Microsoft®
Silverlight™

Silverlight 4 Overview

Technical Feature Overview

Contents

Credits

This document was written in collaboration with many people including most of the Microsoft Silverlight product team, the WCF RIA Services team, and the Expression Blend team. Their help in providing resources, content, and feedback was invaluable.

Special thanks to Adam Kinney who authored the Expression Blend content, which is a selected excerpt from the full Expression Blend Technical Whitepaper found at this link http://go.microsoft.com/fwlink/?linkid=186042.

If you have any feedback on this document, contact:

- John Papa - jopapa@microsoft.com
- Adam Kinney - adkinn@microsoft.com

Introduction

Silverlight 4 enhances the building of business applications, media applications, and applications that reach beyond the browser. New features include printing support, significant enhancements for using forms over data, support for several new languages, full support in the Google Chrome web browser, WCF RIA Services, modular development with MEF, full support in Visual Studio 2010, bi-directional text, web camera and microphone support, rich text editing, improved data binding features, HTML support, MVVM and commanding support, new capabilities for local desktop integration running in the new "Trusted Application" mode such as COM automation and local file access.

This document covers the Silverlight 4 RC. Some elements may change prior to the final release of Silverlight 4.

This document explains the new features in Silverlight, where you can find learning materials, and identifies additional resources.

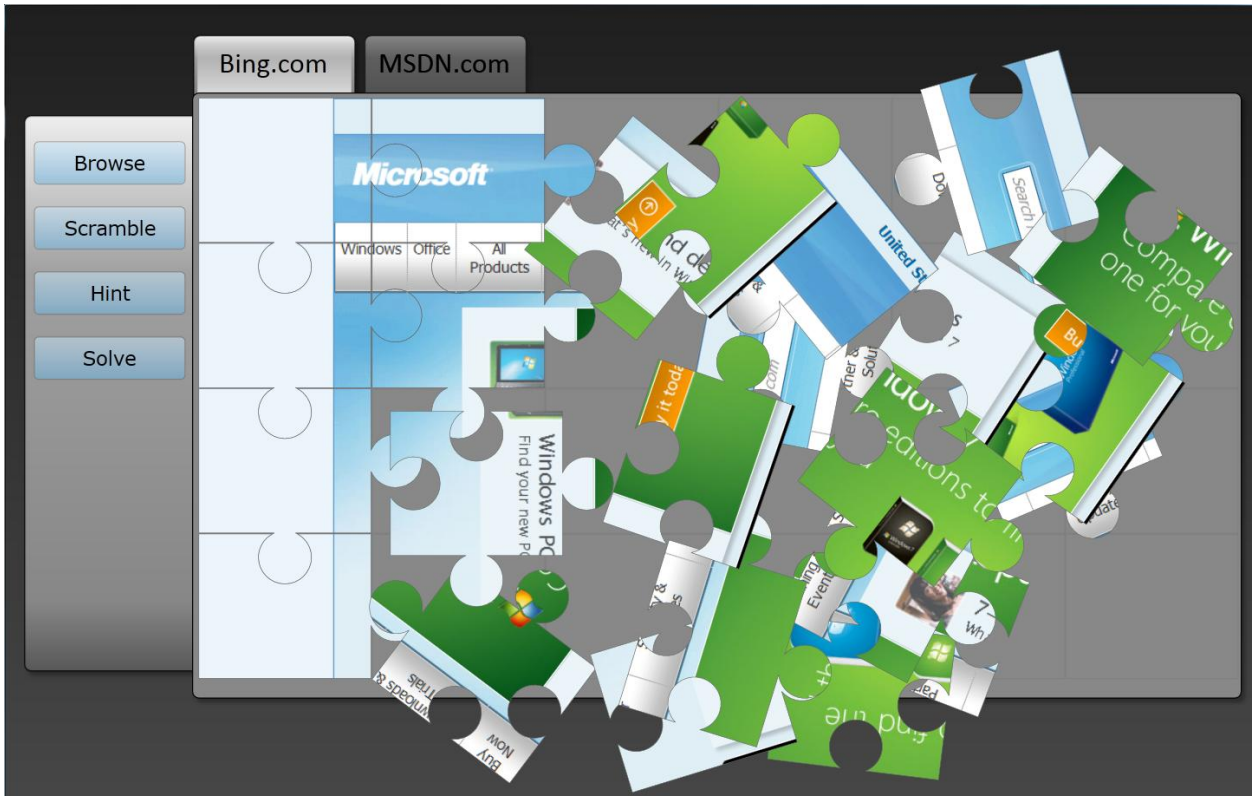

Figure 1

Silverlight 4 Multi Touch Puzzle Using the New Trusted Application, WebBrowser Control, and WebBrowserBrush Features

Enabling Business Application Development

Printing

Silverlight adds printing support that allows developers to control whether their Silverlight application can print itself, how the content is formatted when printed, and determine the content that will appear. For example, you could add a print button to your Silverlight application that opens the Print dialog allows the user to choose the printer, and then prints the Silverlight content.

You can print content by first creating an instance of the PrintDocument class, then specifying the content to print, and finally writing code to handle the PrintPage event. You can print the entire Silverlight control by setting PrintPageEventArgs.PageVisual to the layout root of the Silverlight content. Alternatively, you can print a portion of the Silverlight content by setting PrintPageEventArgs.PageVisual to the named UIElement you wish to print. The following code sample prints all content within the UIElement named in the Silverlight application.

C#

```
PrintDocument pd = new PrintDocument();
pd.PrintPage += (s, args) =>
  {
      args.PageVisual = LayoutRoot;
  };
pd.Print();
```

After the PrintPage event occurs, the content from the PageVisual is sent to the selected printer to be printed. Content that is too large to fit in to the PrintableArea will be clipped. If HasMorePages is true, PrintPage occurs multiple times until HasMorePages is false. If you require customized printing options, you can write code for the StartPrint and EndPrint events to perform special handling.

You can print content that is not in the live visual tree. This allows you to make your printed content look different from your onscreen content. To do this, set PageVisual to a UIElement created in the PrintPage event without adding it to the live visual tree. Another tip to print elements which are not in the live tree is to explicitly run layout on them by calling measure and arrange.

Localization with Bi-Directional and Script-Based Text and Right-to-Left Support

Silverlight introduces support for shaped and script languages that flow right to left, such as Arabic and Hebrew. The ability to change the direction of the flow is available to all Silverlight UI elements by setting the FlowDirection property to LeftToRight or RightToLeft. The default bi-directional (Bi-Di) FlowDirection property is from LeftToRight. Setting FlowDirection to RightToLeft on any element sets the alignment to the right, the reading order to right-to-left, and the layout of the control to flow from the right to the left. Most UIElement controls inherit the FlowDirection from their parent; however

Image, MediaElement, MultiScaleImage, and Popup do not and remain LeftToRight unless overridden explicitly. When brushes and effects are applied to elements whose FlowDirection property is set RightToLeft, the result is flipped horizontally.

Extended Language Support

Silverlight 4 now supports several new languages including Thai and Vietnamese as well as multiple Indic scripts both for rendering and input. The following Indic scripts are now supported:

Script	Languages
Bengali	Bengali, Assamese, Manipuri
Oriya	Oriya
Malayalam	Malayalam
Kannada	Kannada
Tamil	Tamil
Telugu	Telugu
Gujarati	Gujarati
Gurmukhi	Punjabi
Devanagari	Hindi, Marathi, Sanskrit, Konkani, Kashmiri, Nepali, Sindhi

RichTextBox Control

Silverlight adds the new RichTextBox control to the standard controls offered by Silverlight. Through the FlowDirection property, the RichTextBox has Bi-Directional (Bi-Di) support on content including Runs. It also allows hyperlinks, XAML content, embedding of images, changing the font size, foreground color, and making text bold, italicized, and underlined as well as many other rich text features.

The RichTextBox supports the “format then type” feature. For example, with this feature you can choose a format, such as bold or font color of green, and when you start typing the formatting selections you made are applied to the text you type. Text position and selection APIs are also supported.

Another feature of the RichTextBox control is paragraph blocks, which allows different formatting of blocks of text. Built-in keyboard commands can be used to allow shortcuts for clipboard and undo operations. The RichTextBox also supports mouse events, XAML to clipboard features, undo, and language localization.

The following image shows the RichTextBox control using the FlowDirection = RightToLeft and displaying Hebrew text.

Figure 2

RichTextBox with FlowDirection of RightToLeft

The following image shows a sample application with a RichTextBox control containing several different types of content, including embedded images, an embedded Silverlight DataGrid, different font sizes, different fonts, foreground colors, bold, and italics.

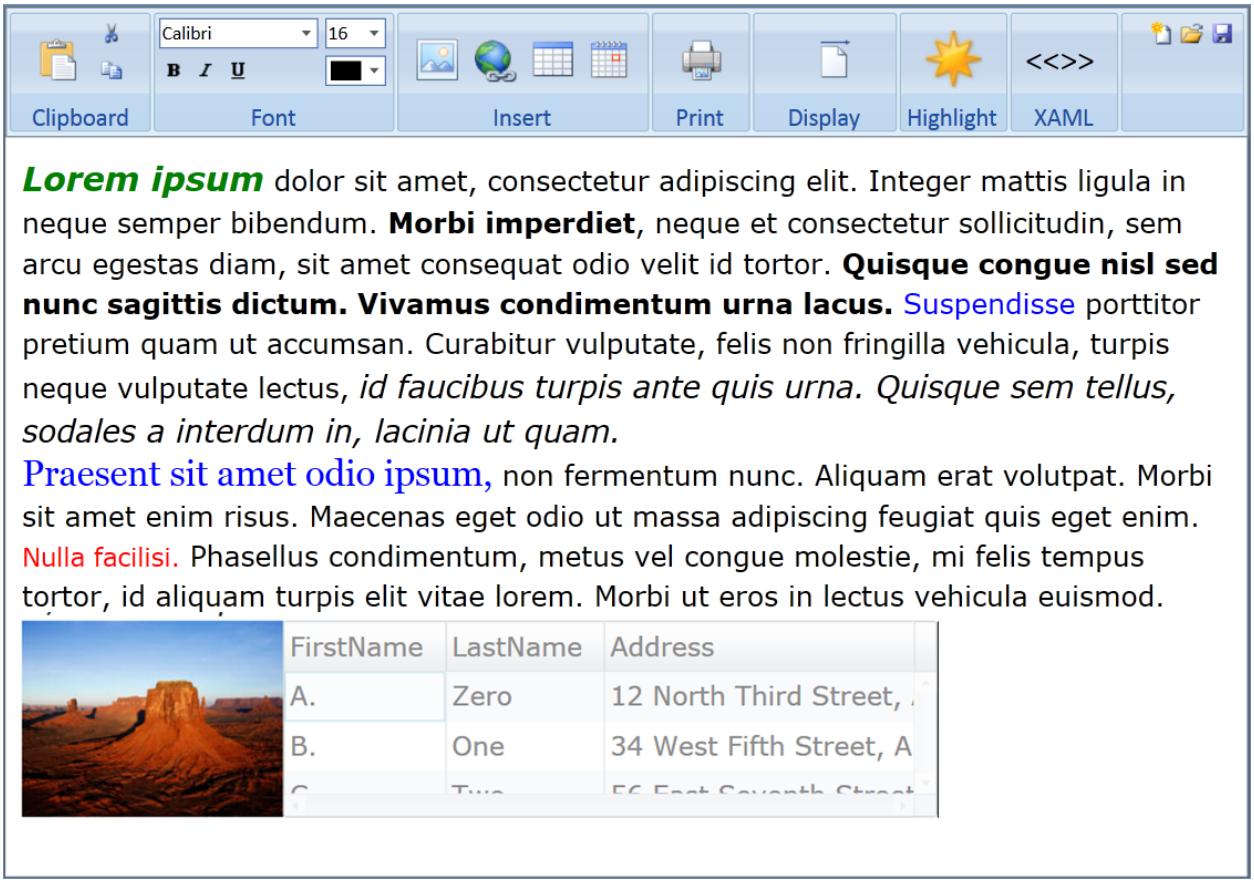

Figure 3

RichTextBox Control

The RichTextBox has a Xaml property that supports serializing text content. You can get the Xaml from the RochTextBox's Xaml property and save it to isolated storage or perhaps send it to a web service. You

can also set the content of the RichTextBox control by setting the Xaml property. You can clear the content of a RichTextBox by calling the Blocks property's Clear method.

Text Input

Silverlight now enhances its support for IME (Input Method Editors), which is a part of text input used when there are more glyphs than keyboard keys (as with East Asian languages). Enhancements include getting and setting IME conversion modes as well as getting and setting IME active state. In addition, the TextInputStart, TextInputUpdate, and TextInput events have been added to the UIElement controls. These events are effective at determining the text that users input when using IME.

Silverlight also has additional event handling for text input via the TextInput event. The TextInput event allows you to get an event from key press generated characters. This includes getting characters that require multiple keyboard presses in order to be entered, such as the "è" in the name "Claudè".

Viewbox Control

Silverlight adds the Viewbox to its list of layout containers. The Viewbox allows precise positioning and flexible resizing support that stretches or scales a single child element to fit the available space. A Viewbox can only contain a single element.

The Viewbox's StretchDirection property determines how to scale its contents. It prevents the contents of the Viewbox from being scaled to a smaller or larger dimension than the original. Possible values are UpOnly, DownOnly and Viewbox Control Both. The Stretch property determines how the content of the Viewbox will be stretched to fill the Viewbox. Possible values are None, Fill, Uniform, and UniformToFill.

The example below shows a Viewbox control with the default settings of Stretch=Uniform and StretchDirection=Both. The image inside the Viewbox is 200x200 pixels and will be stretched uniformly to fill the entire smaller Viewbox.

XAML

```
<Grid x:Name="LayoutRoot" Background="Red" Width="100"
Height="100" >
<Viewbox Margin="5">
             <Image Source="sl4 image.png"/>
</Viewbox>
</Grid>
```

Figure 4

Viewbox with Default Settings

If you set the StretchDirection to UpOnly, then the image will only stretch up in size. Since the container is smaller than the image, the image will not stretch and only the upper portion of the image will be visible.

