring a costly rewrite of your existing applications.

The PureWeb® application framework is built specifically to leverage HTTP and XML for the highest level of flexibility, optimization, consistency and performance. Its architecture is designed to use web standard technologies to deliver your applications through all internet browsers capable of hosting Microsoft Silverlight, as well as in many smart phones and tablet offerings, using the PureWeb® API.

PureWeb® integrates directly into your existing C#, C++, Flash, or Java code and allows you to maintain one code-base from which you can deliver desktop-based applications as well as browser and mobile-based applications such as those that run on Apple iPad, Apple iPhone or Google Android devices.

Applications that leverage PureWeb® provide access in a highly secure and compliant manner, so mobile users can connect to your applications and interact with them as if they were stored locally on their device without any of the application data ever persisting on the device itself.

This is achieved by:

- Reimplementing the existing software user interface (UI) through a web interface.
- Modifying the application to act as a service for the remote interface.

The PureWeb® STK includes both sample applications and documentation to help you with your development.

For more information about the product, visit http://www.getpureweb.com.

Basic Architecture

The PureWeb® platform enables applications to run through a standardized web interface.

PureWeb® solutions are typically composed of three tiers:

- PureWeb® Service—contains most of the application logic. It uses the Service API to *plug in* to the PureWeb® Server, maintain application state, and generate rendered views.
- PureWeb® Server—is responsible for starting/stopping PureWeb® Services, and mediating communication between PureWeb® Clients and PureWeb® Services.
- PureWeb® Client—allows users to interact with PureWeb® Services through web browsers and/or mobile devices.

To PureWeb® enable an existing application means transforming the workstation version of an application into a PureWeb® Service by creating objects and calling methods from the Service API.

Specifically, the application will create an instance of the StateManager class (responsible for maintaining application state in a PureWeb® Service) and pass it to an instance of StateManagerServer.

The StateManagerServer handles communication with the PureWeb® Server over a TCP socket. Messages arriving from the client via the Server as XML text are converted to lists of command objects that are executed by the StateManager. These include commands to change the application state, commands that represent user input events, and custom commands understood by the application.

In executing the commands, the application will generate response objects including changes to the application state and new rendered images. The StateManagerServer converts the response objects to XML and multipart messages that are passed back to the client via the PureWeb® Server.

Once the service creates the StateManager instance, it will register event handlers that are invoked when specific nodes in the application state change.

Every rendered view that is intended to be displayed on the client must be registered with the StateManager. In order to communicate with the StateManager, views must inherit from the IRenderedView interface. Use the adapter pattern (http://en.wikipedia.org/wiki/Adapter_pattern), if it is not possible or convenient for view classes to inherit directly from the IRenderedView interface.

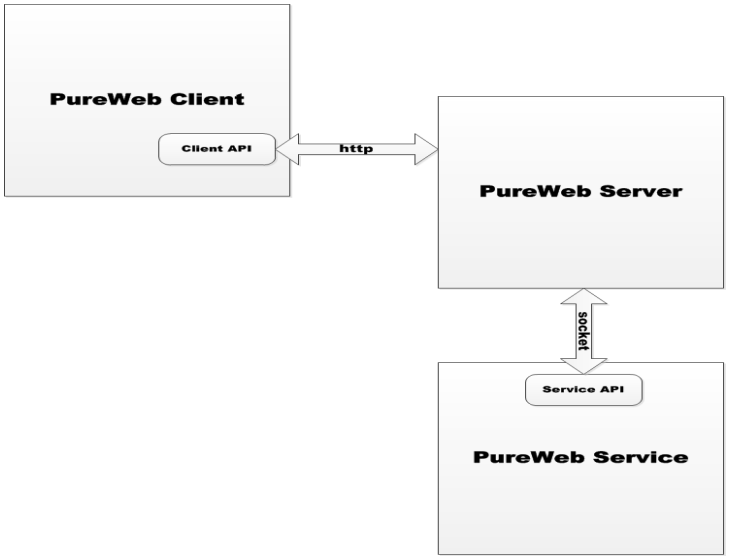

Figure 1: Typical PureWeb® Solution

As shown in Figures 2, the application state is stored on both the client and server sides as XML. To maintain synchronicity, the differences in the state are transmitted to and from both the client and the service. The client sends commands and input events to the service (in XML form). The service sends updated images to the client.

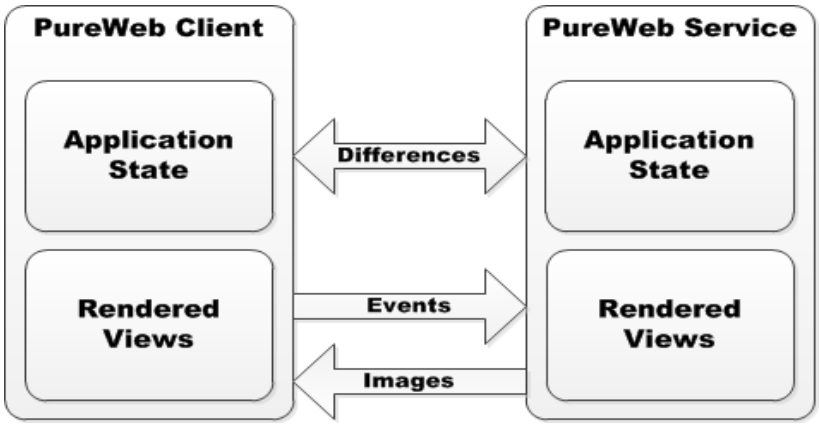

Figure 2: State Views

Prerequisites For Flex Development

The following is required for Flex development:

- Refer to the *PureWeb® Installation Guide for Windows* for instructions on how to install the PureWeb® STK and the list of prerequisites for Flex development.

