PureWeb® STK 3.1

Quick Start Guide: C++

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

ResolutionMD, PureWeb, the Calgary Scientific logo, and the PureWeb logo are trademarks and/or registered trademarks of Calgary Scientific Inc. or its subsidiaries. All other trademarks and trade names referred to in this document are the property of other companies.

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_CPlusPlus_QSG_06-2012_v3.100.00

Table of Contents

List of Tasks

List of Procedures

Preface

Welcome to the *PureWeb STK 3.1 Quick Start Guide: C++*. 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:

For information about related resources and about the conventions that are used in this document, see the supplementary material starting on page 30.

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 Microsoft Visual Studio.
- An intermediate knowledge of C++.
- A basic knowledge of Microsoft Silverlight.

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 C++ 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 ScribbleApp” on page 13.
Build ScribbleApp.	“Building ScribbleApp” on page 19.
Run the ScribbleApp.	“Running ScribbleApp” on page 20.
Exercise the ScribbleApp functionality.	“Sample Functionality” on page 21.
You may find it helpful to debug the ScribbleApp Client application.	“Debugging the ScribbleApp Client Application” on page 26.
You may find it helpful to debug the ScribbleApp Service application.	“Debugging the ScribbleApp Service Application” on page 28.

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 C++ development using the PureWeb® STK.

This chapter contains the following sections:

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.

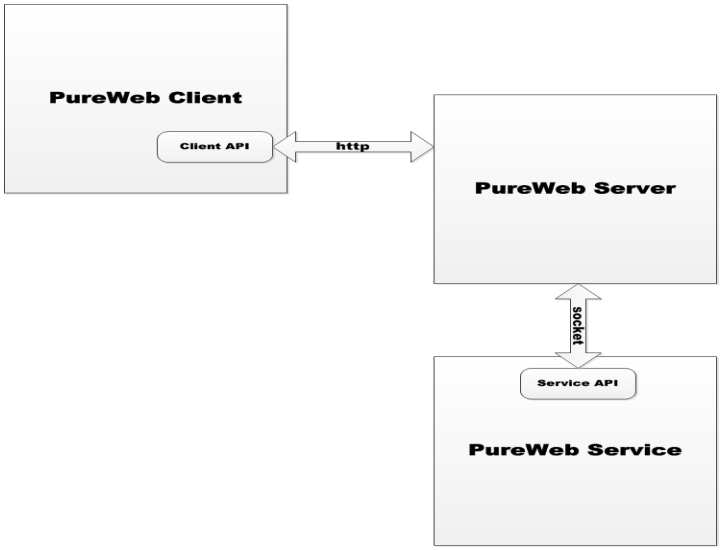

Figure 1: Typical PureWeb® Solution

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.

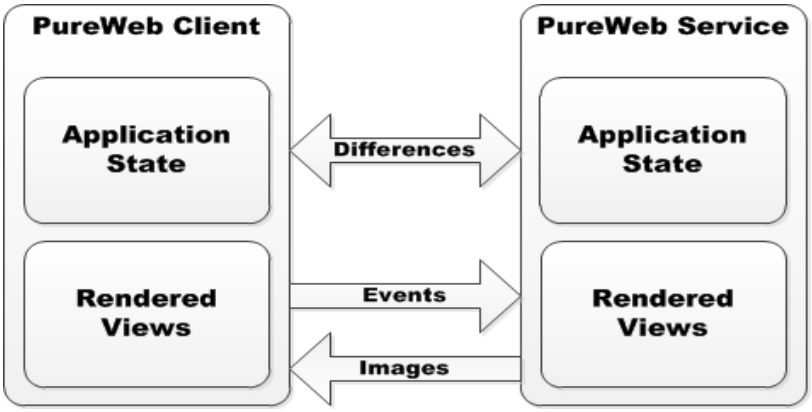

Figure 2: State Views

Prerequisites for C++ Development

The following is required for C++ development:

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

About the Sample Applications

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:

- ScribbleApp—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 ScribbleApp

This section provides detailed information about the C++ ScribbleApp 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 ScribbleApp

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.