Figure 5

Viewbox with StretchDirection = UpOnly

Auto-Sizing Columns and Copy from DataGrid

Silverlight added several new features to the DataGrid control. Through auto-sizing columns, the DataGrid in Silverlight now has the ability to allow columns to share the remaining width of a DataGrid. This feature makes it easy to define specific widths for some columns in a DataGrid while allowing the other columns to share the remaining width of the DataGrid. For example, you might want to have fixed widths for columns for City, State, and the Postal Code. However, you may want the Name column to use 1/3 of the remaining space and the Address column to use 2/3 of it. The following code would accomplish this using * column widths:

XAML

```
XAML
<data:DataGrid AutoGenerateColumns="False" x:Name="dataGrid2"
                   HeadersVisibility="All">
    <data:DataGrid.Columns>
        <data:DataGridTextColumn Header="Name" Binding="{Binding Name}" Width="*"/>
        <data:DataGridTextColumn Header="Address" Binding="{Binding Address}"
                   Width="2*"/>
        <data:DataGridTextColumn Header="City" Binding="{Binding City}" Width="100"/>
        <data:DataGridTextColumn Header="State" Binding="{Binding State}"
                   Width="50" />
        <data:DataGridTextColumn Header="Postal Code" Binding="{Binding PostalCode}"
                   Width="80"/>
    </data:DataGrid.Columns>
</data:DataGrid>
```

Figure 6

DataGrid with Auto Sizing Columns

The DataGrid now supports MouseWheel scrolling and the FlowDirection property. In addition, DataGrid supports row level copy. This means you can copy a row of a DataGrid to the clipboard and then paste it to another program.

Navigation Page Loading Extensibility

Silverlight 3 introduced page navigation that resolves URIs into pages. Every page had to have an associated XAML and navigating to that page required URIs with paths to those XAML files based upon your project's file structure. Silverlight 4 adds a public ContentLoader property to the Frame control and the INavigationContentLoader interface. Together, these changes allow the developer to generate and initialize instances of pages from URIs using their own scheme, and enable scenarios such as authentication redirects, error pages, and MVC-style navigation.

Command Property on ButtonBase and Hyperlink

Silverlight adds commanding support commonly used in the Model-View-ViewModel (MVVM) pattern, which provides separation between the View (the Silverlight user control) and the ViewModel (the logic and bindings for the View). Silverlight adds support for commanding on controls that inherit from ButtonBase and Hyperlink. The Command and CommandParameter are exposed to allow binding from a View to a ViewModel without the need for code in the codebehind. You can implement a command by first creating your own DelegateCommand and binding to it from the XAML.

For example, a ViewModel may contain a LoadProducts command, which implements the ICommand interface. The LoadProductsCommand accepts a parameter that filters the products that start with a specific letter(s). The following XAML shows how a Button's Command property could be bound to the LoadProductsCommand and how its CommandParameter property could be bound to a TextBox's Text property.

XAML

```
<UserControl.Resources>
    <local:ProductViewModel x:Key="vm"/>
</UserControl.Resources>
<Grid DataContext="{StaticResource vm}" Name="LayoutRoot">
<TextBox x:Name="MyFilterTextBox"/>
<Button Content="Load" Width="120"
    Command="{Binding LoadProductsCommand}"
    CommandParameter="{Binding Path=Text, ElementName=MyFilterTextBox}"/>
```

This XAML binds the Button to the LoadProductsCommand in the ViewModel. You can enable or disable the Button depending on the Boolean return value from the command's CanExecute method.

SelectedValue and SelectedValuePath on the Selector

These properties enable the use of ComboBox as a look up table. When combining DisplayMemberPath with these you can display one thing but bind to another. All controls that inherit from the Selector class now have SelectedValue and SelectedValuePath properties. You can now set the SelectedValuePath

property to the string name of the property of the SelectedItem. When this is set, the SelectedValue property will return the SelectedItem's property with the matching name of the SelectedValuePath.

For example, you have a control that derives from Selector (such as the ComboBox control) and it is bound to a collection of Product classes. The Product class has ProductName and ProductId properties. If the SelectedValuePath is set to ProductName , when an item in the Selector control is selected, the SelectedValue property will return the value for the ProductName of the SelectedItem. The following XAML demonstrates this example.

XAML

```
<UserControl.Resources>
    <local:ProductViewModel x:Key="vm"/>
    <DataTemplate x:Key="ProductDataTemplate">
       <Grid>
              <TextBlock TextWrapping="Wrap" Text="{Binding Path=ProductName}" />
       </Grid>
    </DataTemplate>
</UserControl.Resources>
<StackPanel x:Name="LayoutRoot"
            Background="White" DataContext="{StaticResource vm}"
            Orientation="Vertical" Margin="10">
    <ComboBox Height="23" Name="ProductComboBox" VerticalAlignment="Top" Width="120"
            SelectedValuePath="ProductName"
            ItemsSource="{Binding Path=Products}"
            ItemTemplate="{StaticResource ProductDataTemplate}" />
    <TextBlock Height="23" HorizontalAlignment="Left"
            Name="SelectedPathTextBlock" Text="{Binding Path=SelectedValue,
            ElementName=ProductComboBox}" VerticalAlignment="Top" Width="120" />
</StackPanel>
```

The result of this XAML is that when you select a product, such as Strawberry, the ProductName of the SelectedItem is displayed in the TextBlock, as shown below.

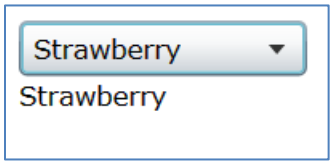

Figure 7

TextBlock Bound to SelectedValue of a ComboBox

Another valuable scenario is when a class, for example a Customer class, has a public string StateId {get;set;} which takes the value "WA" or "NV" etc. You want to display a picker that displays the full state name but when you pick a value pushes the short name into Customer.StateId. You can create a State object with LongName and ShortName and a collection of all the States called StatesCollection. Then you bind ComboBox.ItemsSource to StatesCollection with DisplayMemberPath=LongName, SelectedValuePath=ShortName and bind SelectedValue to the current Customer StateId property. In the end, you will display "Washington" but SelectedValue is "WA" and so Customer.StateId gets set to "WA"

XAML

```
<StackPanel x:Name="LayoutRoot"
            Background="White" DataContext="{StaticResource customerVM}"
            Orientation="Vertical" Margin="10">
    <ComboBox Height="23" Name="ProductComboBox" VerticalAlignment="Top" Width="120"
            SelectedValue=”StateId”
            DisplayMemberPath=”LongName” SelectedValuePath="ShortName"
            ItemsSource="{Binding Source="{StaticResource statesCollection}}"
            ItemTemplate="{StaticResource ProductDataTemplate}" />
</StackPanel>
```

Support for Referer Header

Silverlight allows the Referer request header for both in-browser and out-of-browser scenarios on all networking requests used by WebClient and HttpWebRequest when they use the Client networking stack. This enables services like Live Mesh to determine where the requests originated. The Referer header is a request header field that is automatically set to the base URI of the client from which the Request-URI was obtained. The Referer request header allows a server to generate lists of back links to resources for logging and caching optimization.

UDP Multicast Client Support

Silverlight enables UDP multicast for one-to-many communication over an IP infrastructure. This efficiently scales to a large receiver population. There are two key shapes for multicast groups:

- Single Source Multicast (one-to-many)
  System.Net.Sockets.UdpSingleSourceMulticastClient
- Any Source Multicast (many-to-many)
  System.Net.Sockets.UdpAnySourceMulticastClient

To make multicast group accessible to Silverlight, you need to expose a policy responder (just like Sockets and HTTP).

Networking Enhancements

Silverlight 4 adds the following features to the networking stack:

- UploadProgress support on the client networking stack now exists. This allows you to be notified as content is being uploaded so you can notify the user.
- Caching support on the client networking stack has been added.
- Sockets policy file retrieval via HTTP is also new.
- Accept-Language header has been added.

WCF TCP-based Binding

WCF in Silverlight now offers a TCP-based binding, which enables communicating with WCF services configured with NetTcpBinding. Note that no security is supported in this release.

WCF PollingDuplex Improvements

PollingDuplex now supports a new “multiple messages per poll” mode, where HTTP chunking is used to stream many messages in response to a client poll.

Authentication Support with ClientHttpWebRequest

Silverlight adds the ability to pass user credentials from the application to the server to support NTLM, Basic, and Digest authentication. This allows Silverlight applications to pass user credentials for authentication by services such as ADO.NET Data Services or Live Mesh.

A Silverlight application can now set the Credentials API with a username and password when Web sites require the authorization request header to be sent with a network request to properly authorize users. When a user has already logged in to the machine and the credentials have been saved to the OS, the client may use these default credentials to authenticate when making a client networking request. This makes the most sense in scenarios (such as SharePoint) that use NTLM authentication.

The code below demonstrates how to pass network credentials using the new APIs. Notice that the UseDefaultCredentials property is set to false. When UseDefaultCredentials is set to true, the Credentials property is ignored and the credentials are retrieved from the local machine’s session. In the code above, the credentials are supplied in code, so the UseDefaultCredentials must be set to false.

C#

```
var username = "myTwitterId";
var password = "myTwitterPassword";
var twitterUri =
        @"http://twitter.com/statuses/friends_timeline/{0}.xml?count=50";
var uriString = string.Format(twitterUri, username);

WebClient request = new WebClient();
WebRequest.RegisterPrefix("http://", WebRequestCreator.ClientHttp);
request.UseDefaultCredentials = false;
request.Credentials = new NetworkCredential(username, password);
request.DownloadStringCompleted += new
        DownloadStringCompletedEventHandler(client_DownloadStringCompleted);
request.DownloadStringAsync(new Uri(uriString));
```

Custom authentication is now supported for in browser and out of browser applications through programmatic access to the **Authorization** header via the **Headers** collection on WebRequest. Also, you will need the clientaccesspolicy.xml file to opt-in to the Authorization header being sent.

IDataErrorInfo

Silverlight adds the IDataErrorInfo interface enables the reporting of validation errors that a user interface can bind to. When an entity implements this interface and the entity is involved in a binding

operation, it invokes the indexer to validate the properties. The bound target properties in the UI will receive the error messages and display the validation states if the ValidatesOnDataErrors property is set to true.

The IDataErrorInfo interface exposes an Error property and a string indexer. The Error property should return an error message explaining the error with the object. The indexer should return the error message for the property with the given name passed to the indexer.

C#

```
public interface IDataErrorInfo
{
    string Error { get; }
    string this[string columnName] { get; }
}
```

You can implement validation by executing the IDataErrorInfo interface on an entity model as shown in the code sample below on the Employee. The Employee has custom validation rules that execute when changes occur in the bound UI elements that have ValidatesOnDataErrors set to true.

C#

```
public class Employee : INotifyPropertyChanged, IDataErrorInfo
    {
        private ValidationHandler validationHandler = new ValidationHandler();
        private string _FirstName;
        public string FirstName
        {
            get { return _FirstName; }
            set
            {
                _FirstName = value;
                NotifyPropertyChanged("FirstName");
                bool valid = validationHandler.ValidateRule(
                    "FirstName",
                    "First Name must be at least 5 letters!",
                     () => (value.Length >= 5));
            }
        }
        private float _TaxPercent;
        public float TaxPercent
        {
            get { return _TaxPercent; }
            set
            {
                if (_TaxPercent != value)
                {
                    if (value >= 1)
                        value /= 100;
                    _TaxPercent = value;
                    NotifyPropertyChanged("TaxPercent");
```

```
                bool valid = validationHandler.ValidateRule(
                    "TaxPercent", "The tax has to be positive!",
                     () => (value > 0));
            }
        }
    }
    protected void NotifyPropertyChanged(string PropertyName)
    {
        if (null != PropertyChanged)
            PropertyChanged(this, new PropertyChangedEventArgs(PropertyName));
    }
    public event PropertyChangedEventHandler PropertyChanged;
    public string Error
    {
        get { return null; }
    }
    public string this[string columnName]
    {
        get
        {
            if (this.validationHandler.BrokenRuleExists(columnName))
            {
                return this.validationHandler[columnName];
            }
            return null;
        }
    }
}

public class ValidationHandler
{
    private Dictionary<string, string> BrokenRules { get; set; }
    public ValidationHandler()
    {
        BrokenRules = new Dictionary<string, string>();
    }
    public string this[string property]
    {
        get
        {
            return this.BrokenRules[property];
        }
    }
    public bool BrokenRuleExists(string property)
    {
        return BrokenRules.ContainsKey(property);
    }
    public bool ValidateRule(string property,
                string message, Func<bool> ruleCheck)
```

```
        {
            if (!ruleCheck())
            {
                this.BrokenRules.Add(property, message);
                return false;
            }
            else
            {
                RemoveBrokenRule(property);
                return true;
            }
        }
        public void RemoveBrokenRule(string property)
        {
            if (this.BrokenRules.ContainsKey(property))
            {
                this.BrokenRules.Remove(property);
            }
        }
    }
```

The XAML for these UI elements is shown below.

XAML

```
<TextBox x:Name="FirstNameTextBox"
    Text="{Binding Path=FirstName, Mode=TwoWay,
    ValidatesOnDataErrors=True}" />
<TextBox x:Name="TaxPercentTextBox"
    Text="{Binding Path=TaxPercent, Mode=TwoWay,
    ValidatesOnDataErrors=True}" />
```

When invalid values are entered into these controls, the validation kicks in and displays to the user as shown below.

Figure 8

Data Validation using IDataErrorInfo

INotifyDataErrorInfo

IDataErrorInfo is limited to validating on a per property basis. However, Silverlight 4 also adds the INotifyDataErrorInfo interface that allows validation across properties of an entity. It also allows entity objects to enable notification of data errors in the UI. INotifyDataErrorInfo allows developers to provide custom, asynchronous validation support to access server-side validation logic. It exposes a HasErrors property to indicate if there are any errors and has a GetErrors method to retrieve the errors. The ErrorsChanged event is raised when new errors are added. If the binding property ValidatesOnNotifyDataErrors is set to true and the source object implements the interface, the binding

engine in Silverlight will listen for the ErrorsChanged event. INotifyDataErrorInfo also provides entity level validation support to the DataGrid control. WCR RIA Services takes advantage of the INotifyDataErrorInfo interface to perform validation, too.

Grouping on CollectionViewSource

Silverlight adds capabilities to the CollectionViewSource class by enabling grouping over bound data within a Silverlight DataGrid control. A DataGrid's ItemsSource can be bound to the CollectionViewSource and you can apply a grouping to it. The following code adds 2 new groups to the CollectionViewSource. When it is bound to the DataGrid, the DataGrid will group the items using the GradeLevel and Gender groups.

XAML

```
<UserControl.Resources>
    <CollectionViewSource x:Key="cvs">
        <CollectionViewSource.GroupDescriptions>
            <PropertyGroupDescription PropertyName="GradeLevel" />
            <PropertyGroupDescription PropertyName="Gender" />
        </CollectionViewSource.GroupDescriptions>
    </CollectionViewSource>
</UserControl.Resources>
<Grid x:Name="LayoutRoot">
    <DataGrid ItemsSource="{Binding Source={StaticResource cvs}}" />
</Grid>
```

Editable CollectionView IEditableCollectionView

When a CollectionViewSource is used as the ItemsSource of both a Silverlight DataForm and a DataGrid, the DataForm can use its Add button to add items to the collection. Also, the item can be edited, cancelled, or committed. The item also appears in the DataGrid, which the CollectionView keeps in sync with the DataForm. You can cancel the editing an item in the DataGrid by hitting the escape key twice while editing a field. Both the DataGrid and DataForm perform entity-level validations whenever CommitEdit() is called, giving the user immediate feedback.