About the Sample Application

Although only a 64-bit platform is supported for application development, you can target the applications developed using the PureWeb® STK to deploy on either a 32-bit or a 64-bit platform.

The following sample applications are included to illustrate the fundamentals of using the PureWeb® STK:

- Scribble—Exercises the functionality of drawing lines, changing line color, and erasing lines.

Warning! Cookie sharing between browser tabs may cause issues with the functionality of these sample applications. We recommend that you use two separate browsers (Microsoft Internet Explorer and Mozilla Firefox) to open two sessions of the same application.

Chapter

2 Introducing Scribble

This section provides detailed information about the Flex Scribble sample application. It includes information about the classes and methods used in the sample, as well as, information about building, running, and debugging the sample application.

This chapter contains the following sections:

- About Scribble, page 13
- Build Scribble, page 17
- Run Scribble, page 23
- Sample Functionality, page 29
- Debug ScribbleApp, page 35

About Scribble

Read the section titled “Basic Architecture” on page 10 to gain an understanding of how the client, service and server components of a PureWeb®-enabled application are connected, the kinds of data that are exchanged between client, service and server, the role of the application state on the client, service and server-sides, and how that state is synchronized.

The sample application consists of two components:

- ScribbleApp—A PureWeb®-enabled service application developed in C# that implements the scribble application logic.
- ScribbleClient—A PureWeb®-enabled client application that presents a remote interface to the service application.

The source code for:

- ScribbleApp is located at C:\CSI\PureWeb\SDK\Samples\Scribble\ScribbleAppCsharp.
- ScribbleClient is located at C:\CSI\PureWeb\SDK\Samples\Scribble\ScribbleClientFlex.

ScribbleApp Code Description

To inspect the ScribbleApp code, double-click on the ScribbleApp2010.sln file located at C:\CSI\PureWeb\SDK\Samples\Scribble\ScribbleAppCsharp.

For detailed information about the ScribbleApp code refer to the *Introducing ScribbleApp* chapter of the *PureWeb® Quick Start Guide: C#.*

ScribbleClient Code Description

The ScribbleClient.mxml file contains the code that creates the ScribbleView. ScribbleView is the main application class for this client.

The ScribbleView includes:

- a Diagnostics View — displays the diagnostics information.
- a Share button — allows you to share the session.
- an Erase All button — erases all of your scribbles.
- a Color input field — allows you to input your preferred color.

The code contained in the ScribbleClient.mxml file is shown below.

```
<mx:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
    xmlns:s="library://ns.adobe.com/flex/spark"
    xmlns:app="pureweb.app.*"
    xmlns:scribble="scribble.*"
    xmlns:mx="library://ns.adobe.com/flex/mx"
    xmlns:ui="pureweb.client.ui.*"
    width="100%" height="100%" clipContent="false" horizontalScrollPolicy="off"
    layout="absolute" preloader="mx.preloaders.DownloadProgressBar" verticalGap="0"
    verticalScrollPolicy="off" xmlns:diagnostics="pureweb.client.diagnostics.*"
    applicationComplete="viewModel.connect()">
    <fx:Declarations>
        <scribble:ScribbleClientViewModel id="viewModel" />
    </fx:Declarations>

    <mx:HBox width="100%" height="100%">
        <diagnostics:DiagnosticsView height="100%" />
        <mx:VBox width="100%" height="100%">

    //The color input field and buttons are created.
    <mx:HBox width="100%" height="100%">
        <diagnostics:DiagnosticsView height="100%" />
        <mx:VBox width="100%" height="100%">
            <mx:HBox width="100%">
                <mx:Label text="Color:" />
                <mx:TextInput id="colorTxtInp" width="200"
                    enter="viewModel.updateColor(colorTxtInp.text)"focusOut=
                    "viewModel.updateColor(colorTxtInp.text)" text="@{viewModel.color}" />
                <mx:Button id="eraseAllBn" label="Erase all" click="viewModel.clear()" />
            </mx:HBox>
```

```
            <ui:View id="view" width="100%"
                height="100%" horizontalCenter="0" viewName="ScribbleView"/>

            <mx:HBox width="100%">
                <mx:Button id="shareBn" label="Share" click="viewModel.createShare()" />
                <mx:Label text="Share Url" />
                <mx:TextInput id="textOutput" width="100%" text="{viewModel.shareUrl}"/>
            </mx:HBox>
        </mx:VBox>
    </mx:HBox>
</mx:Application>
```

The ScribbleClientViewModel.as file contains the ScribbleClientViewModel class. It contains the functionality that allows the client to interact with the PureWeb Service including functions for getting and setting the framework instance.

```
public function get framework():Framework
{
    if(_framework == null) _framework = Framework.instance;
        return _framework;
    }

    public function set framework(value:Framework):void
    {
        _framework = value;
}
```

The connect function is called to connect you to the server.

```
public function connect():void
{
    var traceTarget:TraceTarget = new TraceTarget();
    traceTarget.includeCategory = true;
    traceTarget.includeLevel = true;
    Log.addTarget(traceTarget);

    framework.state.stateManager.addValueChangedHandler("/ScribbleColor", onScribbleColorChanged);

    var href:String = ExternalInterface.call("eval", "document.location.href");
    framework.client.connect(href);
}
```

