Ensuring That Your Silverlight 2 or Silverlight 3 Beta Applications Work with the Silverlight 3 Release

# Introduction

This topic discusses the changes made to the Silverlight runtime and Silverlight tools between Silverlight 2 and Silverlight 3. The changes discussed in this topic are focused on changes that might cause your older Silverlight-based applications to now fail or behave differently, not on new features/enhancements for this release. This topic has three primary sections:

1. Breaking Changes since Silverlight 3 Beta: The changes described in this section could potentially break applications written for Silverlight 3 Beta.
2. Breaking Changes since Silverlight 2: The changes described in this section could potentially break applications written for Silverlight 2.
3. Upgrade Breaking Changes. These changes will not break Silverlight 2 applications unless you re-compile using Silverlight 3.

Note   Corrections/Additions to this topic (if any) are listed [here](http://go.microsoft.com/fwlink/?LinkId=119604).

# Table of Contents

[1 Introduction 1](#_Toc233347138)

[2 Table of Contents 1](#_Toc233347139)

[3 Breaking Changes since Silverlight 3 Beta 4](#_Toc233347140)

[3.1 DockPanel, WrapPanel, Expander, HeaderedContentControl, Viewbox, DataForm moved to the Silverlight Toolkit 4](#_Toc233347141)

[3.2 AutoCompleteBox: Search -> Filter 5](#_Toc233347142)

[3.3 New template part for AutoCompleteBox: “Selector”, typeof(Selector) 5](#_Toc233347143)

[3.4 AutoCompleteBox.TryGetSelectionAdapter(object o) is now GetSelectionAdapterPart() 5](#_Toc233347144)

[3.5 AutoCompleteBox: The “DropDownToggle” template part of type ToggleButton has been removed 6](#_Toc233347145)

[3.6 DataGrid RowGroupExpanding and RowGroupCollapsing events removed 6](#_Toc233347146)

[3.7 DataGrid.SortDescriptions and GroupDescriptions properties removed 7](#_Toc233347147)

[3.8 DataGrid Entity level validation errors are shown using a ValidationSummary instead of a ListBox 7](#_Toc233347148)

[3.9 DataGrid no longer selects the first item by default 8](#_Toc233347149)

[3.10 DataForm Changes 8](#_Toc233347150)

[3.11 Breaking Changes to System.ComponentModel.dll and the DataPager 10](#_Toc233347151)

[3.12 PagedCollectionView 10](#_Toc233347152)

[3.13 DataPagerAutomationPeer 10](#_Toc233347153)

[3.14 BindableAttribute and BindingDirection have been removed 11](#_Toc233347154)

[3.15 Changes to ValidationAttribute 11](#_Toc233347155)

[3.16 Signatures for [CustomValidation] methods have changed 12](#_Toc233347156)

[3.17 Frame only supports navigating to Pages 13](#_Toc233347157)

[3.18 Frame breaking change related to UriMapper 13](#_Toc233347158)

[3.19 Effects Files (.ps) can no longer be loaded as Content 14](#_Toc233347159)

[3.20 WritableBitmap constructor parameter removed 14](#_Toc233347160)

[3.21 RenderOptions.TextRenderingOptions is now TextOptions.TextHintingMode 14](#_Toc233347161)

[3.22 Transparent platform extensions are not hosted on Microsoft.com 15](#_Toc233347162)

[3.23 Out-of-browser API name changes 15](#_Toc233347163)

[3.24 ExecutionStates.DetachedUpdatesAvailable is removed 16](#_Toc233347164)

[3.25 Icon.Size type change and <Icon Size=”…”> format changed 17](#_Toc233347165)

[3.26 XmlConvert.IsNCNameSurrogatePair() and IsStartNCNameSurrogatePair() have been removed 17](#_Toc233347166)

[3.27 SaveFileDialog.File -> SaveFileDialog.SafeFileName 17](#_Toc233347167)

[3.28 WriteableBitmap[] is now WritableBitmap.Pixels[] 18](#_Toc233347168)

[3.29 WriteableBitmap.Lock() and Unlock() have been removed; calling Invalidate() is now required 18](#_Toc233347169)