XAML

```
<UserControl.Resources>
    <CollectionViewSource x:Key="cvs"/>
</UserControl.Resources>
<Grid x:Name="LayoutRoot">
    <DataGrid ItemsSource="{Binding Source={StaticResource cvs}}" />
    <DataForm ItemsSource="{Binding Source={StaticResource cvs}}" />
</Grid>
```

Binding to String Indexers

Silverlight allows targets to be bound to string indexers. This means objects can define custom sets of properties and bind them to UIElements. An object that requires custom properties would expose an

indexer that takes a string. This can be done either on the object itself or on a dictionary property. Each entry in the dictionary property represents a custom property. The key in the dictionary describes the name of the property and the value corresponding to that key represents the value of that property. The binding engine supports binding to string indexers, which enables binding to such a dictionary. The XAML below demonstrates how to bind to a string indexer.

XAML

```
<TextBox Text="{Binding Path=CustomProperties[Nickname]}"/>
```

The DataContext is an Employee class that has a property named CustomProperties. CustomProperties contains a dictionary of keyed name value pairs.

C#

```
var emp = new Employee() { FirstName = "John" }
emp.CustomProperties.Add("Nickname", "Johnny");
this.DataContext = emp;
```

The definition for the Employee class shows the CustomProperties property defined as a Dictionary.

C#

```
public class Employee : INotifyPropertyChanged
{
  private Dictionary<string, Object> _customProps;
  public Dictionary<string, Object> CustomProperties
  {
    get
    {
      if (_customProps == null)
        _customProps = new Dictionary<string, object>();
      return _customProps;
    }
  }
}
```

TargetNullValue

Silverlight 4 introduces a new binding extension property for TargetNullValue. The TargetNullValue binding extension property applies its value to the target when the source value of the binding operation is Null. The example below will display 0 when the UnitsInStock bound property is Null.

XAML

```
<TextBox Text="{Binding Path=UnitsInStock, Mode=TwoWay, TargetNullValue=0}" />
```

StringFormat

In Silverlight 3, values were often formatted by creating a converter class that implements the IValueConverter interface. Silverlight 4 introduces the StringFormat binding extension property that

formats a value using either a predefined format or a custom format without the need for a converter class. The examples below show how to format a bound value to a currency and to a percentage, respectively, using the StringFormat feature.

XAML

```
<TextBox Text="{Binding Path=UnitPrice, Mode=TwoWay, StringFormat=C}" />
<TextBox Text="{Binding Path=Discount, Mode=TwoWay, StringFormat=P}"/>
```

You can also set the StringFormat to a custom format such as a date format. The example below shows an order date formatted as MM-dd-yyyy (for example: 11-18-2009).

XAML

```
<TextBox Text="{Binding Path=OrderDate, Mode=TwoWay, StringFormat='MM-dd-yyyy'}"/>
```

FallbackValue

The FallbackValue binding extension displays a value when the binding operation is unsuccessful. In the example below, the TextBox attempts to bind to a property ManagesEmployees. Assuming that the DataContext is set to a Manager class instance, this property may return a value. However if the DataContext is set to an Employee object that does not have a ManagesEmployees property, the FallbackValue will be displayed instead.

XAML

```
<TextBox
    Text="{Binding Path=ManagesEmployees, Mode=TwoWay, FallbackValue=N/A}" />
```

Databinding Support for Dependency Objects

Silverlight introduces the ability to bind properties on a DependencyObject (DO) and not just on FrameworkElements. For example, in Silverlight you can bind the rotation angle of a RotateTransform to a Slider control using the following XAML:

XAML

```
<Canvas Width="100" Height="100"
        RenderTransformOrigin="0.5, 0.5" Background="#FF2B6092">
    <Canvas.RenderTransform>
        <RotateTransform Angle="{Binding ElementName=slider, Path=Value}" />
    </Canvas.RenderTransform>
</Canvas>
<Slider x:Name="slider" Height="20" Margin="0,225,0,55" Minimum="0"
        Maximum="360" />
```

The following example shows how to bind the plane projection rotation angles to a series of Slider controls using the new DependencyObject binding feature:

XAML

```
<Canvas Width="100" Height="100"
RenderTransformOrigin="0.5, 0.5" Background="#FF2B6092">
```

```
            <Canvas.Projection>
                <PlaneProjection RotationX="{Binding ElementName=sliderX,Path=Value}"
                                 RotationY="{Binding ElementName=sliderY,Path=Value}"
                                 RotationZ="{Binding ElementName=sliderZ,Path=Value}"
                />
            </Canvas.Projection>
        </Canvas>
        <Slider x:Name="sliderX" Height="20" Margin="0,10,0,55"
            Minimum="0" Maximum="360" />
        <Slider x:Name="sliderY" Height="20" Margin="0,30,0,35"
            Minimum="0" Maximum="360" />
        <Slider x:Name="sliderZ" Height="20" Margin="0,50,0,15"
            Minimum="0" Maximum="360" />
</Canvas>
```

Also, Silverlight includes a DependencyObjectCollection which provides a way for third parties to define collection properties that contain DependencyObjects and have properties on those DependencyObjects in binding operations.

XPath Support for XML

Silverlight provides the ability to parse XML using a variety of techniques including LINQ to XML and XPath support. XPath support, available in Silverlight 4's SDK in the System.Xml.XPath.dll, is new to Silverlight 4. It is valuable for those who need to parse XML and already have extensive XPath libraries.

ObservableCollection<T> Constructor Accepts IList and IEnumerable

Silverlight 4 also can take advantage of new constructor overloads that allow it to initialize an ObservableCollection<T> from an IEnumerable or IList. The example below shows a List<string> used to initialize the ObservableCollection<string>.

C#

```
List<string> ColorList = new List<string> { "red", "blue", "yellow" };
ObservableCollection<string> ColorOC = new
            ObservableCollection<string>(ColorList);
```

Isolated Storage Enhancements

Isolated Storage is ideal for storing data on the local client where the Silverlight application can read and write to it. Performance improvements have been made to make accessing data in Isolated Storage faster.

Managed Extensibility Framework (MEF)

The Managed Extensibility Framework (MEF) enables you to build modularized applications whose components (Parts, as MEF calls them) can be added in an incrementally. MEF allows development teams to attach new functionality after the application has been deployed, including while the application is running. MEF allows you to deploy your application in multiple Silverlight applications

(XAP) files. This allows the XAP files to be dynamically downloaded at runtime and greatly reduces the startup time of the application. MEF allows applications to be context-aware where the available parts change based on the state of the application. It also improves the general maintainability of your Silverlight applications by greatly reducing coupling between its components. MEF can be used anywhere, anytime within Silverlight development, including within elements created in XAML such as a custom user control. MEF can also be integrated into existing Silverlight applications without the need to rewrite them.

The code sample shows how to extend the Silverlight DataGrid to make it an extensible grid. In the grid, the CompositionInitializer.SatisfyImports static method allows the DataGrid to be composed by MEF through an ambient container. When SatisfyImports is called, a container is created that contains all the exports in the current XAP file. The [ImportMany] attribute tells MEF that it should supply all IGridExtension exports to the grid.

C#

```
public class ExtensibleDataGrid : DataGrid
{
    public ExtensibleDataGrid()
    {
        CompositionInitializer.SatisfyImports(this);

        foreach (IGridExtension extension in Extensions)
            extension.Initialize(this);
    }

    [ImportMany]
    public IEnumerable<IGridExtension> Extensions { get; set; }
}
```

This grid will receive all available IGridExtension exports at runtime. MEF discovers the extensions, creates them, and provides them to the grid. Below is an example of a simple grid extension that makes all fonts in the grid bold. The export attribute tells MEF that the BoldExtension class is a part that offers an export of type IGridExtension.

C#

```
public interface IGridExtension
{
    void Initialize(DataGrid grid);
}

[Export(typeof(IGridExtension))]
public class BoldExtension : IGridExtension
{
    public void Initialize(DataGrid grid)
    {
        grid.FontWeight = FontWeights.Bold;
    }
```

```
}
```

MEF allows you to keep your application decoupled so you can import and export parts as needed. This is ideal for a plug-in model or for simply keeping loosely coupled architecture. MEF can also dynamically load XAP files on demand, which is a great when you have a large application where the XAP size is larger than ideal. The code sample below loads the GridExtensions.xap. Any XAP file can be loaded as long as it is within the same ClientBin folder as the hosting XAP file. Once downloaded, the parts in the XAP are immediately available for import into the application. This is ideal when you have a large modular application.

C#

```
private void LoadGridExtensions()
{
    var catalog = new DeploymentCatalog("GridExtensions.xap");
    catalog.DownloadAsync();
    catalog.DownloadProgressChanged +=
        new EventHandler<DownloadProgressChangedEventArgs>(
        catalog_DownloadProgressChanged);
    catalog.DownloadCompleted +=
        new EventHandler<System.ComponentModel.AsyncCompletedEventArgs>(
        catalog_DownloadCompleted);
}
```

SDK Enhancements

Assemblies and namespace support have been added to Silverlight 4 to include MEF, frame and navigation refresh enhancements, and the System.Numerics.dll.

Microsoft Silverlight 4 Tools for Visual Studio 2010

This package is an add-on for Visual Studio 2010 that provides tooling for Microsoft Silverlight 4 and WCF RIA Services. It can be installed on top of either Visual Studio 2010 or Visual Web Developer 2010 Express. It extends the existing Silverlight 3 features and multi-targeting capabilities in Visual Studio 2010 to also support creating, designing, building and debugging Silverlight 4 applications. The tooling supports:

- Elevated trust settings
- Out of Browser window and chrome settings
- Streamlined F5 for out of browser applications

The tooling also includes design time support for WCF RIA Services with the following highlighted features:

- Implicit Styles
- Databinding enhancements

    - StringFormat
    - NullableValue
- Validation
    - IDataErrorInfo
    - INotifyDataErrorInfo
- ICommand, CommandParameter
- RichTextBox, Viewbox and WebBrowser controls
- Creating RTL pages

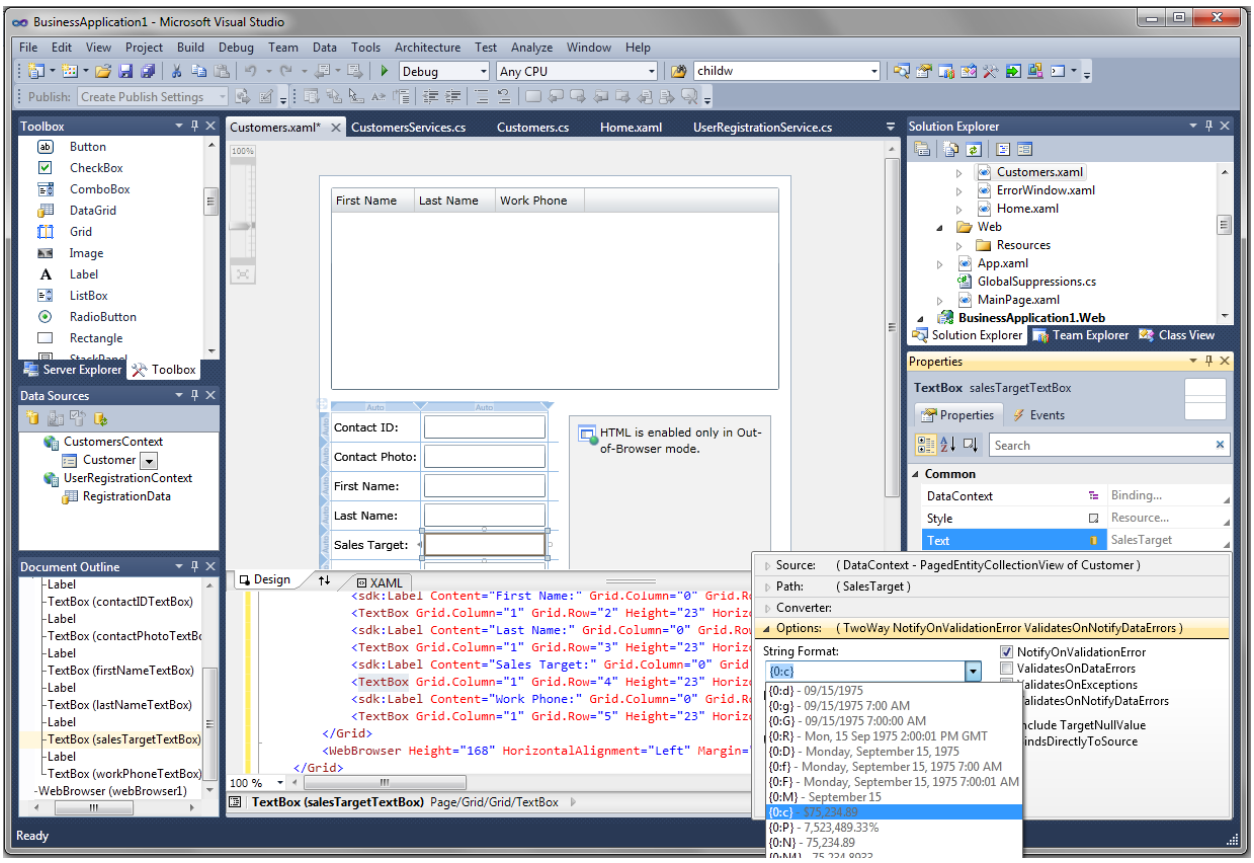

Figure 9

Visual Studio 2010 Tooling Support for Silverlight 4

WCF RIA Services

Overview of WCF RIA Services in Silverlight 4

Silverlight can take advantage of WCF RIA Services and its features to help build business applications. WCF RIA Services makes n-tier development as simple as traditional two-tier development by including enterprise-class networking and data access for building n-tier applications with transactional support and data paging. WCF RIA Services allows you to expose an object model on the server through an optimized .NET to .NET binary format as well as a set of open extensions to the ATOM formatted known as OData and an open JavaScript Object Notation (JSON) format to Silverlight application. It offers support for including metadata on the object model to describe validation requirements such as required fields and checking for valid ranges. It also supports custom validation for a single property or across an entire class. In addition, WCF RIA Services has features to assist with change tracking on the client in Silverlight, user authentication, and personalization. All of these features and others help WCF RIA Services make it easier to build business applications with Silverlight.

Microsoft WCF RIA Services simplifies the traditional n-tier application pattern by bringing together the ASP.NET and Silverlight platforms using the power of WCF for communication. WCF RIA Services provides a pattern to write application logic that runs on the mid-tier and controls access to data for queries, changes and custom operations. It also provides end-to-end support for common tasks such as data validation, authentication and authorization based on roles by integrating with Silverlight components on the client and ASP.NET on the mid-tier.