When you input a color in the Color input field, the onScribbleColorChanged function is called. The current color is stored in application state in the /ScribbleColor value. When this value is set, it is relayed to the service application and the Service STK triggers the application to change the color and refresh the display. That updated display is then propagated back to the client.

```
public function onScribbleColorChanged(e:XmlChangeEvent):void
{
    color = framework.state.stateManager.getValue("/ScribbleColor");
}

public function updateColor(color:String):void
{
    framework.state.stateManager.setValue("/ScribbleColor", color);
}
```

When you click on the Erase All button, a Clear command is sent to the service application. The service application responds by clearing previously drawn scribbles, and sends an updated blank image back to the client.

```
public function clear():void
{
    framework.client.queueCommand("Clear");
}
```

When you click on the Share button, the client creates an AppShare service request and queues it for execution by the PureWeb® Server. The AppShare service request creates a collaboration session for the application. The URL that is returned from the request can be shared with others to allow them to interact with the same application session. It also creates an event listener to monitor the share request.

```
public function createShare():void
{
    var appShare:AppShare = new AppShare("Scientific", "Collable", 600000);
    appShare.addEventListener(Event.COMPLETE, onShareRequestComplete);
    Framework.instance.client.queueRequest(appShare);
}

public function onShareRequestComplete(event:Event):void
{
    var appShare:AppShare = AppShare(event.target);
    shareUrl = appShare.shareUrl;
}
```

Build Scribble

Follow the steps in the procedures below to build the sample application.

Building Scribble Service Application

Purpose: To build the Scribble service application in Microsoft Visual Studio 2010.

Start of procedure

1. Follow the steps in the Building ScribbleApp procedure in the *PureWeb® STK Quick Start Guide: C#*.
2. Reload the plugin by following the steps in the Reloading a Plugin procedure in the *PureWeb® Server Administration Guide*.

End of procedure

Next Steps

- Follow the steps in the "Building the Scribble Client Application with Flash Builder" procedure or the "Building the Scribble Client Application with Apache Ant" procedure.

Building the Scribble Client Application with Flash Builder

Purpose: To build and deploy the Scribble client application in Adobe Flash Builder.

Prerequisites

- Ensure that you have all the required software prerequisites installed to build and run the samples. Refer to the *PureWeb® Installation Guide for Microsoft Windows*.
- Build the Scribble service application by following the steps in the "Building Scribble Service Application" procedure.

Start of procedure

1. Open Adobe Flash Builder.
2. Select Windows | Preferences from the main menu.

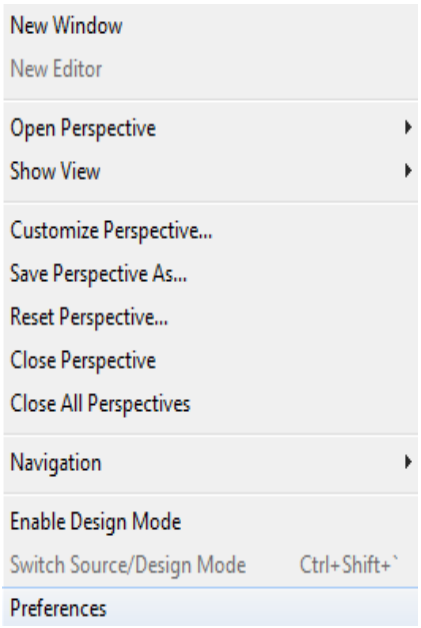

Figure 3: Navigating to Preferences

3. Select General | Workspace | Linked Resources from the Preferences menu.
4. Click New to display the New Variable dialog.

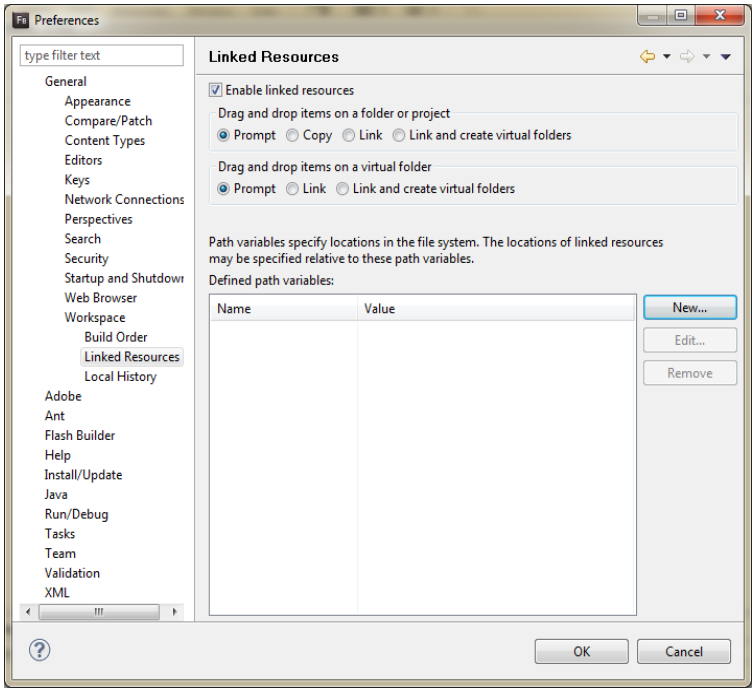

Figure 4: Linked Resources Dialog

5. Enter PUREWEB_HOME for the variable Name.
6. Click Folder... and browse to the Location of the PureWeb® Server directory. If you installed PureWeb® in the default location, the Server directory is located at C:\CSI\PureWeb\Server.
7. Click OK to save your changes.