Note: The ScribbleAppCpp directory contains both a ScribbleApp2008.sln and ScribbleApp2010.sln file. Only the Microsoft Visual Studio 2010 version of the sample is documented in this Quick Start Guide. Implement the appropriate solution for your the version of Microsoft Visual Studio that you have installed.

The ScribbleApp2008.sln solution does *not* contain a client application.

The sample application consists of two components:

- ScribbleAppCpp—A PureWeb®-enabled service application that implements the scribble application logic.
- ScribbleClient—A PureWeb®-enabled client application that presents a remote interface to the service application.

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

ScribbleApp Code Description

The ScribbleView.cpp file contains the code that implements the CScribbleView class. The CScribbleView class contains the business logic of the ScribbleApp Service application, providing the ability to draw lines, respond to color changes, and erase lines.

There are a number of methods and inner classes of the CScribbleView class that are noteworthy from a PureWeb®-enablement perspective.

The CScribbleView constructor performs a number of important actions including:

- Registering the ScribbleView instance with the state manager.
- Adding an application state property change listener to listen for color changes.
- Adding a command handler to listen for erase commands from the client.

```
CScribbleView::CScribbleView()
{
    m_startpt = -1;
    m_endpt = -1;
    m_cColor = COLORREF(RGB(0,0,0));
    CScribbleApp::StateManager().ViewManager().RegisterView("ScribbleView", this);

    ViewImageFormat viewImageFormat;
    viewImageFormat.PixelFormat = PixelFormat::Bgr24;
    viewImageFormat.ScanLineOrder = ScanLineOrder::TopDown;
    viewImageFormat.Alignment = 4;
    CScribbleApp::StateManager().ViewManager().SetViewImageFormat("ScribbleView",viewImageFormat);

    CScribbleApp::StateManager().CommandManager().
        AddUiHandler("Clear", Bind(this, &CScribbleView::OnExecuteClear));
    CScribbleApp::StateManager().XmlStateManager().
        AddValueChangedHandler("ScribbleColor", Bind(this, &CScribbleView::OnScribbleColorChanged));
    m_hdcOffscreen = CreateCompatibleDC(NULL);
    m_dcOffscreen.Attach(m_hdcOffscreen);
}
```

The ~CScribbleView destructor unregisters the ScribbleView instance:

```
CScribbleView::~CScribbleView()
{
    CScribbleApp::StateManager().ViewManager().UnregisterView("ScribbleView");
}
```

The OnScribbleColorChanged method responds to changes to the ScribbleColor application state property. Whenever the client application changes ScribbleColor, a state difference is generated and transmitted to the service application. The StateManager merges the difference into the application state on the server-side, and notifies any registered event handlers of the change:

```
void CScribbleView::OnScribbleColorChanged(ValueChangedEventArgs args)
{
    String color = args.NewValue().As<String>().ToLower(Locale::Invariant());
    NamedColors c = black;
    if (!(c.TryParse(color, c)))
    {
        TypelessStateLock lock(CScribbleApp::StateManager().LockAppState());
        lock.Element("ScribbleColor").SetValue("black");
    }
    m_cColor = c;
}
```

The OnExecuteClear method respond to Clear commands sent by the client application:

```
void CScribbleView::OnExecuteClear(Guid sessionId, Typeless command, Typeless responses)
{
    CBrush br(RGB(255,255,255));
    m_dcOffscreen.SelectObject(&br);
    CRect rect(0,0,m_Width,m_Height);
    CBrush *pbr = &br;
    m_dcOffscreen.FillRect(&rect,pbr);
    m_startpt = -1;
    CScribbleApp::StateManager().ViewManager().RenderViewDeferred("ScribbleView");
}
```