Key Problems Solved

Currently building a Rich Internet Application (RIA) involves a fair amount of plumbing. For a developer familiar with a web application or a 2-tier smart client application, there are additional concepts to deal with including managing state on the presentation tier, working with an asynchronous model and providing good validation capabilities. In addition to writing application logic specific to your domain, you have to think of the two parts of your application spread across tiers and manage the corresponding projects together.

Key Classes: DomainService and DomainContext

WCF RIA Services provides framework and tooling support for some of the common RIA patterns. A RIA developer authors a DomainService to define the application logic on the mid-tier that operates on a set of data classes called entities. RIA Services then generates the corresponding client-side entity proxies and DomainContext class on Silverlight. The generated classes allow a developer to retrieve data, work with it with rich binding and change tracking capabilities and submit a unit of work corresponding to application tasks completed by the end user of the application. The two parts of the application are weaved together in a single Visual Studio solution to simplify iterative development and testing. In addition, a set of common tasks such as validation, authentication and authorization are pre-plumbed and simplified through declarative attributes.

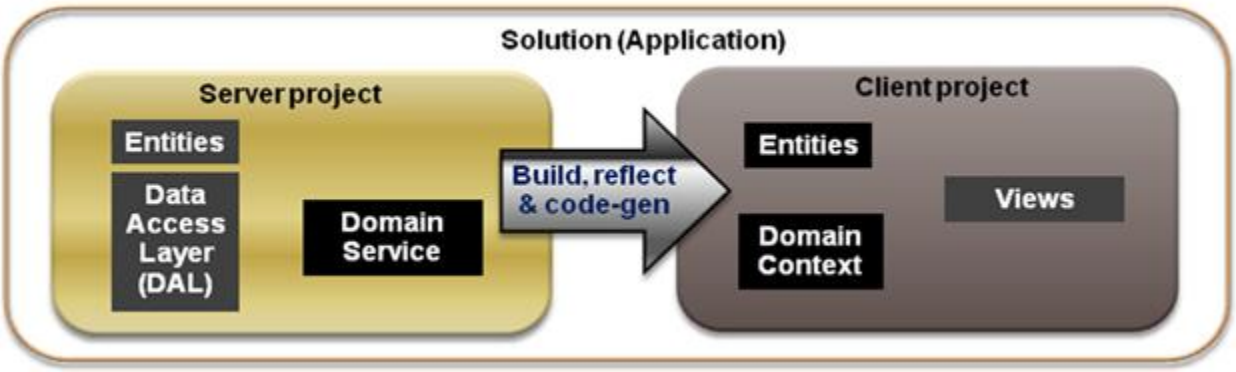

Figure 10

WCF RIA Services in an Architecture

Entities act as containers or data and associated validation information specified as attributes or external metadata. They are obtained via Data Access Layer (DAL) of your choice. Examples include Entity Framework, LINQ to SQL, web services, DataReader, or DataSet.

DomainService Operations

The **DomainService** provides higher-level programming constructs by supporting a set of methods that can be used for authoring arbitrary application logic with entities. It also supports LINQ-enabled query methods that allow the presentation tier to formulate flexible queries while executing them most efficiently in the data tier (database or a web-service).

The DomainService supports data modifications through insert, update, delete methods for enabling the "unit of work" pattern and using transactions supported by the Data Access Layer (DAL). It can also contain a set of named update methods to capture change logic specific to your domain such as approving an expense report or cancelling an order.

When network load is a concern, server throttling can be achieved through regulating the per request load on the database server by specifying maximum result limit on a query method.

Endpoints

Endpoint configuration support is included in WCF RIA Services. By default, only binary endpoint will be exposed but we have made it really easy to add additional/different endpoints. OData endpoints are also supported in WCF RIA Services. You can point PowerPivot to a DomainService OData endpoint and analyze data by calling the appropriate query method. This is the first step in lighting up OData with WCF RIA Services.

DomainContext Generation

On the Silverlight presentation tier the DomainContext class is generated for a DomainService. It enables asynchronous calls to the DomainService query methods to load data into EntityCollections (bindable collections of entities). The relationships between entities are also maintained and tracked on the client to significantly simplify the use of rich data models. When the user completes a logical unit of work, the client part of the application can submit the changes in one shot to the mid-tier. The mid-tier then

validates the submitted entities and operations and invokes the developer-authored DomainService methods.

Authentication and Authorization

Authentication and role-based authorization attributes may be applied to the DomainService methods to enforce restricted access to the privileged data and operations on the mid-tier. WCF RIA Services enforces the authentication and authorization requirements and ensures that only permitted users/roles are allowed access. The **AuthorizationContext** class enables you to have more options for implementing custom authorization rules including entity-based authorization.

Validation and MetaData

Validation metadata can be applies to entities, their members, DomainService operations and their parameters. WCF RIA Services copies the validation rules to the client (unless they are designated as server-only rules). It also provides rich validation experience in the UI. Validation is also performed on the client for early detection of violations. The server independently enforces all validations when changes are submitted.

WCF RIA Services includes validation support based on Silverlight 4's INotifyDataErrorInfo interface. This eliminates the former technique of throwing exceptions in property setters to handle validaiton. Instead, through INotifyDataErrorInfo support WCF RIA Services enables metadata driven validation as well as some server only validation scenarios.

DomainDataSource

The task of rich databinding and operations like filtering, sorting and paging is significantly simplified by the DomainDataSource (DDS). It can be used programmatically or through drag-and-drop from the **Data Sources** window for a rapid application development (RAD) experience. The DDS also supports the ICommand interface in Silverlight 4 for executing commands. DDS also has support for Load, SubmitChanges, RejectChanges, and integrates very nicely into binding in the Visual Studio 2010 designer.

WCF RIA Services Toolkit

Similar to the Silverlight Toolkit, the WCF RIA Services Toolkit will have preview and experimental components. This provides a vehicle to ship components with lower overhead and get feedback while keeping the more mature components in the product.

Performance and Reliability

Key improvements have been made to optimize performance across all areas of WCF RIA Services. One such improvement is with optional count queries which are only executed when needed. WCF RIA Services has been enhanced to provide increased resilience, especially with server side components.

Notes

WCF RIA Services continues to be a part of Silverlight 4 Tools. The MIX 2010 build of WCF RIA Services will continue to support **bin** deployment and medium trust scenarios. It also supports server-only installation option, without the development tools, if desired. LINQ to SQL support and ASP.NET DomainDataSource (but not the Silverlight DomainDataSource) has moved to the toolkit

The build of WCF RIA Services released at MIX 2010 will require .NET 4, Visual Studio 2010, and Silverlight 4. The PDC 2009 drop of WCF RIA Services is the last build that is targeted purely for developing for .NET 3.5 SP1, Visual Studio 2008, and Silverlight 3. The newer WCF RIA Services drops significantly benefit from and rely on the work done in .NET 4, Visual Studio 2010, Silverlight 4. This is a cost-benefit tradeoff in favor of longer term needs over near term expediency.

Empowering Richer Experiences

Hardware Accelerated PlaneProjection

The PlaneProjection can now be hardware accelerated like the other transform types (Scale, Skew, Rotate, and Translate) in Silverlight.

Word Based Text Trimming (trailing ellipses)

The TextBlock has a new TextTrimming property that accepts either of the TextTrimming enumerator values of None or WordEllipsis. When this property is set to WordEllipsis and the text in the TextBlock exceeds the visible limit, the text appears truncated with a trailing three ellipsis. The example below shows the XAML for two TextBlocks with the same text. The first TextBlock sets the TextTrimming to WordEllipsis while the second TextBlock does not.

XAML

```
<TextBlock Height="23" HorizontalAlignment="Left" VerticalAlignment="Top"
    Text="The quick brown fox jumped over the tall white fence"
    TextTrimming="WordEllipsis" Width="120" />
<TextBlock Height="23" HorizontalAlignment="Left" VerticalAlignment="Top"
    Text="The quick brown fox jumped over the tall white fence"
    Width="120" />
```

When the application runs, the TextBlock controls appear as shown below:

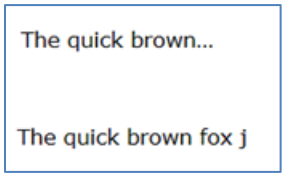

Figure 11
TextTrimming with WordEllipsis

Implicit Styles

Silverlight 4 introduces new styling features that allow you to create a style as a resource that can be used implicitly by all elements of a target type. This allows application developers to customize the look across multiple instances of a control and modify the appearance of these control instances by changing the implicit style.

An implicit style can be created by creating a style, omitting the x:Key name, and setting the TargetType. The XAML below shows an implicit style for all Button controls that makes the foreground red. When this implicit style exists for the Button control, it will be applied automatically without having to set the style to each button.

XAML

```
<Style TargetType="Button">
<Setter Property="Foreground" Value="Red" />
</Style>
```

Implicit styles can be used to declare a style and apply it to a particular control as shown above or to style commonly used properties such as FontFamily, FontSize, Foreground (font color), and Background. If you create an implicit style at the application level, whether embedded in the App.xaml or in a resource dictionary, the style will be applied to all controls that match the target type within your application.

MouseWheel Support

Silverlight has support for responding to mouse wheel events on controls where scrolling is appropriate. The Silverlight ScrollViewer automatically handles the MouseWheel event so the ListBox, ComboBox, TextBox, and ScrollViewers automatically scroll. For example, when the mouse wheel scrolls over a ListBox, the ListBox begins to scroll. Silverlight can also use MouseWheel events to perform custom functionality. For example, you can configure the MouseWheel event so that when a user turns the mouse wheel the application zooms in and out using a scale transform.

The MouseWheel event exists on all UIElements. This event fires when the mouse wheel is turned while the mouse pointer is over a UIElement. The MouseWheelEventArgs expose a Handled property. Since this event occurs synchronously it can be marked as handled to prevent the HTML event from being raised. The Delta property of the MouseWheelEventArgs indicates the relative change that occurred in the mouse wheel rotation. A positive Delta indicates a forward turn of the mouse wheel while a negative Delta indicates a backward turn.

The snippet below demonstrates event handler coding for the MouseWheel event. This handler applies the turns in the mouse wheel to a slider control.

C#

```
private void OnMouseWheel(object sender, MouseWheelEventArgs e)
{
    if (e.Delta > 0)
        slider_X.Value += slider_X.LargeChange;
    else
        slider_X.Value -= slider_X.LargeChange;
}
```

Right Mouse Click

Silverlight 4 adds the ability to right click a visual element to perform custom actions such as displaying a menu that provides contextual commands or invoking commands for a Silverlight game. The MouseRightButtonDown and the MouseRightButtonUp events are now exposed on all UIElement controls. These events allow you to handle the right-mouse-click events and display a custom context menu instead of the default Silverlight menu. The events are routed to a particular element and use the same MouseButtonEventArgs that the MouseLeftButtonUp and MouseLeftButtonDown events use so you can

determine which element was affected and where on that element a right mouse button event occurred in order to show a context menu at that location.

When the MouseRightButtonDown event is handled and the MouseButtonEventArgs.Handled property is set to true, the default Silverlight configuration menu will not be displayed. The MouseButtonEventArgs.StylusDevice.DeviceType property will report the source device type for the event such as mouse, stylus, or touch.

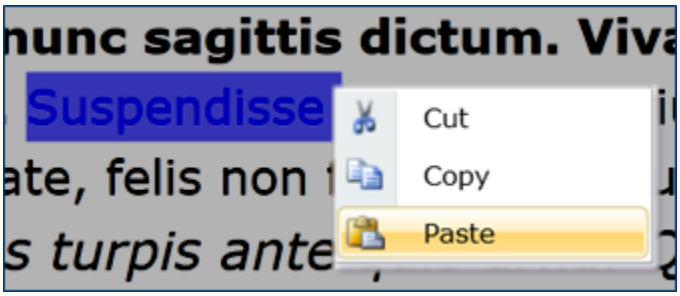

Figure 12

Right Click Displaying a Custom Context Menu

The code example below shows how to implement the right click functionality and display a custom menu control.

C#

```
public MainPage()
{
    InitializeComponent();
    mediaElement.MouseRightButtonDown +=
        new MouseButtonEventHandler(MouseRightButtonDownHandler);
    mediaElement.MouseRightButtonUp +=
        new MouseButtonEventHandler(MouseRightButtonUpHandler);
}

private void MouseRightButtonDownHandler(object sender, MouseButtonEventArgs e)
{
    e.Handled = true;
}

private void MouseRightButtonUpHandler(object sender, MouseButtonEventArgs e)
{
    var menu = new MyCustomMenuControl(mediaElement);
    menu.Show(e.GetPosition(LayoutRoot));
}
```

Programmatic Clipboard Access

Silverlight 4 adds the ability to programmatically access the clipboard to format and modify data during copy, cut, and paste operations. To copy data from a Silverlight application to the clipboard use the Clipboard.SetText method:

C#

```
Clipboard.SetText(rtb.Selection.Text);
```

Once the text is copied to the clipboard, it can be retrieved from the clipboard using the Clipboard.GetText method:

C#

```
rtb.Selection.Text = Clipboard.GetText();
```

This effectively performs a copy and paste operation. To perform a cut operation, simple use the Clipboard.SetText method and set the source of the cut operation to an empty string.

C#

```
Clipboard.SetText(rtb.Selection.Text);
rtb.Selection.Text = "";
```

The SetText and GetText methods of the clipboard object can only be invoked in the context of a user initiated action.

When an attempt is made to programmatically access the clipboard for the first time, Silverlight will prompt the user to request permission.

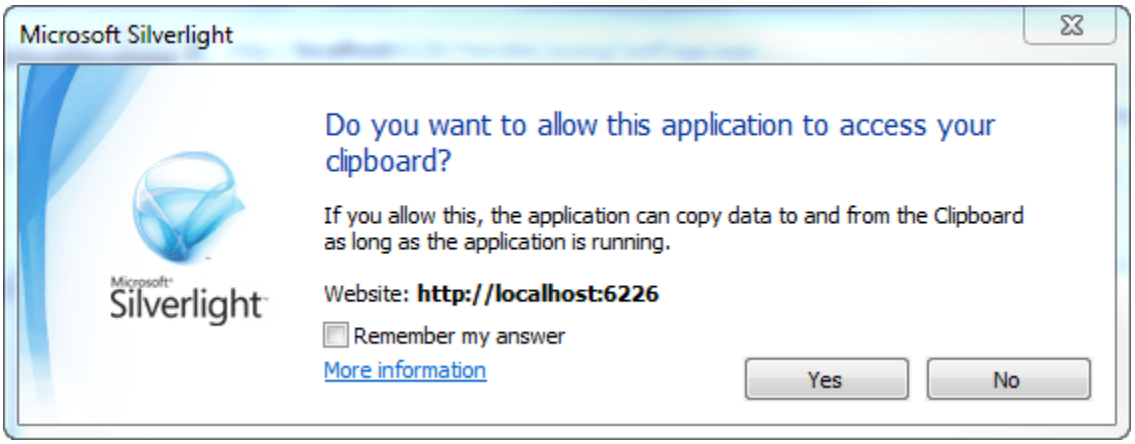

Figure 13

Clipboard Access Prompt

Silverlight as a Drop Target

Silverlight now supports being a drop-target for the common drag-and-drop practice of clicking on a virtual object and dragging it to a different location or onto another virtual object. Both Windows and Mac platforms support this feature. Silverlight only supports dragging of file(s) from local client and dropping them onto a Silverlight application. It does not support dragging of folders (directories). If multiple files need to be dragged, you can select those files and drag them as a group. You must forward the relevant DOM events from HTML-level scripting to the specific API of the control (see attached doc).