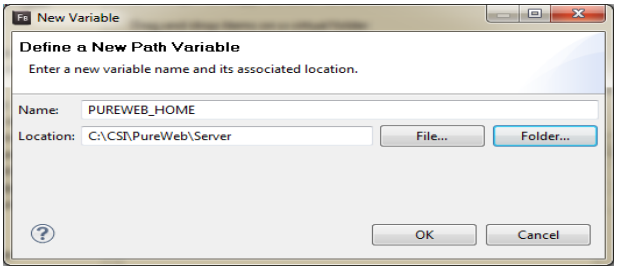

Figure 5: PUREWEB_HOME Variable

8. Click New to display the New Variable dialog again.
9. Enter PUREWEB_LIBS for the variable Name.
10. Click Folder... and browse to the Location of the PureWeb® Libs directory. If you installed PureWeb® in the default location, the Libs directory is located at C:\CSI\PureWeb\SDK\Redistributable\Libs.
11. Click OK to save your changes.

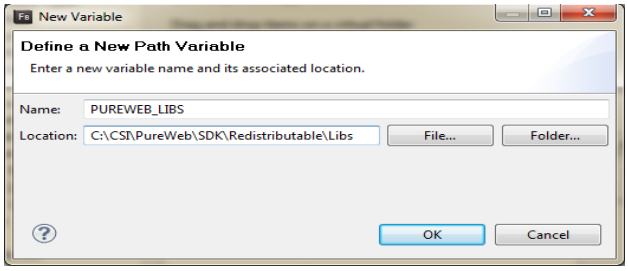

Figure 6: PUREWEB_LIBS Variable

12. Click OK to close the Preferences dialog.
13. Select File | Import Flash Builder Project... from the main menu.
14. Choose to import a Project folder.
15. Browse to C:\CSI\PureWeb\SDK\Samples\Scribble\ScribbleClientFlex.
16. Click Finish to import the project. Once imported the project should build automatically.

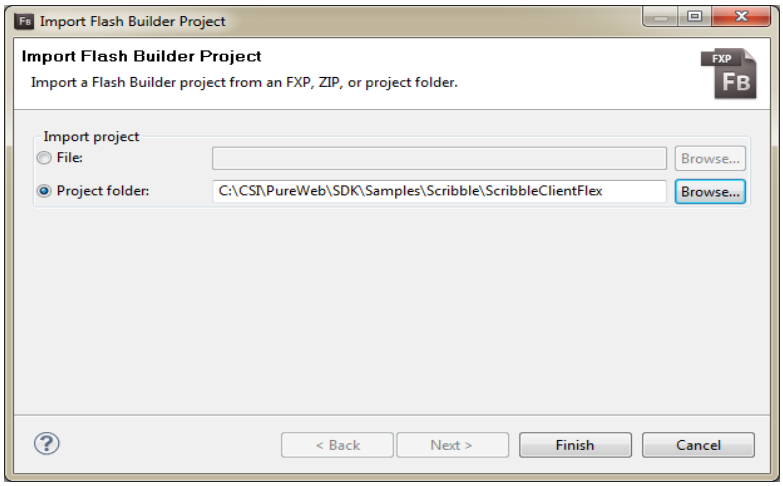

Figure 7: Import the Project

End of procedure

Next Steps

- Follow the steps in the "Configuring the Scribble Project Run/Debug Settings in Adobe Flash Builder" procedure.

Setting the FLEX_HOME Environment Variable

Purpose: To set the FLEX_HOME environment variable to point to your version of Flex.

Start of procedure

1. Select Control Panel | System | Advanced system settings.
2. Click the Environment Variables button.
3. Select New in the System Variable section.
4. Enter FLEX_HOME in the Variable name field.
5. Enter the path to your version of Flex in the Variable value field.
6. Click the OK button to save your changes.

End of procedure

Building the Scribble Client Application with Apache Ant

Purpose: To build and deploy the Scribble client application with Apache Ant.

Prerequisites

- Ensure that you have all the required software prerequisites installed to build and run the samples. Refer to the *PureWeb® Installation Guide for Microsoft Windows*.
- Build the Scribble service application by following the steps in the "Building Scribble Service Application" procedure.
- You must have installed Apache Ant by following the steps in the Installing Apache Ant procedure found in the *PureWeb® STK Installation Guide for Microsoft Windows*.
- You must set the ANT_HOME variable by following the steps in the Setting the ANT_HOME Variable procedure found in the *PureWeb® STK Installation Guide for Microsoft Windows*.
- You must set the FLEX_HOME variable by following the steps in the "Setting the FLEX_HOME Environment Variable" procedure.

Start of procedure

1. Open a console window.
2. Change directories to C:\CSI\PureWeb\SDK\Samples\Scribble\ScribbleClientFlex.
3. Type ant to run the build.xml build script.

End of procedure

Next Steps

- Follow the steps in the "Running Scribble using the TestPage.html File" procedure.

Run Scribble

Use the procedures below to run the Scribble sample application.

Configuring the Scribble Project Run/Debug Settings in Adobe Flash Builder

Purpose: To configure the Scribble project run/debug setting in Adobe Flash Builder.

Prerequisites

- Complete the steps in the "Building the Scribble Client Application with Flash Builder" procedure.