Two final noteworthy aspects of ScribbleView are the calls to CScribbleApp::StateManager().SetViewInteracting and CScribbleApp::StateManager():RenderViewDeferred in the OnLButtonDown, OnLButtonUp, and OnMouseMove methods. Calling SetViewInteracting("ScribbleView", true) causes images to be returned to the client application using a reduced encoding quality. When the user is interacting with the ScribbleView, new images are generated at an increased rate. Using a lower quality to encode the images conserves bandwidth. Calling SetViewInteracting("ScribbleView", false) restores the image encoding quality. RenderViewDeferred is used (as opposed to RenderViewImmediate) to defer the service from obtaining an updated image for the client until an image response is about to be sent.

```
void CScribbleView::OnLButtonDown(UINT nFlags, CPoint point)
{
    m_startpt.x = point.x;
    m_startpt.y = point.y;
    CScribbleApp::StateManager().ViewManager().SetViewInteracting("ScribbleView", true);
}

void CScribbleView::OnLButtonUp(UINT nFlags, CPoint point)
{
    m_startpt = -1;
    CView::OnLButtonUp(nFlags, point);
    CScribbleApp::StateManager().ViewManager().SetViewInteracting("ScribbleView", false);
}

void CScribbleView::OnMouseMove(UINT nFlags, CPoint point)
{
    m_endpt.x = point.x;
    m_endpt.y = point.y;

    if (m_startpt.x != -1)
    {
        CPen pen( PS_SOLID, 4, m_cColor);
        CPen* pOldPen = m_dcOffscreen.SelectObject( &pen );
        m_dcOffscreen.MoveTo(m_startpt.x, m_startpt.y);
        m_dcOffscreen.LineTo(m_endpt.x, m_endpt.y);
        m_startpt.x = m_endpt.x;
        m_startpt.y = m_endpt.y;
        CScribbleApp::StateManager().ViewManager().RenderViewDeferred("ScribbleView");
        Invalidate();
    }

    CView::OnMouseMove(nFlags, point);
}
```

ScribbleClient Code Description

The following code is found in the MainPage class of the MainPage.xaml.cs file.

The connect_Click method listens for buttons clicks on the connect/disconnect button. When the button is clicked, a the method first determines if the is connected. If the client is connected then the client is disconnected. If the method determines that the client is *not* connected then basic authentication is done before the connection is made.

```
private void connect_Click(object sender, RoutedEventArgs e)
{
    if (Framework.Client.IsConnected)
    {
        Framework.Client.Disconnect();
    }
    else
    {
        // delayed connect model
        string href = (string)HtmlPage.Window.Eval("document.location.href");
        var uri = new Uri(href);
        href = string.Format("{0}://{1}:{2}/pureweb/app?name=ScribbleApp", uri.Scheme, uri.Host, uri.Port);
        Trace.WriteLine("Connecting to server {0}", href);

        BasicAuthorizationInfo auth = new BasicAuthorizationInfo();
        auth.Name = "admin";
        auth.Password = "admin";
        Framework.Client.AuthorizationInfo = auth;

        Framework.Client.Connect(href);
    }
}
```

The color_LostFocus method listens for focus lost events on the color text field. When the color text field loses focus, the ScribbleColor application state property is set to the current color.

```
private void color_LostFocus(object sender, RoutedEventArgs e)
{
    // set the scribble color
    Framework.State["ScribbleColor"] = this.color.Text;
}
```

The clear_Click method listens for buttons clicks on the Erase All button. When the button is clicked, a Clear command to the service application. The service application responds by clearing previously drawn scribbles, and sends an updated blank image back to the client.

```
private void clear_Click(object sender, RoutedEventArgs e)
{
    // send a clear command
    Framework.Client.QueueCommand("Clear");
}
```

The share_Click method is invoked when the Share button is clicked. The method calls the ShowCollaborationDialog method which will display in the Share URL dialog.

```
private void share_Click(object sender, System.Windows.RoutedEventArgs e)
{
    Framework.Client.GetSessionShareUrlAsync
        ("Scientific","Collable",TimeSpan.FromMinutes(30),null,(url, exception) =>
    {
        if (exception == null)
        {
            ShowCollaborationDialog(url);
        }
        else
        {
            MessageBox.Show("Unable to create share: " + exception);
        }
    });
}
```

Build ScribbleApp

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

Building ScribbleApp

Purpose: To build ScribbleApp in Microsoft Visual Studio 2010.