[4 Breaking Changes since Silverlight 2 18](#_Toc233347170)

[4.1 System.Web.Silverlight.dll has been removed from the SDK 18](#_Toc233347171)

[4.2 Silverlight Applications now respond to Browser Zoom 19](#_Toc233347172)

[4.3 Popup Tab key Change 20](#_Toc233347173)

[4.4 ComboBox Popup Clipping Fixed 20](#_Toc233347174)

[4.5 Silverlight always redraws Popups when PopupRenderTransform changes 20](#_Toc233347175)

[4.6 ContentPresenter.Content = "string" no longer changes the ContentTemplate property 20](#_Toc233347176)

[4.7 ContentControl now ignores Content when using an empty ContentTemplate 21](#_Toc233347177)

[4.8 Setting ContentControl.Content does not cause unnecessary OnApplyTemplate calls 21](#_Toc233347178)

[4.9 ComboBox and ContentControl now displays strings rather than numbers when ItemsSource is set to enum with typeconverter 21](#_Toc233347179)

[4.10 ComboBox with no selected item takes an extra keystroke to select second item 22](#_Toc233347180)

[4.11 HorizontalScrollBarVisibility and VerticalScrollBarVisibility can now be set in a style for ComboBox and ListBox 22](#_Toc233347181)

[4.12 ListBoxItem.HorizontalContentAlignment is now respected 22](#_Toc233347182)

[4.13 ListBoxItem.Style now takes precedence over ListBox.ItemContainerStyle 22](#_Toc233347183)

[4.14 Tabbing inside a ListBox now moves focus to next control rather than next ListBox Item 22](#_Toc233347184)

[4.15 Removed TemplatePart declarations from PasswordBox/TextBox 22](#_Toc233347185)

[4.16 TextAlignment in TextBox now works relative to the document instead of the TextBox 23](#_Toc233347186)

[4.17 Default colors for read-only TextBoxes have changed 23](#_Toc233347187)

[4.18 ENTER key will now respects Button.ClickMode which defaults to Release (KeyUp) 23](#_Toc233347188)

[4.19 Slider control's hit test area now matches what is visible 23](#_Toc233347189)

[4.20 Malformed URLs inside clientaccesspolicy.xml now cause that URL to be ignored, not the rest of the file 24](#_Toc233347190)

[4.21 Bug fix where changing Opacity did not always redraw 24](#_Toc233347191)

[4.22 DataBinding requires property setter to be public 24](#_Toc233347192)

[4.23 Changing Binding.Path after SetBinding() is not allowed 24](#_Toc233347193)

[5 Upgrade Breaking Changes 24](#_Toc233347194)

[5.1 OpenFileDialog.ShowDialog() can now only be opened from a user initiated event 25](#_Toc233347195)

[5.2 Tabbing processing and associated callbacks changed from asynchronous to synchronous 26](#_Toc233347196)

[5.3 Mouse events inside a Popup now give positions relative to the pop-up not the entire application 26](#_Toc233347197)

[5.4 Popup can now only have one parent 26](#_Toc233347198)

[5.5 Namescopes inside Popup now work correctly 27](#_Toc233347199)

[5.6 FindElementsInHostCoordinates now searches Popup's 27](#_Toc233347200)

[5.7 Removed extra events associated with replacing Item in ItemsControl 27](#_Toc233347201)

[5.8 ScrollViewer now clamps scrollbar positions to ScrollableWidth/ScrollableHeight 27](#_Toc233347202)

[5.9 Rectangle and Ellipse now appear when Stretch is set and Height and/or Width are not set 27](#_Toc233347203)

[5.10 Width of TextBlock, TextBox, and PasswordBox now rounded up 27](#_Toc233347204)

[5.11 textblock.FontFamily = null throws ArgumentNullException 28](#_Toc233347205)

[5.12 Spaces no longer rendered between Run elements within template 28](#_Toc233347206)

[5.13 RadioButton.GroupName now searches outside template for RadioButtons 28](#_Toc233347207)

[5.14 ComboBox content now gets IsHitTestVisible="false" 28](#_Toc233347208)

[5.15 TextBox now respects the Opacity property 28](#_Toc233347209)