Start of procedure

1. Open Adobe Flash Builder.
2. Right-click the ScribbleClientFlex project and select Properties from the menu.

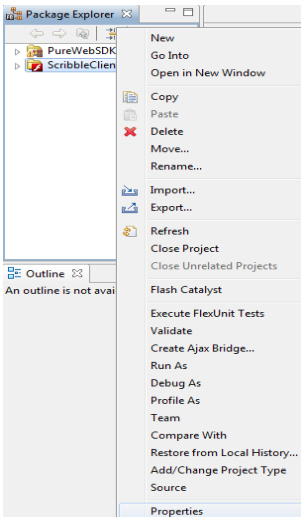

Figure 8: Navigate to Project Properties

3. Select Run/Debug Settings from the Project Properties menu.
4. Click New to open the Select Configuration Type dialog.

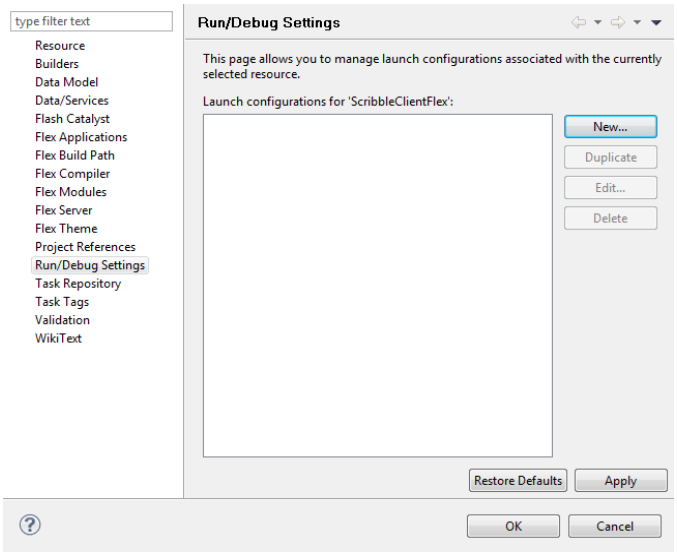

Figure 9: ScribbleClientFlex Properties

5. Select Web Application and click OK to Edit the Lunch Configuration Properties.

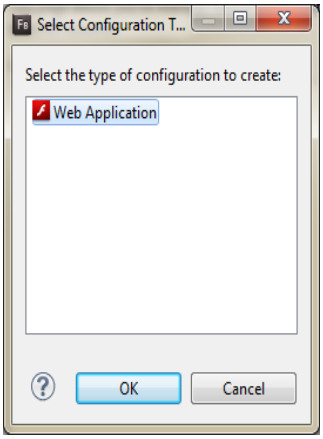

Figure 10: Select Configuration Type

6. Uncheck the Use default check box in the URL or path to launch section.
7. Browse to C:\CSI\PureWeb\SDK\Samples\Scribble\ScribbleClientFlex\TestPage.html.
8. Click OK.

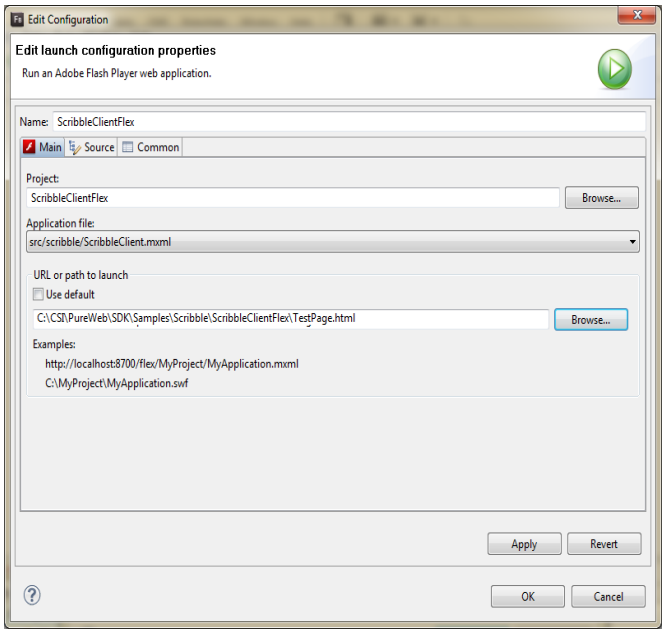

Figure 11: Launch Configuration

9. Click OK to close the Run/Debug Settings dialog.

End of procedure

Next Steps

- To run ScribbleApp, follow the steps in the "Running Scribble in Adobe Flash Builder" procedure.
- To debug ScribbleApp, follow the steps in the "Debugging the ScribbleApp Client Application in Flash Builder".

Running Scribble in Adobe Flash Builder

Purpose: To run Scribble in Adobe Flash Builder.

Prerequisites

- Complete the steps in the "Building Scribble Service Application" procedure.
- Complete the steps in the "Building the Scribble Client Application with Flash Builder" procedure.
- Complete the steps in the "Configuring the Scribble Project Run/Debug Settings in Adobe Flash Builder" procedure.
- Start the PureWeb® Server and follow the steps in the Reloading a Plugin procedure in the *PureWeb® Server Administration Guide*.

Start of procedure

1. Open Adobe Flash Builder.
2. Select the ScribbleClientFlex project in the Package Explorer and select Run | Run from the main menu.
3. Click on the Launch Scribble Demo.Net with diagnostics link.

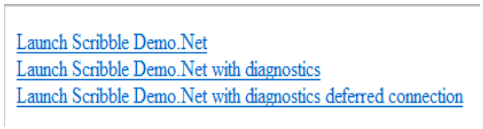

Figure 12: Launch Scribble

4. Enter admin/admin into the name and password fields.

Figure 13: PureWeb Login

5. Click the Sign In button to launch the application.

Figure 14: Scribble Initial View

End of procedure

Next Steps

- Examine the "Sample Functionality".