Start of procedure

1. Navigate to C:\CSI\PureWeb\SDK\Samples\Scribble\ScribbleAppCpp\.
2. Double-click on the ScribbleApp2010.sln file to open it in Microsoft Visual Studio 2010.
3. Build the solution from the Build menu.
4. Check the output to ensure that the build was successful.

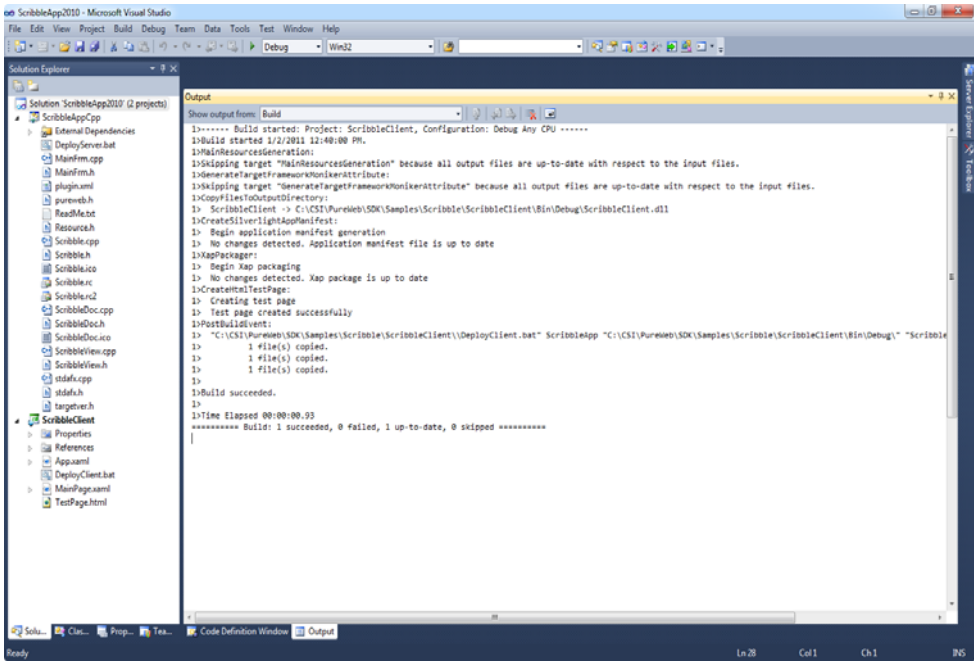

Figure 3: Building the ScribbleApp Solution

End of procedure

Next Steps

- Follow the steps in the "Running ScribbleApp" procedure.

Run ScribbleApp

Follow the steps in the procedure below to run the sample application.

Running ScribbleApp

Purpose: To run ScribbleApp.

Prerequisites

- You must have completed the steps in the "Building ScribbleApp" procedure.
- You must have completed the steps in the Reloading a Plugin procedure, found in the *PureWeb® Server Administration Guide*.
- Restart the PureWeb® Server.

Start of procedure

1. Launch the PureWeb Apache Tomcat Server with the Start PureWeb desktop icon.
2. Open a browser window.
3. Navigate to

http://localhost:8080/app?name=ScribbleAppCpp&client=silverlight&_diagnostics=true.