The AllowDrop property of the UIElement must be set to true and its Visibility property must be set to Visibility.Visible for the UIElement to be a drop target. Once these are set, the following events can be handled to implement the drop target features:

- DragEnter – Fires when the source is dragged from the local client and it enters the boundary of the UIElement that is the drop target.
- DragLeave – Fires when the file(s) being dragged from the client's local files leaves the boundary of the UIElement
- DragOver – Fires when the input system reports a drag event with this element as the potential drop target.
- Drop – Fires when an object is dropped within a UIElement's boundary.

When handled, these events contain a DragEventsArg that contains a Data property of type IDataObject. The Data property contains information associated with the corresponding drag or drop event. Data will return null when it is accessed outside the context of the OnDrop event and when the item dropped is not a file. When a valid drop has occurred you can access the list of file(s) that were dropped using the GetData method and passing it the DataFormats.FileDrop enumerator. This returns an array of files that can then be accessed for file name, size, and be opened if needed.

C#

```
IDataObject data = e.Data;
if (data.GetDataPresent(DataFormats.FileDrop))
{
       FileInfo[] files = data.GetData(DataFormats.FileDrop) as FileInfo[];
       foreach (var file in files)
       {
                    //FileStream reader = file.OpenRead();
      }
}
```

Handle Drag-and-Drop Events for Macintosh

Drag-and-drop events in Silverlight have some platform-imposed differences. Hosts that run on a Macintosh computer must use different techniques for processing a user-initiated drag-and-drop action. This is because the drag-drop input actions are not reported within the particular API that the Silverlight runtime uses in the Macintosh version of the plug-in.

Drag-and-drop events for applications hosted by Safari on Macintosh require that you forward the relevant DOM events from your own HTML-level scripting to the specific API of the control. This is a requirement because the plug-in model for this architecture does not provide these events to plug-ins such as the Silverlight runtime. The following is an excerpt of the **embed**/**object** tagging you would use to define DOM event handlers for the HTML element that instantiates the Silverlight plug-in:

XML

```
<embed id="AgControl1"
```

```
...
  ondragenter="handleDragEnter(event)"
  ondragover="handleDragOver(event)"
  ondragleave="handleDragLeave(event)"
  ondrop="handleDropEvent(event)"
.../>
```

The following is an example of an event handler that you write in JavaScript to account for event forwarding of drag-drop events on Safari / Macintosh. You should write one handler for each of the drag-related DOM events **dragEnter**, **dragLeave**, **dragOver**, and **drop**. This is the example for **dragEnter**. Note the **getElementById** call to return the Silverlight plug-in instance, which is the object where you forward the HTML DOM level handling to.

JavaScript

```
function handleDragEnter(oEvent) {
  // Prevent default operations in DOM
  oEvent.preventDefault();
  agControl = document.getElementById("AgControl1");
  var flag = agControl.dragEnter(oEvent);
    // If handled, then stop propagation of event in DOM
    if (flag) oEvent.stopPropagation();
}
```

The FireFox architecture for Mac OSX distributions of FireFox restricts access to object-level access to the **drop** event. For this reason, the workaround suggested above for Safari might not function for FireFox.

Webcam and Microphone Support

Silverlight now has webcam and microphone support. Through its device capture API Silverlight can determine which video and audio devices are connected to the computer. The API exposes features to retrieve the devices, prompt the user for permission to access the devices, and use to the devices to capture audio and/or video.

The System.Windows.Media namespace exposes several classes for hardware device capture. The CaptureDeviceConfiguration class's AllowedDeviceAccess property returns true if the user has already granted access to the devices during the current session. The CaptureDeviceConfiguration.RequestDeviceAccess method requests access to the devices from the user. This will display a dialog to the user asking for permission, as shown below.

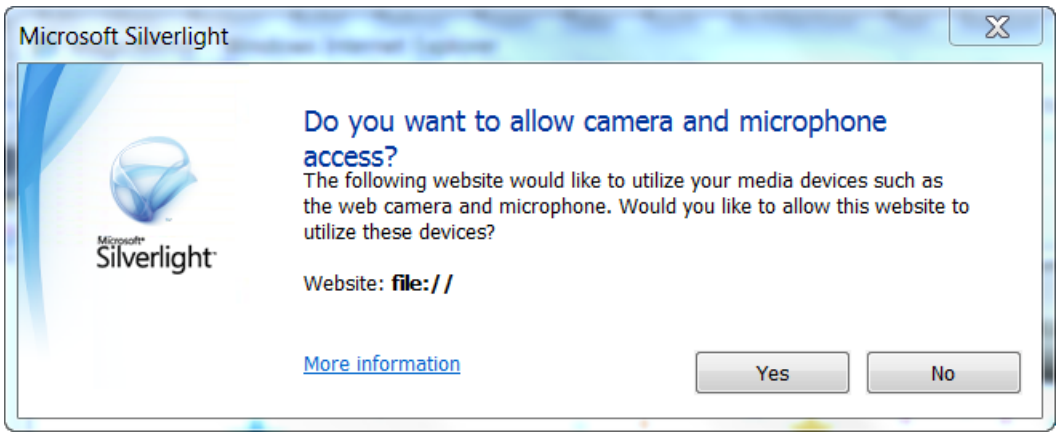

Figure 14

Device Access User Prompt

Once access has been granted to the devices they can be retrieved using the API. The CaptureDeviceConfiguration class's GetDefaultVideoCaptureDevice method gets the default video device and the GetDefaultAudioCaptureDevice method gets the default audio device. Once the video device is retrieved, the video capture can be initiated by creating a new instance of the CaptureSource class and setting its VideoCaptureDevice property to the video device. At this point, the video capture can be started by using the CaptureSource's Start method. The code sample below shows the basic steps to request access to the devices, begin the video capture process, and paint the video using a VideoBrush.

C#

```
if (CaptureDeviceConfiguration.AllowedDeviceAccess
        || CaptureDeviceConfiguration.RequestDeviceAccess())
{
    VideoCaptureDevice vcd =
        CaptureDeviceConfiguration.GetDefaultVideoCaptureDevice();
    AudioCaptureDevice acd =
        CaptureDeviceConfiguration.GetDefaultAudioCaptureDevice();
    if (null != vcd && null != acd)
    {
        cs = new CaptureSource();
        cs.VideoCaptureDevice = vcd;
        cs.AudioCaptureDevice = acd;
        cs.Start();

        VideoBrush videoBrush = new VideoBrush();
        videoBrush.Stretch = Stretch.Uniform;
        videoBrush.SetSource(cs);
        TO_FILL.Fill = videoBrush;
    }
    else
        MessageBox.Show("Error initializing Webcam or Microphone.");
}
```

The CaptureSource class points to specific devices and starts and stops capture. The CaptureDeviceConfiguration class provides the CaptureSource with the configured VideoCaptureDevice and AudioCaptureDevice. Set the default video and audio devices for the Silverlight application by right clicking the Silverlight application and going to the Webcam/Mic tab as shown below.

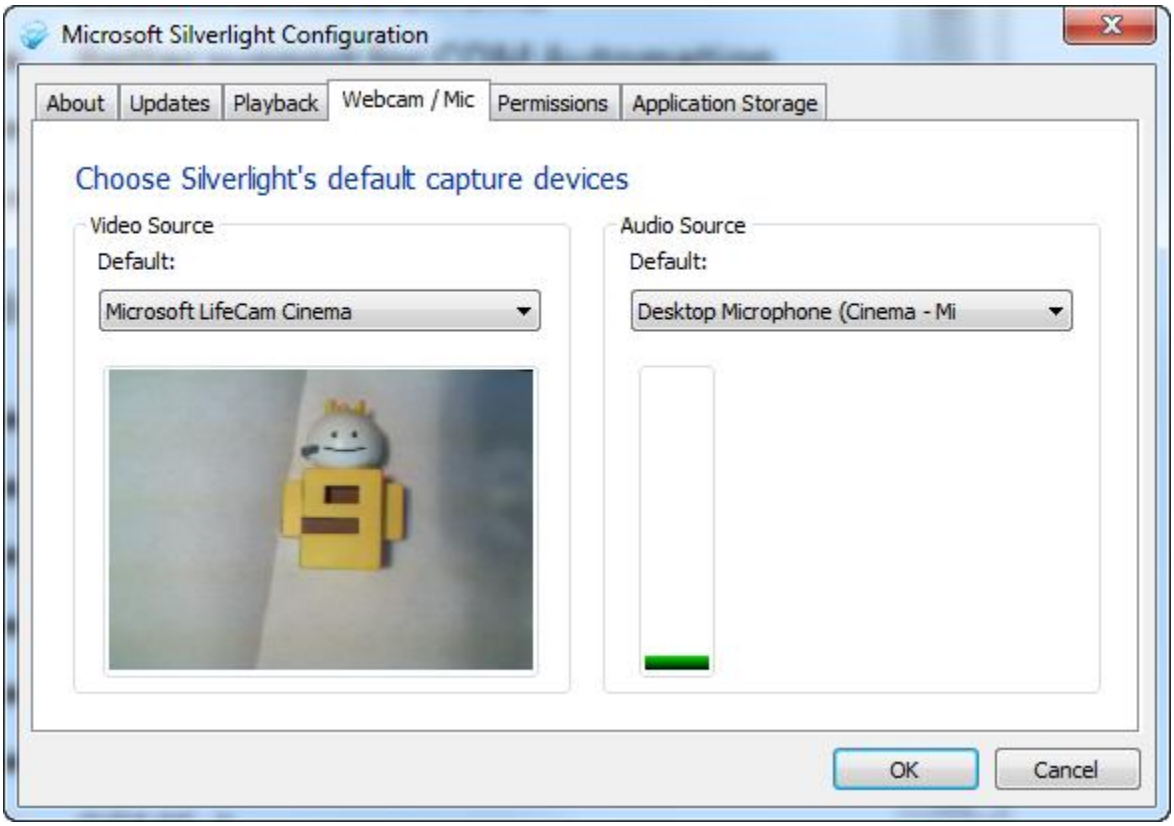

Figure 15

Default Capture Devices

Silverlight 4's configuration window adds new support for webcam and microphone previews, as shown in the figure above. Notice that the figure shows the selected webcam and microphone both displaying a preview of their inputs.

The capture process can be used to grab a bitmap from the webcam. This can be done by first handling the CaptureImageCompleted event and then executing the CaptureImageAsync method. The code sample below shows the capture of an image from the CaptureSource.

C#

```
cs.CaptureImageCompleted += (s, pe) =>
{
    var bmp = pe.Result;
    ImageBrush brush = new ImageBrush();
    brush.ImageSource = bmp;
    brush.Stretch = Stretch.UniformToFill;
    someBorder.Background = brush;
};
```

```
cs.CaptureImageAsync();
```

Webcam and Microphone Permissions

Silverlight 4's configuration window also adds new support for viewing permissions. Figure 16 shows that the webcam and microphone access has been requested and allowed by the user. If a user decides they want to change the permissions they can allow, deny or simply remove the permissions from this tab of the configuration window.

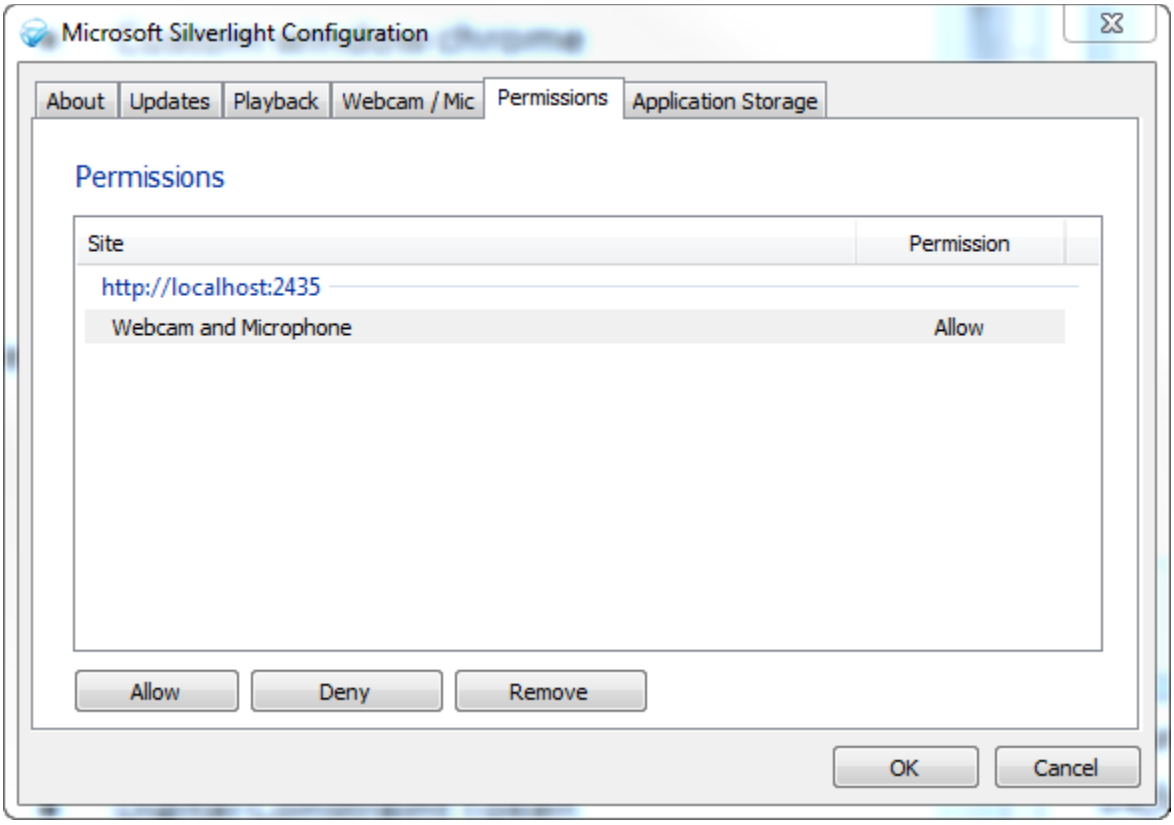

Figure 16

Permissions

CompositeTransform

Silverlight adds a CompositeTransform type that enables the creation of transform groups with one XAML node. This new transform type reduces XAML and collapses all of the types of transform that can be in a TransformGroup into a single node that applies the combined transform in the common order.