Running Scribble using the TestPage.html File

Purpose: To run Scribble using the TestPage.html File.

Prerequisites

- Complete the steps in the "Building Scribble Service Application" procedure.
- Complete the steps in the "Setting the FLEX_HOME Environment Variable" procedure.
- Complete the steps in the "Building the Scribble Client Application with Apache Ant" procedure.
- Start the PureWeb® Server and follow the steps in the Reloading a Plugin procedure in the *PureWeb® Server Administration Guide.*

Start of procedure

1. Navigate to C:\CSI\PureWeb\SDK\Samples\Scribble\ScribbleClientFlex.
2. Double-click on the TestPage.html file.

3. Click on the Launch Scribble Demo.Net with diagnostics link.

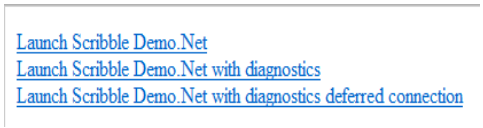

Figure 15: Launch Scribble

4. Enter admin/admin into the name and password fields.

Figure 16: PureWeb Login

5. Click the Sign In button to launch the application.

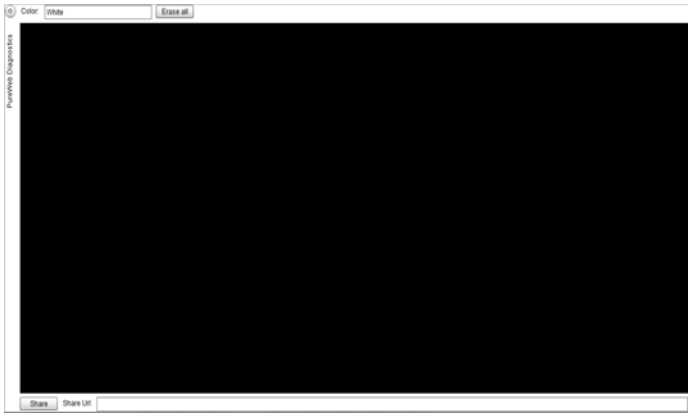

Figure 17: Scribble Initial View

End of procedure

Next Steps

- Examine the “Sample Functionality”.

Sample Functionality

The Scribble client application supports the following functionality:

- Draw lines on the canvas by left-clicking the mouse and dragging. Even though it appears that the ScribbleApp client is doing all the drawing, it is actually the service application that is doing the drawing.
- Change colors by entering the name of a new color in the Color text field.
- Clear the canvas by clicking on the Erase All button.
- Examine diagnostics using the panel provided.
- Share the application with other users by clicking the Share button.

Figure 18: Scribble Flex Functionality

Accessing the Diagnostics Panel

Purpose: To show or hide the diagnostics panel.

Start of procedure

1. Click on the expand/contract button.

Figure 19: Accessing the Diagnostics Panel

The diagnostics panel provides the following functionality:

- Set various options in the Options tab.
  - Client side filtering: When enabled, this filters the client-side commands to reduce the bandwidth overhead.
  - Interactive quality: When SetViewInteracting is enabled, lowering the interactive quality reduces the bandwidth usage by lowering the image quality.
  - Full quality: When SetViewInteracting is enabled, this controls bandwidth and interactive quality.

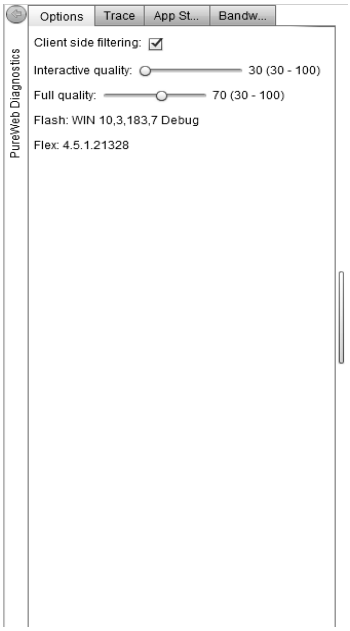

Figure 20: Options Tab

- Examine the messages going to and from the server in the Trace tab.
  - Displays trace messages from the application. The controls allow you to set the behavior of the trace view, including the buffer length, scrolling, and clearing of the buffer.

Note: To write information to the trace log and IDE output window, use mx.logging.ILogger interface methods.

Figure 21: Trace Tab

- Examine the application state in the App State tab.
  - Displays an XML representation of the application's StateManager.
  - Click Refresh to update the view with the latest state.

Figure 22: App State Tab

- Perform bandwidth and latency tests in the Bandwidth tab.
  - Test Latency: For the specified number of iterations, the system latency is measured by sending a payload of 1 byte.
  - Test Bandwidth: For the specified number of iterations, the system bandwidth is measured by sending a payload of the specified size between the client and server.

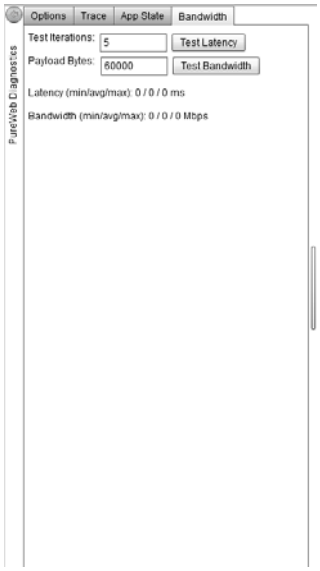

Figure 23: Bandwidth Tab

End of procedure

Sharing the Scribble Client Application

Purpose: To share the Scribble client application.