4. Enter admin/admin into the Name and Password fields.

Figure 4: PureWeb® Server Login

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

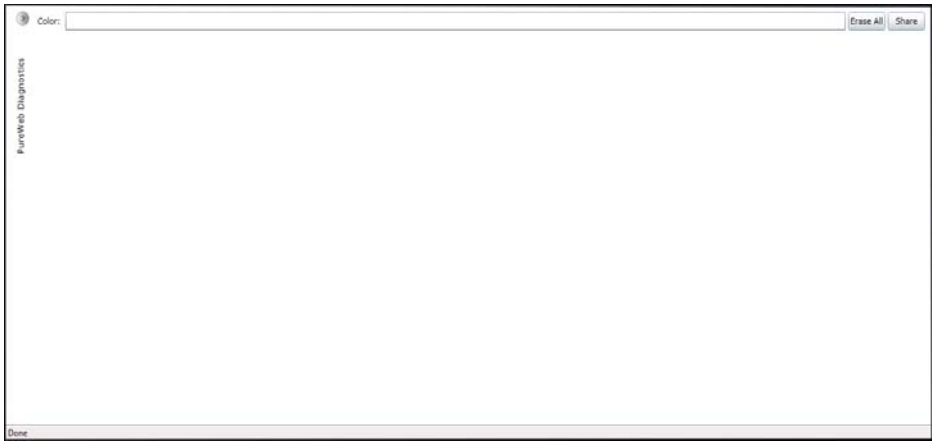

Figure 5: Scribble Client

End of procedure

Sample Functionality

The ScribbleApp client application supports the following functionality, as seen in Figure 6:

- Draw lines on the canvas by left-clicking the mouse and dragging.
- 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.
- Close the browser window or tab containing the Silverlight application. Note that the service application window also closes.

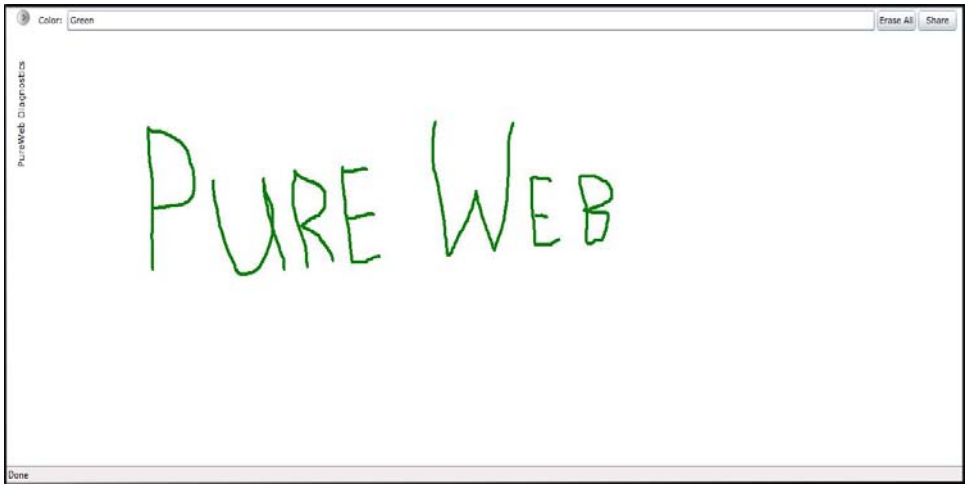

Figure 6: Scribble Functionality

Accessing the Diagnostics Panel

Purpose: To show or hide the diagnostics panel.

Start of procedure

1. Click on the arrow button located in the upper-left corner of the ScribbleApp client, see Figure 7.

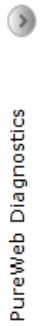

Figure 7: Show/Hide Diagnostics

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 (see page 16), lowering the interactive quality reduces the bandwidth usage by lowering the image quality.
  - Full quality: When SetViewInteracting is enabled (see page 16), this controls bandwidth and interactive quality.