For example, this XAML as written in Silverlight 3:

XAML

```
<Rectangle Height="100" Width="100" Fill="Red">
    <Rectangle.RenderTransform>
        <TransformGroup>
            <ScaleTransform ScaleX="2"/>
```

```
            <RotateTransform Angle="45"/>
            <TranslateTransform Y="42"/>
        </TransformGroup>
    </Rectangle.RenderTransform>
</Rectangle>
```

could instead be written like this:

XAML

```
<Rectangle Height="100" Width="100" Fill="Red">
    <Rectangle.RenderTransform>
        <CompositeTransform ScaleX="2" Rotation="45" TranslateY="42"/>
    </Rectangle.RenderTransform>
</Rectangle>
```

Support for all PNG Formats

Silverlight 4 adds support to display all PNG image formats with proper transparency.

Offline Digital Rights Management

Silverlight now allows you to use Digital Rights Management (DRM) in content that your Silverlight applications can access offline. This allows you to better protect and more securely deliver content cross-platform for a variety of scenarios, including streaming, progressive download, rentals, and subscriptions. The Silverlight client can use two forms of DRM protected content: the traditional Windows Media Digital Rights Management 10 (WMDRM 10) and the newer PlayReady that uses the Advanced Encryption Standard (AES).

The next major wave of PlayReady innovation being built into Silverlight focuses on meeting the top media customer ask for the Silverlight DRM client – support for Offline scenarios. The three key business models targeted for this release of the Silverlight DRM client are Rental, Subscription, and Purchase. The Silverlight PlayReady ecosystem has several features that are valuable for these business models.

- PlayReady licenses can be persisted for use by the Silverlight Offline client when accessing PlayReady protected content in the disconnected state (offline).
- Content access policies such as an Expiration Date and an Expire-after-first-play time value support for rental and subscription scenarios
- Managed APIs to request a license given only the content header or a key identifier support for the pre-fetching of licenses
- Support for license chaining can improve the customer experience during subscription renewal by only requiring one license, the root license, to be renewed in order to light up their entire library of subscription content bound to that root license
- PlayReady Domain support to enable content portability and limiting playback to a certain number of PlayReady clients

MP4 Playback Protected DRM

Silverlight 4 can play back H264 and AAC-LC content protected using PlayReady DRM via Silverlight's MediaStreamSource.

WMS Multicast

Silverlight's MediaElement can now take in a Windows Media Station file (.nsc) to enable delivery of media files via multicast, which saves bandwidth on multicast capable networks. Multicast streaming is very similar to traditional over-the-air radio or TV broadcast where a single signal is transmitted and individual receivers "tune in" to the program. WMS Multicast is compatible with existing WMS-based Multicast for easy migration from WMP to Silverlight. It is the lowest cost and the most scalable of all streaming types. In IP Multicast, routers that are multicast-enabled replicate the media stream and broadcast it over a network port where clients can tune in.

Multicast streaming is the most efficient way to deliver audio and video over a network as it scales in a non-linear way. That is, each new viewer does not increase the amount of bandwidth required. In Silverlight 3, Multicast is enabled only through an open source plug-in developed in cooperation with Microsoft by Qumu. Silverlight 4 does not break compatibility with the plug-in although the plug-in is now unnecessary.

Output Protection

Silverlight 4 adds the capability to read Output Protection policies inside of PlayReady licenses and to engage output protections based on those policies. This enables content providers and developers to protect content from the output ports of the Graphics card to Input port of a display device.

Parser Enhancements

The parser in Silverlight 4 adds performance and stability improvements to the parser engine, support for setting order dependent properties, and the XmlnsPrefix and XmlnsDefinition attributes. New valid XAML constructs are now accepted and there is better validation of XAML errors. For example, direct content is now handled properly, making the following XAML behave just as it would in WPF: "<Button>Hello, World!</Button>". In addition, custom dictionaries are now supported in Silverlight XAML (using x:Key to add items).

The new parser provides significant improvements to runtime error reporting in XAML. With the newly rearchitected parser, far more error messages provide useful information about what went wrong and on what line/column the error occurred within the XAML document.

The parser has also added support for XmlnsDefinition, allowing you to eliminate the plethora of xmlns declarations in every XAML document. You can take advantage of intellisense across all of your referenced control libraries and namespaces using a single XML namespace. The Silverlight SDK and Toolkit have been updated to take advantage of this functionality, meaning any toolkit control can be

accessed with the prefix “toolkit:” (or “sdk:” for SDK controls), and only one xmlns declaration at the top of your document.

Support has been enabled for components that implement ISupportInitialize, too. This allows such components to be notified when the parser is done setting properties so that order-dependent property setting can be handled appropriately.

Deep Zoom

Deep Zoom’s code base has been updated with support for hardware acceleration, resulting in increased performance.

Google Chrome Support

Silverlight 4 adds support for Google Chrome browser, in addition to the Safari, FireFox and Internet Explorer.

Private Mode Browsing

Private mode browsing, supported by many web browsers, is now supported by Silverlight 4.

Pinned Full-Screen Mode on Secondary Display

Silverlight 4 now supports pinning an application to full screen mode when the application is not in focus. This is useful when you have more than 1 display monitor and you are watching a video in full screen mode on a monitor and interacting with other applications in another monitor. The following code sets the application to allow full screen when not in focus.

C#

```
Application.Current.Host.Content.FullScreenOptions =
    FullScreenOptions.StaysFullScreenWhenUnfocused;
Application.Current.Host.Content.IsFullScreen = true;
```

Media Updates

Several updates have been made to the media stack in Silverlight 4. Silverlight 4 provides more descriptive MediaSourceStream errors. Instead of seeing 4001 AG_E_NETWORK error for the MediaStreamSource, you will now have the improved support for seeing information about the error that is more relevant to the media pipeline.

Content protection updates are included in Silverlight 4. The MediaSourceStream is able to use the sub sample encryption mechanism for protecting Mp4 content as described in the PIFF specification. Also, there is now support for the CGMS-A analog output protection mechanism.

As an additional layer of security when engaging output protections as required by content providers, the graphics card will be checked to make sure it is appropriately up-to-date.

Digital Constraint Token support exists as a new output protection policy that is based on both the frame quality (High-Def or Standard-Def) as well as the desired protection level. This allows content owners to protect their content under adaptive streaming.

Key Expression Blend Features

Expression Blend has added several fantastic new features. This section highlights some of the key features including a PathListBox which visually changes the way you think about the ListBox control. Also covered are enhancements to integrate MVVM development, new design time data features, binding enhancements, and new behaviors.

Interoperability with Visual Studio 2010

Expression Blend 4 Beta and Visual Studio 2010 use the same project format. Expression Blend 4 Beta can convert a Visual Studio 2008 project for you in order to work with it.

PathListBox Control

The PathListBox is a new control that introduces a new and flexible way to layout multiple items. Maintaining the selection and binding features of the traditional ListBox, the layout of the items is determined by defining one or more UIElements as LayoutPaths. Many properties are made available to customize the positioning and orientation of the items along the paths. Additionally, individual PathListBoxItems may be modified directly to change their appearance or layout.

The image below demonstrates a dataset of years bound to the Items property of a PathListBox and the red arcing path is defined as the LayoutPath for the PathListBox.

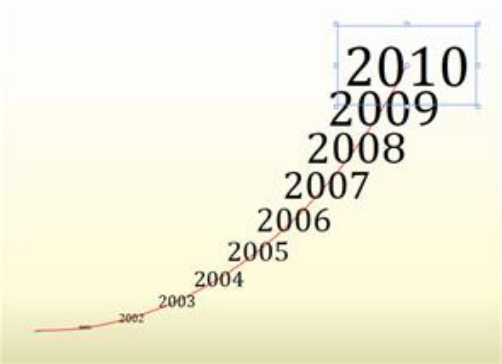

Figure 17
PathListBox Control

Create your own PathListBox by following the steps below:

1. Add one or two paths to your scene. The Path, Ellipse, Rectangle and Shape controls work well as LayoutPaths.
2. Add a PathListBox control.
3. Add items to the PathListBox manually or by databinding. Without a LayoutPath defined, the items will all be positioned in the same spot.
4. With the PathListBox selected, find the Items Layout pane on the Properties Panel. Use the pick widget, with the target icon, to select one or more objects to use as the LayoutPaths.

5. The items of the PathListBox will be arranged along the selected path or paths.

Transition Effects

Transition Effects blend two visuals using a pixel effect over time and are applied in Blend 4 when configuring Visual State transitions. The TransitionEffects dropdown can be found under the “fx” and arrow icon, in the States panel on each transition. If a TransitionEffect is applied to a transition between States, the transition will be visible and animated with other changed properties during the State change.

TransitionEffects available include the following:

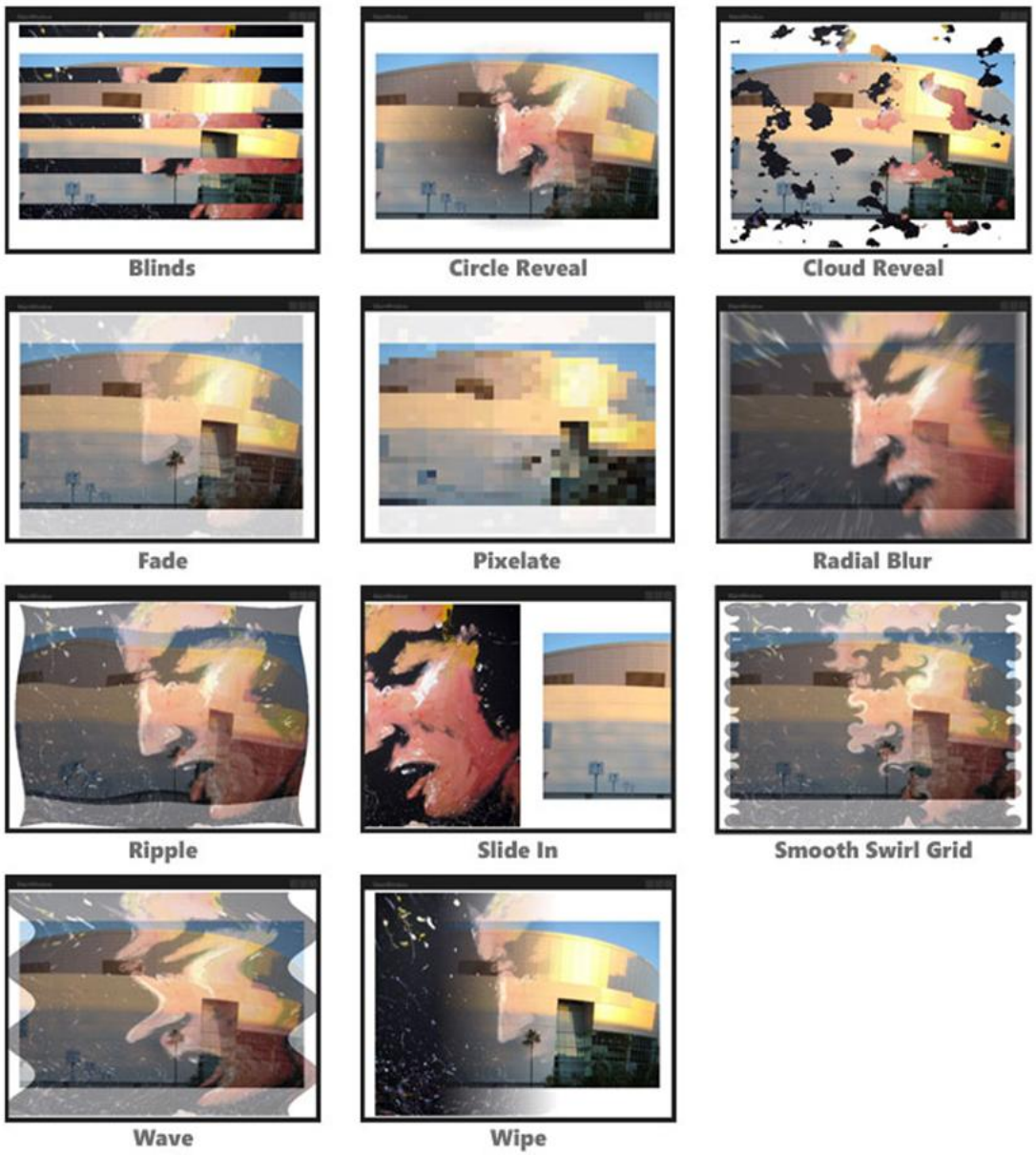

Figure 18

Transition Effects

Conditional Behaviors

Any Action can now be associated with a set of conditions that must be met in order to execute the Action. This means that with Expression Blend 4 Beta you can now build conditional logic into your prototypes and production applications without the need to write code.

Bindable Properties

Properties of behaviors are now bindable in the same way as FrameworkElement properties. This allows behaviors to be more dynamic and aware of the current context. To create a quick sample and see this new feature in action follow the steps below:

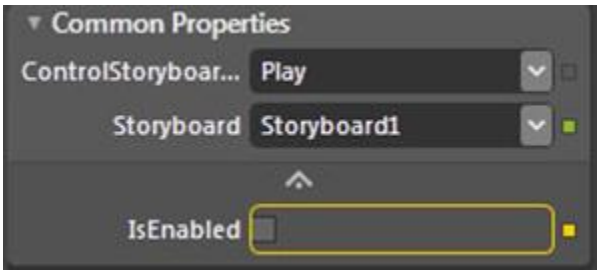

Figure 19

Bindable Behavior Properties

1. Add a Rectangle to your scene.
2. Create a Storyboard animating the Rectangle's position across the screen.
3. Add a ControlStoryboardAction to the Rectangle and set the Storyboard property to the one you just created.
4. Add a Checkbox to your scene.
5. Show the advanced properties in the Common Properties pane
6. Click the Advanced properties button to the right of the IsEnabled property and select the "Element Property Binding…" option.
7. Using the new pick widget, select the Checkbox and from the Create Data Binding dialog select the IsChecked property.
8. Now when you run the project, the Storyboard should only play when you click on the Rectangle and the Checkbox is checked.