Start of procedure

Warning! By default the Scribble client application connects using the hostname of the local host. If you are trying to share the application from another computer, you must modify the shared URL, http://localhost:8080/pureweb/app?name=ScribbleApp&client=flex, by replacing localhost with the IP address.

First person:

1. Click the Share button located in the lower-left corner of the Scribble client application.
2. Notify a second person of the URL displayed in the Session URL field displayed as the bottom of the screen.

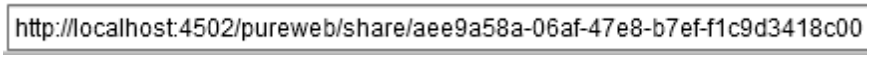

Figure 24: Share URL

Second person:

1. Open a browser and navigate to the shared URL to display the PureWeb Collaboration Login page.
2. Select your Preferred Client from the drop down box, enter Scientific as your Password and click the Sign In button to join the session.

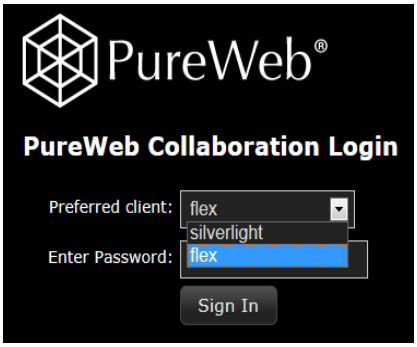

Figure 25: Collaboration Login Page

End of procedure

Debug ScribbleApp

The information in this section explains the procedures for debugging the client and the service applications.

Note: Google Chrome includes a plug-in of Flash Player that *can not* be used to debug Flex applications. For instructions on how to debug Flex application in Google Chrome see, http://cookbooks.adobe.com/post_Debug_Flex_Applications_using_Google_Chrome-17271.html.

Debugging the ScribbleApp Client Application in Flash Builder

Purpose: To debug the ScribbleApp client application in Adobe Flash Builder.

Prerequisites

- Complete the steps in the "Building Scribble Service Application" procedure.
- Complete the steps in the "Building the Scribble Client Application with Flash Builder" procedure.
- Complete the steps in the "Configuring the Scribble Project Run/Debug Settings in Adobe Flash Builder" procedure.
- Start the PureWeb® Server and follow the steps in the Reloading a Plugin procedure in the *PureWeb® Server Administration Guide*.

Start of procedure

1. Open Adobe Flash Builder.
2. Set a breakpoint in the eraseAllBn_click method in the ScribbleClientImpl.as file.
3. Select the ScribbleClientFlex project in the Package Explorer and select Run | Debug from the main menu.
4. Click on the Launch Scribble Demo.Net with diagnostics link.

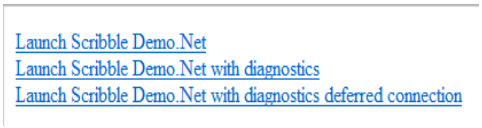

Figure 26: Launch Scribble

5. Enter admin/admin into the name and password fields.

Figure 27: PureWeb Login

6. Click the Sign In button to launch the application.

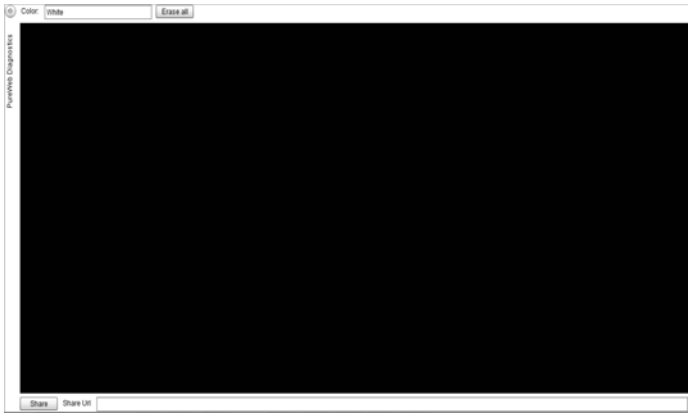

Figure 28: Scribble Initial View

7. Draw lines on the canvas by left-clicking the mouse and dragging.

8. Click the Erase All button to trigger the breakpoint.

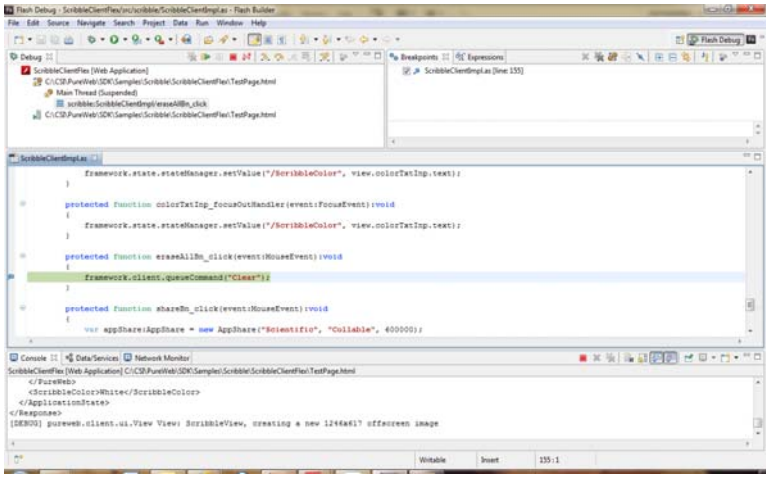

Figure 29: Trigger the Breakpoint

End of procedure

Debugging the ScribbleApp Service Application

Purpose: To debug the ScribbleApp service application in Microsoft Visual Studio 2010.