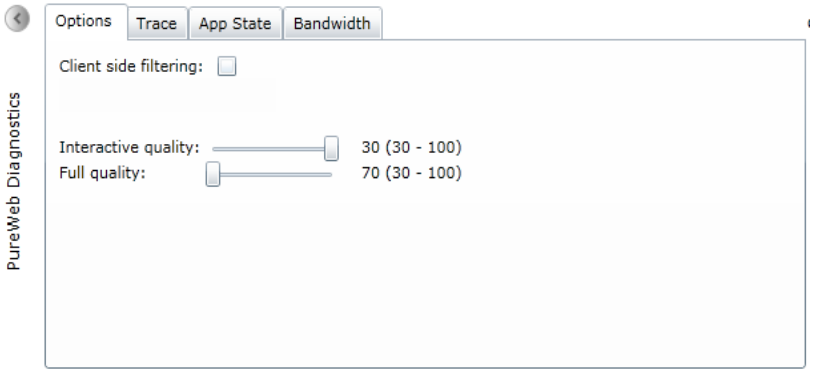

Figure 8: Options Tab

- 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 the PureWeb.Diagnostics.Trace.WriteLine method.

Figure 9: 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.

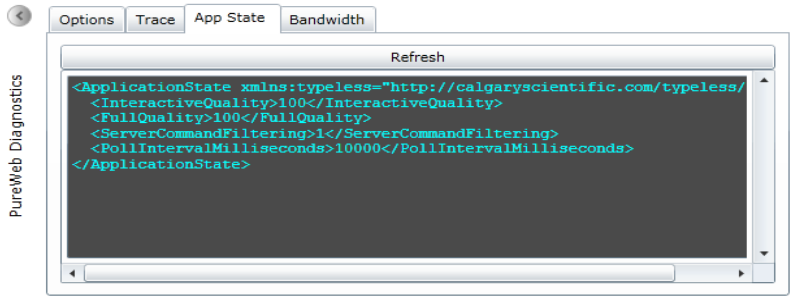

Figure 10: App State Tab

- 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.

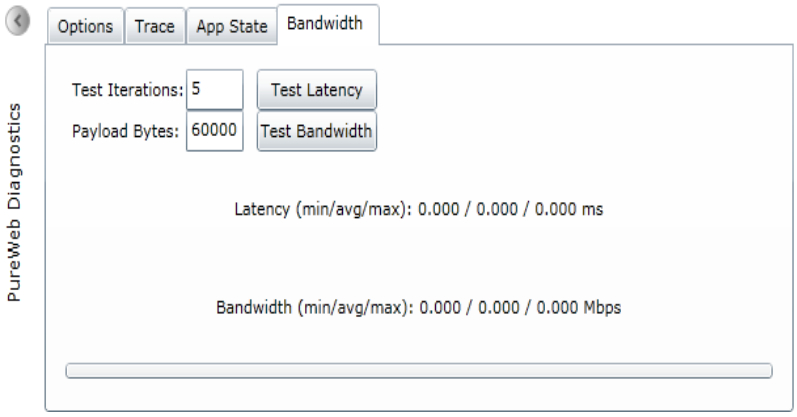

Figure 11: Bandwidth Tab

End of procedure

Sharing ScribbleApp

Purpose: To share the ScribbleApp client application.

Start of procedure

Warning! By default the ScribbleApp client 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, by replacing localhost with the IP address.

First person:

1. Click the Share button located in the upper-right corner of the ScribbleApp client application.

Figure 12: Share Button

2. Notify a second person of the URL displayed in the popup window and click the close button.

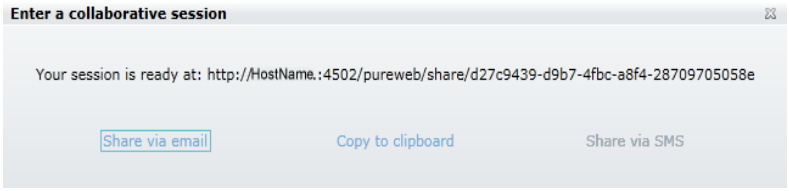

Figure 13: Share URL

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 ScribbleApp session.