New Behaviors

Adding to the original set of eight behaviors, seven new behaviors have been added with a strong focus on interacting with data. Additionally the FluidMoveBehavior has been updated to provide more functionality than before. The new behaviors include the following:

- **CallMethodAction** – Calls a method on the target object. This can be very useful when your control has a custom class set as the DataContext. For example, when using MVVM the View may need to invoke a method on the ViewModel. If the ViewModel is set as the DataContext for the View, the View can easily call a method on the ViewModel using data bindings through the CallMethodAction behavior.
- **DataStateBehavior** – Toggles between two states based on a conditional statement. This is a simple merge of the DataTrigger and the GoToStateAction into a single Behavior.
- **DataTrigger** – Any time the bound property changes, the trigger compares this value to the Value property. If the two values match, the trigger fires.
- **DataStoreChangeTrigger** – Fires when a specified property in the Data Store changes.
- **FluidMoveBehavior** – Enhanced with additional properties to expand its scope beyond a single container. By setting a tag property, an element's current state can be used to dynamically animate discrete properties of the element. FluidMoveBehavior is optimized for two main scenarios; animating list items from one list to another list and animating from a master list to a detail view.
- **FluidMoveSetTagBehavior** – Sets a tag property to act as a pointer that can be referenced by a FluidMoveBehavior instance. In a Master/Detail scenario, the elements of a master list may only need to broadcast their position. The detail view can then be animated from that position using the recorded values.
- **InvokeCommandAction** – Invokes a specified ICommand. This can be very useful when your control has a custom class set as the DataContext. For example, when using MVVM and the ViewModel is set as the DataContext of the View, the View can invoke command on the ViewModel through data bindings and the InvokeCommandAction behavior.
- **SetDataStoreValueAction** – Changes the value of a property and optionally animates the change over a duration of time. Properties are found in the new Data Store explained below.

MVVM Project and Item Templates

The MVVM templates are meant to be a starting point that illustrates some of the designer techniques available in Blend against a data model. They are not meant to teach developers about their architecture. In order to enable designers to work against a “real” data model, the data model needs to have a façade that is reasonably close to a real design pattern: our focus is on the techniques a designer would use to connect to a business data model and connections, rather than the implementation of the data models and backend logic themselves.

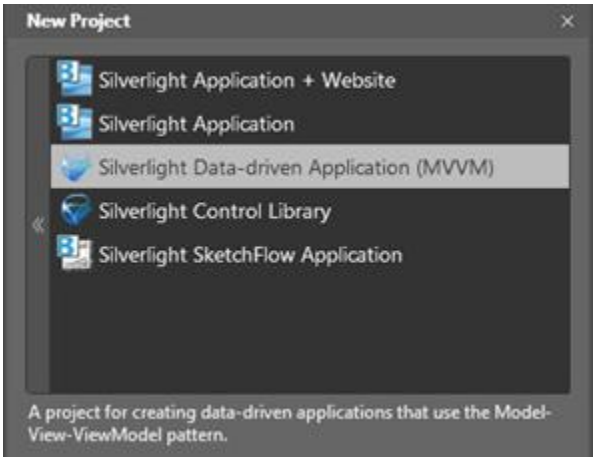

Figure 20

MVVM Templates

Along with the new templates, new behaviors have been added to enable MVVM-based development. The CallMethodAction and InvokeCommandAction behaviors are a valuable asset when working with the interaction between Views and ViewModels.

Design-time data from CLR types

In addition to sample data based on XML, you can now create design-time sample data from your CLR types. Even if a type has no public constructor, or it has properties with no public setter, it can still be made into design-time sample data with the “Create Sample Data from Existing Class” command on the Data panel.

The Data panel provides a view of an object’s DataContext, whether design-time or run-time, and allows you to drag properties and Commands onto the design surface to bind your UI to them.

Design-time ViewModels

If your application uses the MVVM pattern then you can use the “Create Sample Data from Existing Class” command on the Data panel to generate design-time sample ViewModels so that you can continue to design your application in the context of this data.

Moving Beyond the Browser – Sandboxed Applications

Out-of-Browser Windowing Updates

Silverlight 4 has some new Out-of-Browser features including the ability to resize the window programmatically, position the window to start up anywhere on the screen, and set the window's z-order. You can also set the window state to maximized or minimized.

Hosting Web Content within Silverlight Applications

The WebBrowser control allows you to display HTML in the control when running Out-of-Browser. When you use a WebBrowser control in an application that runs inside a browser, a rectangle the same size as the WebBrowser control will display instead of the WebBrowser and its content. There are three ways to load web content inside of the WebBrowser control:

1. The NavigateToString method
2. The Navigate method
3. The Source property

The NavigateToString method accepts a string that represents content (HTML and JavaScript). The content is passed to this method is rendered in the WebBrowser control. The Navigate method accepts a URI that points to the HTML content that you want to render. The URI does not need to be in the same domain as the Silverlight application. The Source property is another way to set the content to a URI.

The following code loads the WebBrowser control **wb** with HTML content when run Out-of-Browser.

C#

```
string content = "<table border=3 cellpadding=3><tr><td><font size=36" +
        "color='red'>Red</font></td><td><font size=36" +
        "color='blue'>Blue</font></td></tr></table>";
wb.NavigateToString(content)
```

Figure 21

Embedded Web Content

The following code loads the www.microsoft.com Web page in the WebBrowser control when run Out-of-Browser.

C#

```
wb.Navigate(new Uri("http://www.microsoft.com"));
```

Figure 22

Loading www.microsoft.com in a WebBrowser Control

WebBrowserBrush

Silverlight adds a new type of brush called an WebBrowserBrush that accepts a WebBrowser control as its source. The SetSource method accepts a WebBrowser control and displays the content using the WebBrowserBrush in another UI Element. The WebBrowserBrush is a brush which means it is non-interactive HTML. For example, a Path, Ellipse, or Rectangle can be painted using the WebBrowserBrush, which gets its source from the www.Microsoft.com Web page. The XAML below defines a Path with an WebBrowserBrush.

XAML

```
<Path Data="M0.5,0.5 L68.247093,0.5 L67.29586,1.0469482 C60.223965,5.343935
55.5,13.120297 55.5,22.000002 C55.5,35.530975 66.469025,46.5 80,46.5 C93.530975,46.5
104.5,35.530975 104.5,22.000002 C104.5,13.120297 99.776031,5.343935
92.70414,1.0469482 L91.752907,0.5 L159.5,0.5 L159.5,57.298336 L159.91597,56.378239
C163.87273,48.166615 172.27461,42.5 182,42.5 C195.53098,42.5 206.5,53.469025 206.5,67
C206.5,80.530975 195.53098,91.5 182,91.5 C172.27461,91.5 163.87273,85.833389
159.91597,77.621765 L159.5,76.70166 L159.5,133.5 L89.70166,133.5 L90.621765,133.91597
C98.833389,137.87274 104.5,146.27461 104.5,156 C104.5,169.53098 93.530975,180.5
80,180.5 C66.469025,180.5 55.5,169.53098 55.5,156 C55.5,146.27461 61.166618,137.87274
69.378235,133.91597 L70.298332,133.5 L0.5,133.5 L0.5,78.752907 L1.0469483,79.70414
C5.343935,86.776031 13.120296,91.5 22.000002,91.5 C35.530975,91.5 46.5,80.530975
46.5,67 C46.5,53.469025 35.530975,42.5 22.000002,42.5 C13.120296,42.5
5.343935,47.223965 1.0469483,54.29586 L0.5,55.247093 z"
```

```
        Stroke="#FF666666" Stretch="Fill" StrokeThickness="0.5"
        StrokeLineJoin="Round">
    <Path.Fill>
        <WebBrowserBrush x:Name="WebBrowserBrush"
                AlignmentX="Left" AlignmentY="Top" Stretch="None"/>
    </Path.Fill>
</Path>
```

You can set the WebBrowserBrush to a WebBrowser control that loads the www.microsoft.com Web page, as shown in the code below.

C#

```
wbBrush.SetSource(wb);
```

Figure 23

Painting a Path with an WebBrowserBrush

Notifications (Toast)

Now Silverlight's Out-of-Browser applications can use the NotificationWindow class to display notifications (also known as toasts). Notifications let you alert the user that a significant event has occurred in the Out-of-Browser application. You can create a new instance of the NotificationWindow, set its height and width, then set its Content property to some XAML content. Finally, to display the notification window, invoke the Show method and pass to it the number of milliseconds it should be displayed, as shown in the sample code below.

C#

```
NotificationWindow notify = new NotificationWindow();
notify.Width = 330;
notify.Height = 75;
TextBlock tb = new TextBlock();
tb.Text = "Sample notification";
tb.FontSize = 24;
notify.Content = tb;
notify.Show(3000);
```

The previous code shows a simple example that displays a basic notification. However, you can also customize the content for a notification to look any way you want. The image below shows a customized notification.

Figure 24

Custom Notification

Window Closing Event

Silverlight 4 adds the ability to discover when an out of browser application's window is being closed. The Closing event offers the opportunity to cancel this operation (unless it is being closed due to the user shutting down or logging off from the system). The following code could be added to handle the Closing event.

C#

```
Window mainWindow = Application.Current.MainWindow;

void SomeMethod()
{
    if (Application.Current.IsRunningOutOfBrowser)
    {
        mainWindow.Closing += MainWindow_Closing;
    }
}
```

The handler for the Closing event (shown below) checks if a CheckBox named canCloseCheckBox is checked. If so and the application is being closed by the user (not by a user shut down or logging off the system), the window is then closed.

C#

```
void MainWindow_Closing(object sender,
    System.ComponentModel.ClosingEventArgs e)
{
    if (canCloseCheckBox.IsChecked == false &&
        e.IsCancelable)
    {
        e.Cancel = true;
    }
}
```

Moving Beyond the Browser – Elevated Trust Applications

Silverlight introduces out of browser application with elevated trust. These applications require user consent at install time and run outside the browser with elevated trust. To enable a trusted application, the developer must first check the “Require elevated trust when running outside the browser” permissions checkbox in the Out-of-Browser settings for the Silverlight project as shown at the bottom of the image below.

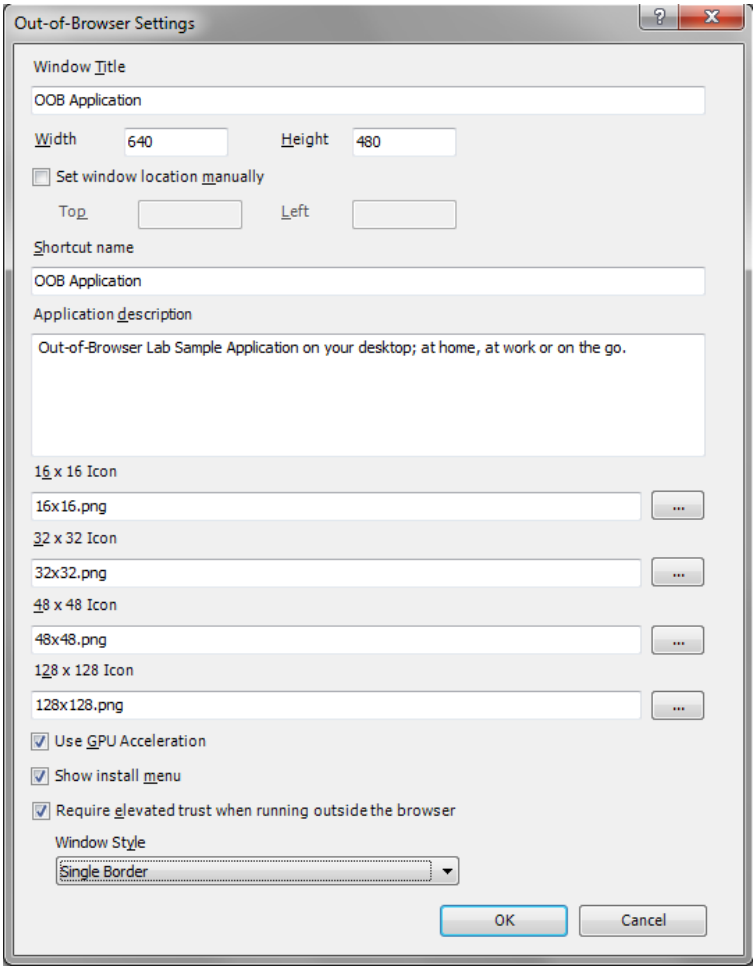

Figure 25

Out of Browser Settings

Setting this checkbox allows the Silverlight application, when installing Out-of-Browser, to prompt the user to request permission to operate as a Trusted Application. You can check if the application is running with elevated permissions using the following code.

C#

```
if (App.Current.HasElevatedPermissions}
```

Native Integration

One feature of a Trusted Application is the ability to use native integration to perform automation operations such as COM Interop with Windows. This feature also has support for IEnumVARIANT; so you can foreach over a collection returned by the Automation server.

You can check if the application is running Out-of-Browser by checking the AppCurrent.IsRunningOutOfBrowser property. You should also check the AutomationFactory.IsAvailable property before using automation.

> When checking the **AutomationFactory.IsAvailable** property, it will also make an internal call to **AutomationFactory.IsRunningOutOfBrowser**. So there is no need to explicitly call both.

If both return true and Microsoft Office is installed, you can create an instance of an object such as Microsoft Word using the AutomationFactory.CreateObject method. Thanks to the new dynamic keyword in .NET 4 that allows you to create an object which is unknown at compile time, you can create an instance of Word and manipulate it. The code below shows how to do this.

C#

```
if (AutomationFactory.IsAvailable)
{
    dynamic word = AutomationFactory.CreateObject("Word.Application");
    word.Documents.Add();
    word.Selection.Paste();
    word.Quit();
}
```

File System Access

Another feature of a Trusted Application allows access to the file system; specifically the Documents, Videos, Pictures, and Music folders under the user's profile. You can specify which folder you want to open by passing one of the Environment.SpecialFolder enumerator values to the Environment.GetFolderPath method. This method returns the path of the folder which can then be used to perform file operations, as shown in the code below:

C#

```
var sampleFile = @"\sample.txt";
var path = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures));
if (System.IO.File.Exists(path + sampleFile))
{
    System.IO.File.Delete(path + sampleFile);
}
StreamWriter sw = File.CreateText(path + sampleFile);
sw.WriteLine("<root>some data</root>");
sw.Close();
```

Cross-Domain Networking Access

An Out-of-Browser Trusted Application can perform cross-domain network calls. This allows the Silverlight client to make calls to a domain other than that in which the Silverlight application is hosted. Twitter.com has a policy file in place that prohibits cross-domain access. Cross-domain calls must be made using the client networking stack, which can be done using the following code.

C#

```
var uriString =
    string.Format(@"http://twitter.com/statuses/friends_timeline/{0}.xml?count=50",
    username);
WebClient request = new WebClient();
WebRequest.RegisterPrefix("http://", WebRequestCreator.ClientHttp);
request.DownloadStringCompleted += new
    DownloadStringCompletedEventHandler(client_DownloadStringCompleted);
request.DownloadStringAsync(new Uri(uriString));
```

Full File Path on Open and Save Dialogs

Silverlight Trusted Applications also allows full access to the file path of the local computer. The OpenFileDialog and SaveFileDialog objects both can allow access to the full file path so you can open, read, and save files as shown in the sample code below.