Start of procedure

1. Follow the steps in the Debugging the ScribbleApp Service Application procedure in the *PureWeb® STK Quick Start Guide: C#.*

End of procedure

Supplements

Related Documentation Resources

The following resources provide additional information that is relevant to this software. Consult these additional resources as necessary.

PureWeb® STK

- *PureWeb® STK Installation Guide for Microsoft Windows*, which provides detailed information on installing PureWeb® STK on a Microsoft Windows operating system.
- *PureWeb® Server Administration Guide*, which provides information about the PureWeb® Server.
- *PureWeb® Flex Client STK API Reference*, which describes the Flex Client STK.
- *PureWeb® Troubleshooting Guide*, which describes solutions to common issue.
- *PureWeb® STK Release Notes.*
- *PureWeb® STK Application Upgrading Notes.*

Document Conventions

This document uses certain stylistic and typographical conventions—introduced here—that serve as shorthands for particular kinds of information.

Document Version Number

A version number appears at the bottom of the inside front cover of this document. Version numbers change as new information is added to this document. Here is a sample version number:

PW3.0_Java_QSG_03-2011_v3.0.001.00

You will need this number when you are talking with Calgary Scientific Support about this product.

Screen Captures Used in This Document

Screen captures from the product graphical user interface (GUI), as used in this document, may sometimes contain minor spelling, capitalization, or grammatical errors. The text accompanying and explaining the screen captures corrects such errors *except* when such a correction would prevent you from installing, configuring, or successfully using the product. For example, if the name of an option contains a usage error, the name would be presented exactly as it appears in the product GUI; the error would not be corrected in any accompanying text.

Type Styles

Table 1 describes and illustrates the type conventions that are used in this document.

Table 1: Type Styles

Type Style	Used For	Examples
Italic	• Document titles • Emphasis • Definitions of (or first references to) unfamiliar terms • Mathematical variables Also used to indicate placeholder text within code samples or commands, in the special case where angle brackets are a required part of the syntax (see the note about angle brackets on page 40).	Please consult the *Calgary Scientific Migration Guide* for more information. Do *not* use this value for this option. A *customary and usual* practice is one that is widely accepted and used within a particular industry or profession. The formula, $x + 1 = 7$ where x stands for . . .

Table 1: Type Styles (Continued)

Type Style	Used For	Examples
Monospace font (Looks like teletype or typewriter text)	All programming identifiers and GUI elements. This convention includes: • The *names* of directories, files, folders, configuration objects, paths, scripts, dialog boxes, options, fields, text and list boxes, operational modes, all buttons (including radio buttons), check boxes, commands, tabs, CTI events, and error messages. • The values of options. • Logical arguments and command syntax. • Code samples. Also used for any text that users must manually enter during a configuration or installation procedure, or on a command line.	Select the Show variables on screen check box. In the Operand text box, enter your formula. Click OK to exit the Properties dialog box. Scribble service distributes the error messages in EventError events. If you select true for the inbound-bsns-calls option, all established inbound calls on a local agent are considered business calls. Enter exit on the command line.
Square brackets ([ ])	A particular parameter or value that is optional within a logical argument, a command, or some programming syntax. That is, the presence of the parameter or value is not required to resolve the argument, command, or block of code. The user decides whether to include this optional information.	smcp_server -host [/flags]
Angle brackets (< >)	A placeholder for a value that the user must specify. This might be a DN or a port number specific to your enterprise. **Note:** In some cases, angle brackets are required characters in code syntax (for example, in XML schemas). In these cases, italic text is used for placeholder values.	smcp_server -host <confighost>

Index

Symbols

[] (square brackets) . . . 40
< > (angle brackets) . . . 40

A

angle brackets . . . 40
architecture . . . 10, 13
audience, for document . . . 6

B

brackets
  angle . . . 40
  square . . . 40

C

class
  ScribbleClientViewModel . . . 15
  ScribbleView . . . 14
code description
  ScribbleApp . . . 14
  ScribbleClient . . . 14
commenting on this document . . . 7
conventions
  in document . . . 39
  type styles . . . 39

D

document
  audience . . . 6
  conventions . . . 39
  errors, commenting on . . . 7
  reading recommendations . . . 6
  version number . . . 39

E

environment variable
  ANT_HOME . . . 22
  FLEX_HOME . . . 21, 22

F

font styles
  italic . . . 39
  monospace . . . 40
functions
  clear . . . 16
  connect . . . 15
  createShare . . . 16
  get framework . . . 15
  onScribbleColorChanged . . . 16
  onShareRequestComplete . . . 16
  set framework . . . 15
  updateColor . . . 16

I

intended audience . . . 6
italics . . . 39

M

methods
  SetViewInteracting . . . 30
monospace font . . . 40

P

prerequisites . . . 12
pureweb
  client . . . 10
  security . . . 9–10
  server . . . 10

service 10
solution 10
web technologies 9

R

reading recommendations 6

S

sample applications
scribble 12
scribble sample code location
ScribbleApp 13
ScribbleClient 13
scribble sample components
ScribbleApp 13
ScribbleClient 13
square brackets 40
state view 11
StateManager 32
support
contacting 7

T

tasks
development tasks 7
scribble application tasks 8
type styles
conventions 39
italic 39
monospace 40
typographical styles 39

V

version numbering, document 39