Figure 14: Collaboration Login Page

End of procedure

Debug ScribbleApp

The information in this section explains the procedure for debugging the client and the service applications using Microsoft Visual Studio 2010.

Note: You should make Microsoft Internet Explorer your default browser when debugging Silverlight applications using Microsoft Visual Studio.

Note: To debug Silverlight applications in Microsoft Internet Explorer 8/9, deselect the Enable Protected Mode option found at Tools | Internet Option | Security.

For more information, refer to the *Cannot trigger breakpoints in debug mode* section of the *PureWeb Troubleshooting Guide*.

Debugging the ScribbleApp Client Application

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

Start of procedure

1. Open ScribbleApp2010.sln in Microsoft Visual Studio 2010.
2. Select the ScribbleClient project and Set as Startup Project (from Project menu or Context menu).
3. Select TestPage.html in the ScribbleClient project and Set as Start Page (from Project menu or Context menu).
4. Launch the PureWeb Apache Tomcat Server with the Start PureWeb desktop icon.
5. Start Debugging (from the Debug menu or by pressing F5). Your default web browser (Microsoft Internet Explorer) will open with the test page displayed.
6. Click on the Launch Scribble Demo C++ with diagnostics link.

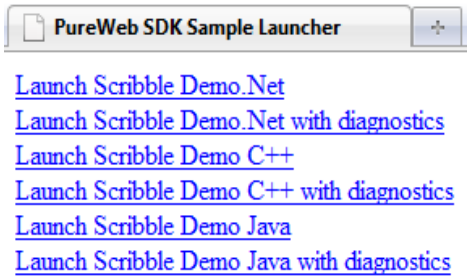

Figure 15: Launch Scribble Client

7. Enter admin/admin into the Name and Password fields.

Figure 16: PureWeb® Server Login

8. Click the Sign In button to launch the application.
9. Set a breakpoint in the OnScribbleColorChanged method in the MainPage.xaml.cs file.