C#

```
OpenFileDialog dlg = new OpenFileDialog();
dlg.ShowDialog();
var file = dlg.File;
if (file != null)
{
    MessageBox.Show(file.DirectoryName);
    FileStream fs = file.OpenRead();
   //...
}
```

You can access the full path as long as you have access to the location. So if you use the open dialog and pick an image file from c:\Foo\ you will not get the full path. But if you picked it from the Pictures folder, you would.

Sockets Security

The sockets security restrictions do not apply when running a Silverlight 4 Trusted Application.

XAP Signing

Elevated trust out-of-browser applications enable developers to take advantage of platform features that are inaccessible to sandboxed Silverlight applications. You can digitally sign your XAP files to reassure end users of the authenticity of an application’s publisher and that the code’s integrity is intact.

This feature only applies to trusted apps; sandboxed XAPs may be signed but doing so will have no effect on it.

When a user attempts to install an elevated trust out of browser application, the user will be presented with a dialog as shown below.

Figure 26

Unverified Publisher Install Dialog on Windows

Figure 27

Unverified Publisher Install Dialog on Macintosh

A signed XAP would prompt the user with a dialog similar to the following:

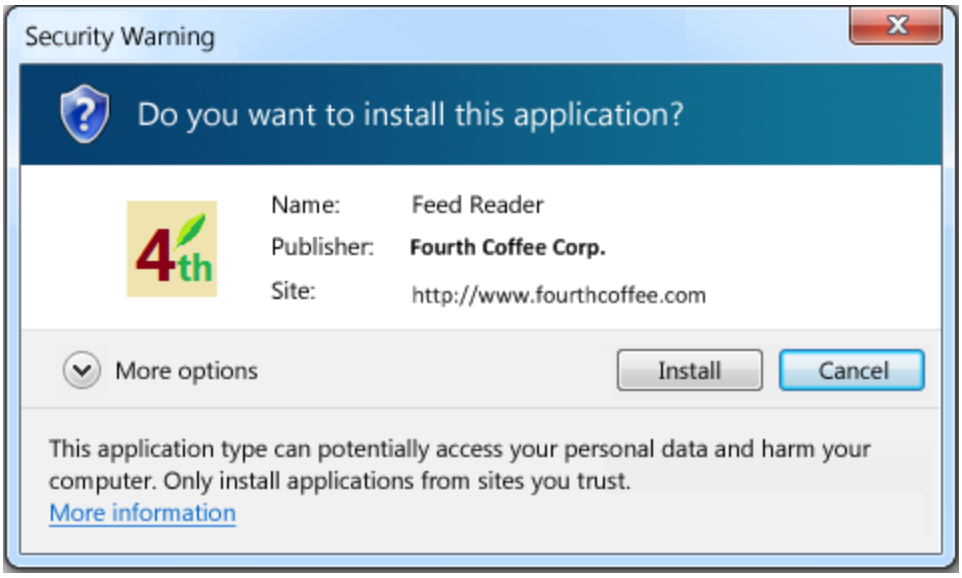

Figure 28

Verified Publisher Install Dialog on Windows

Figure 29

Verified Publisher Install Dialog on Macintosh

XAP signing also affects an elevated trust application's ability to update itself. For an update to be allowed, the installed XAP and the update candidate (the new XAP) must be signed with matching certificates that have not expired.

A XAP can be signed post-build using the SignTool.exe command line tool which is in the Windows SDK, as part of Visual Studio 2010 and a handful of other packages. XAPs must be signed using code signing certificates.

> You can obtain a digitally signed certificate from various publishers. Prices range and most tend to be valid for 1 year before expiring.

To sign a XAP using a test certificate for development purposes, open a Visual Studio Command Prompt and type the following to create a root certificate:

Command Line

```
makecert -n "CN=My Root Certificate Authority" -r -a sha1 -sv
c:\Demo\TestOOBRootCA.pvk c:\Demo\TestOOBRootCA.cer -sr LocalMachine -sky signature
```

When prompted for a password enter a password and write it down so you do not forget it. You'll be prompted to enter the password a few times. Enter the same password each time. Now type the following into the command window and press Enter to create a child certificate that can be used for code signing. It will be signed by the root certificate created earlier.

> We strongly recommend using a password that uses some combination of letters, numbers and special characters.

Command Line

```
makecert -sv c:\Demo\TestOOBCodeSigningCA.pvk -iv c:\Demo\TestOOBRootCA.pvk -n
"CN=Test OOB Crew Code Signing CA" -ic c:\Demo\TestOOBRootCA.cer
c:\Demo\TestOOBCodeSigningCA.cer
```

Enter the password when prompted. Generate a PFX file (contains the password and the private key in one file for convenience). Note that the same password entered earlier is used.

Enter the following into the command window and press Enter:

Command Line

```
pvk2pfx -pvk c:\Demo\TestOOBCodeSigningCA.pvk -spc c:\Demo\TestOOBCodeSigningCA.cer -
pfx c:\Demo\TestOOBCodeSigningCA.pfx -po password
```

Enter the password when prompted. Now that you have a certificate you are ready to sign the XAP. If you purchased a digital certificate you would skip right to the next step where you sign the XAP. Type the following command to sign the XAP:

Command Line

```
signtool sign /v /f c:\Demo\TestOOBCodeSigningCA.pfx /p password
c:\Demo\SilverlightApplication2\SilverlightApplication2.Web\ClientBin\SilverlightAppl
ication2.xap
```

If the XAP was successfully signed you'll see verbiage similar to the following in the command window.

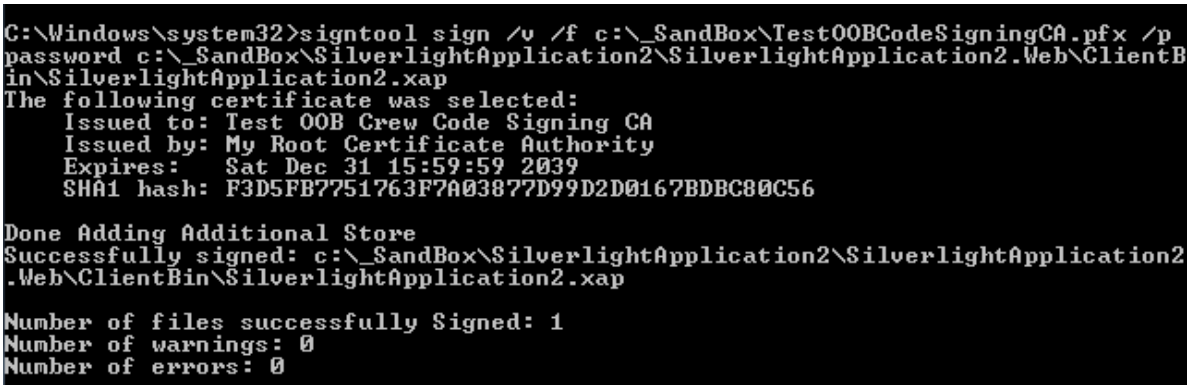

Figure 30

Signing a XAP using signtool.exe

Every time an out of browser Silverlight project with elevated trust is built with Visual Studio 2010, a new XAP is created. This new XAP must be signed once again. For development purposes you can add a post build event and perform the **signtool sign** command to sign the XAP after each build.

Silent Install with SLLauncher.exe

You can perform a silent installation using sllauncher.exe with Silverlight 4. Silent installs are useful for installing a Silverlight application directly on a machine using a command line, .bat file or from a DVD. The sllauncher.exe can be run using the /install switch to point to the location of the XAP file to install. Also, set the /origin switch to the URL where the XAP file is hosted. The /origin switch indicates where the XAP file will get any future updates from. The following example

Command Line

```
"C:\Program Files\Microsoft Silverlight\sllauncher.exe"
    /install:"C:\Demo\OOBDemo.Web\ClientBin\OOBDemo.xap"
    /origin:"http://localhost:49786/ClientBin/OOBDemo.xap"
    /shortcut:desktop+startmenu
    /overwrite
```

Custom Window Chrome

When running a trusted out of browser application the window chrome can be customized. You can change the window chrome by opening the out of browser settings in Visual Studio and selecting the Window Style to be one of the options as shown in the figure below.

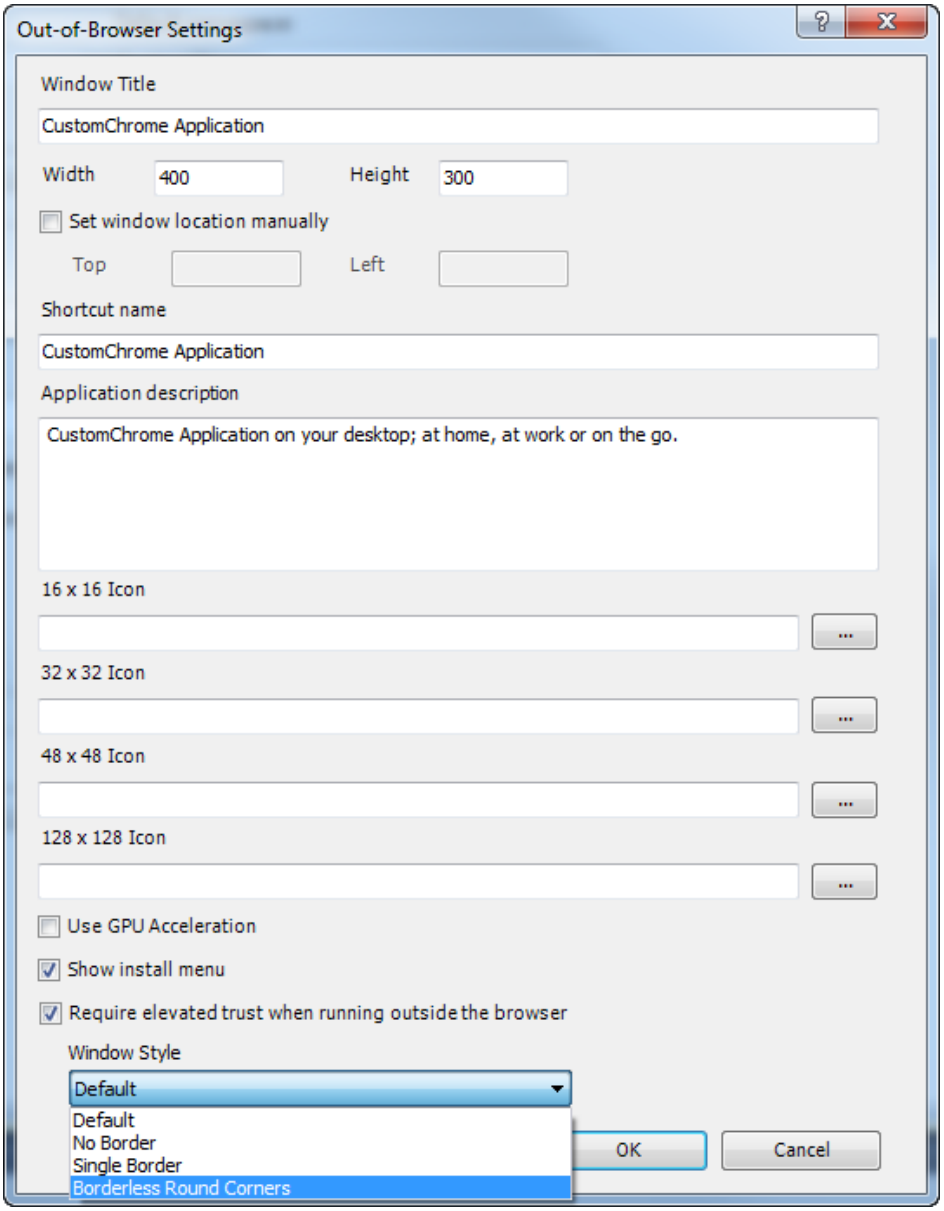

Figure 31

Customizing the Window Chrome

Full Keyboard in Full Screen Mode

A Silverlight Trusted Application enables all keyboard input when in full screen mode. The Content.IsFullScreen property determines whether the Silverlight plug-in displays as a full-screen plug-in or as an embedded plug-in. If you set the IsFullScreen property to true, the Silverlight plug-in displays in full-screen mode.

C#

```
var isFull = this.Host.Content.IsFullScreen
```

When the screen is put into or taken out of full screen mode, the FullScreenChanged event fires. When a Silverlight plug-in displays in full-screen mode, it briefly displays the message "Press ESC to exit full-screen mode". This message alerts the user that the application is now in full-screen mode, and provides information about how to return to embedded mode.

Figure 32

Full Screen Mode Message

The Silverlight plug-in does not support OpenFileDialog and SaveFileDialog in full-screen mode, so you should exit full-screen mode before using these classes. Full-screen mode does not support multi-touch input either.

Silverlight 4 Resources

Tools

There are several resources to help you get up to speed with the Silverlight 4 including videos, screencasts, hands on labs, feature samples, and blog posts from the Microsoft Silverlight teams. I recommend starting with the Channel 9 video since it pulls everything together and provides a great starting point for Silverlight, and then branching out from there depending on your interests. I have listed these below, but before diving into the Silverlight 4, you will need to install some tools. You can find the instructions and tools at this link.

Training Kit

Silverlight TV on Channel 9 – Go behind the scenes at Microsoft with John Papa and learn what the Silverlight product team is dreaming up next. See exclusive interviews with the Silverlight product team, watch how community leaders are using Silverlight to solve real problems, and keep up with the latest happenings with Silverlight. Catch the inside scoop on Silverlight with Silverlight TV!

Hands-on Labs – 5 hands-on labs that guide you step-by-step in creating applications that highlight some of the great new features in Silverlight.

Building Business Applications with Silverlight 4 – This course covers many key scenarios that are faced when building business applications and shows how Silverlight 4 can help address them. Topics range from right click, context menus, WCF RIA Services, MEF, navigation, RichTextBox, implicit styles, webcam, DataGrid features, validation, drop targets, several out of browser features in depth, and much more. You can find this course at http://r.ch9.ms/sl4

Screencasts

Training Screencast Videos – Several training videos guide you through the new Silverlight features! You can find these videos and the source code for their sample applications at http://silverlight.net/learn

---

Blogs

You can gain more insight by reading the post about Silverlight on these blogs or on Twitter:

- John Papa
- Adam Kinney
- Tim Heuer

- Jesse Liberty
- Team Silverlight

Twitter

- @john_papa
- @adkinn
- @timheuer
- @jesseliberty
- @teamsilverlight
- @silverlightTv

Key Links

You can also find more information at:

- Silverlight 4 Training Course on Channel 9
- Silverlight 4 Forums on the Silverlight community site
- Watch Silverlight TV on Channel 9 and catch exclusive interviews with Silverlight product team members and community leaders
- The Silverlight.net community site, where you can find what you need to get started, learning materials, and more
- The Expression Gallery is a great place to find behaviors, controls, themes, and other samples
- Fantastic tutorials for designers can be found at Project Rosetta

---