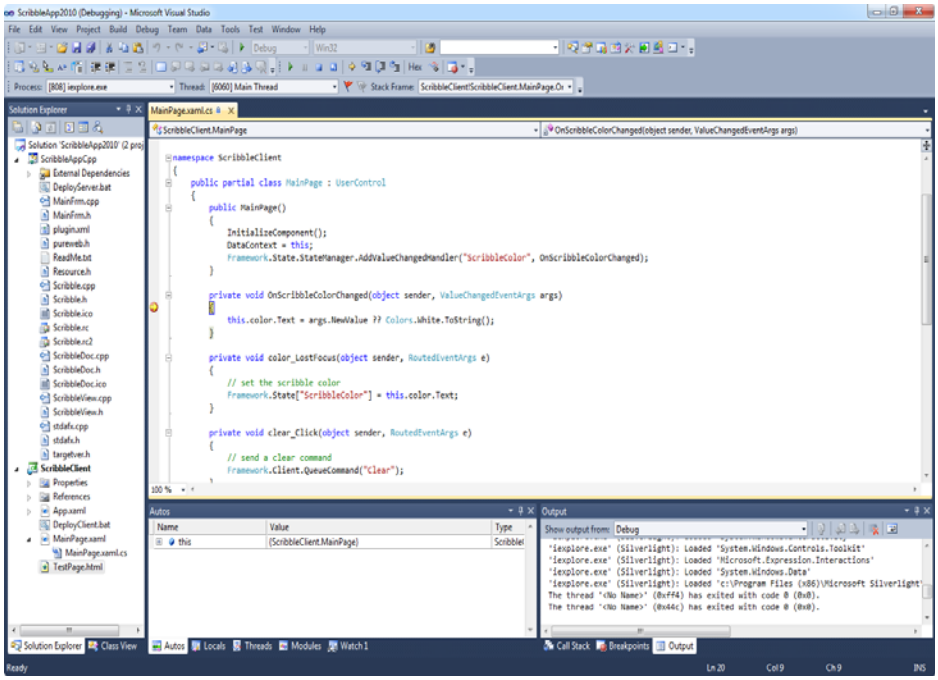

Figure 17: Debug Scribble Client

10. Enter a new color in the Color text box of the client application to trigger the breakpoint. Note that the field must lose focus in order to 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. Open ScribbleApp2010.sln in Microsoft Visual Studio 2010.
2. Rebuild the solution from the Build menu or by pressing Ctrl+Alt+F7.
3. Launch the PureWeb Apache Tomcat Server with the Start PureWeb desktop icon.
4. Open a browser window (Microsoft Internet Explorer).
5. Navigate to http://localhost:8080/app?name=ScribbleAppCpp&client=silverlight.
6. Enter admin/admin into the Name and Password fields.

Figure 18: PureWeb® Server Login

7. Click the Sign In button to launch the application.
8. Select Debug | Attach To Process in Microsoft Visual Studio 2010.
9. Select ScribbleAppCpp.exe from the list of Available Processes and click the Attach button.

10. Set a breakpoint in the OnExecuteClear method in the ScribbleView.cpp file.

11. Click the Erase All button in the client application to trigger the breakpoint.

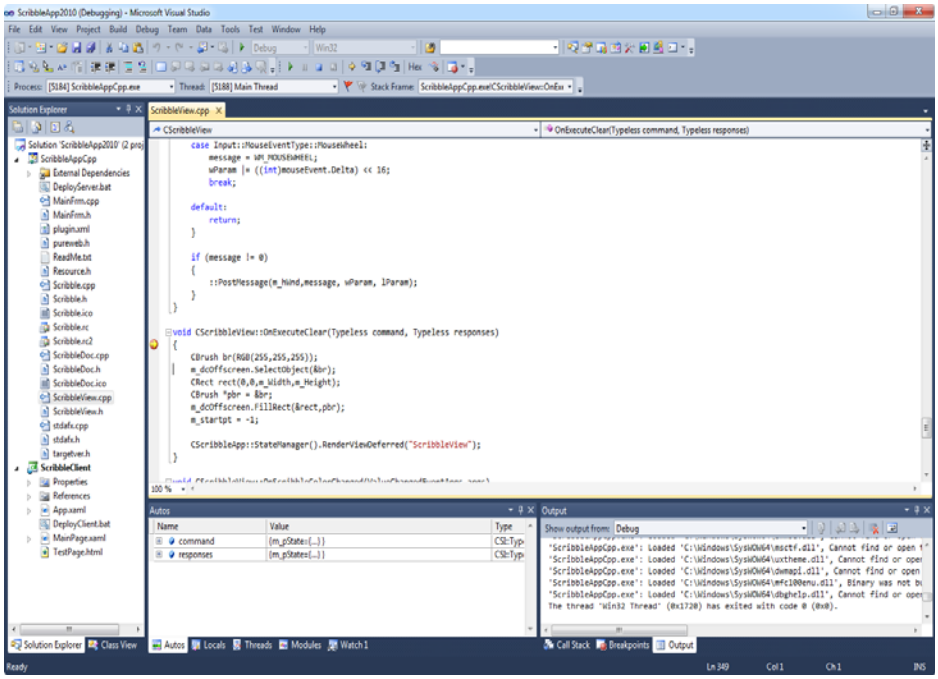

Figure 19: Debug Scribble Service

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® Server Administration Guide*, which provides information about the PureWeb® Server.
- *PureWeb® STK Installation Guide for Microsoft Windows*, which provides detailed information on installing PureWeb® STK on a Microsoft Windows operating system.
- *PureWeb® Silverlight Client STK API Reference*, which describes the Silverlight Client STK.
- *PureWeb® C++ Server STK API Reference*, which describes the C++ Server 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 32).	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

R

S

T

U

V