[5.16 FrameworkElementAutomationPeer ctor with null parameter throws NullReferenceException 28](#_Toc233347210)

[5.17 HyperlinkButton now navigates the entire window and not just the IFRAME that Silverlight is in 29](#_Toc233347211)

[5.18 Correct base URI used when Silverlight v2+ application does not have XAP file extension 29](#_Toc233347212)

[5.19 Application.Current.Host.Source now returns an absolute URL even when .xap URL has a query string 29](#_Toc233347213)

[5.20 ResourceDictionary enum bug fixed 29](#_Toc233347214)

[5.21 ReadOnlyObservableCollection moved from System.Windows.Controls.Data.dll to System.Windows.dll 29](#_Toc233347215)

[5.22 DataGridEndingEditEventArgs was removed from the SDK 30](#_Toc233347216)

[5.23 PollingDuplexHttpBinding Changes 30](#_Toc233347217)

[5.24 Bug fix with removing event handlers 30](#_Toc233347218)

[5.25 Bug fix allowing <Cursors> instead of <Cursor> 30](#_Toc233347219)

[5.26 Custom attached properties in XAML require xmlns are prefixed 30](#_Toc233347220)

[5.27 Bug fix where subclass control could load wrong resource 30](#_Toc233347221)

[5.28 Bug fix where ValueConverter is not called when leaf object on the path is null 31](#_Toc233347222)

[5.29 Thumb control now calculates drag relative to the absolute position rather than relative to its parent 31](#_Toc233347223)

[5.30 Setting Binding properties after SetBinding() no longer supported 31](#_Toc233347224)

[5.31 Child of a ContentPresenter can now only have one parent 31](#_Toc233347225)

[5.32 Relative URLs pointing to resources outside the .xap are now relative to the .xap file not the HTML page 32](#_Toc233347226)

[5.33 Silverlight 3 automatically displays validation errors 32](#_Toc233347227)

[5.34 Image.Source and ImageBrush.Source now return null when not initialized 32](#_Toc233347228)

# Breaking Changes since Silverlight 3 Beta

## DockPanel, WrapPanel, Expander, HeaderedContentControl, Viewbox, DataForm moved to the Silverlight Toolkit

The following controls and types have been removed from the Silverlight SDK:

         DockPanel

         WrapPanel

         Expander

         HeaderedContentControl

         Viewbox

         DataForm

         ExpandDirection

         ExpanderAutomationPeer

         LengthConverter

         StretchDirection

They are now in the Silverlight Toolkit, which is available from <http://www.codeplex.com/Silverlight>.

## AutoCompleteBox: Search -> Filter

* AutoCompleteBox.SearchMode property is now AutoCompleteBox.FilterMode
* AutoCompleteSearchMode enum is now AutoCompleteFilterMode
* AutoCompleteSearchPredicate delegate is now AutoCompleteFilterPredicate

**Prior to this change:**

<input:AutoCompleteBox SearchMode=”Contains” />

**After the change:**

            <input:AutoCompleteBox FilterMode=”Contains” />

## New template part for AutoCompleteBox: “Selector”, typeof(Selector)

If you have a AutoCompleteBox control template with a part named “SelectionAdapter”, you should rename it to “Selector”.

Previously, AutoCompleteBox would grab any template part named ‘SelectionAdapter’, and attempt to either use it as an adapter, or wrap it in a pre-defined SelectorSelectionAdapter.

With this change, the basic logic to prepare the Selection Adapter for the AutoCompleteBox is:

         Find a template part named ‘Selector’.

o   If the control *implements* ISelectionAdapter, use it.

o   Else, if a Selector was found, create a new SelectorSelectionAdapter for it.

         If no SelectionAdapter has been prepared, locate a template part named SelectionAdapter.

## AutoCompleteBox.TryGetSelectionAdapter(object o) is now GetSelectionAdapterPart()

Previously, the AutoCompleteBox API contained a protected virtual method named “TryGetSelectionAdapter.” The API did not actually implement the TryGet pattern.

The new API is simply a protected virtual method with no parameters named GetSelectionAdapterPart.

## AutoCompleteBox: The “DropDownToggle” template part of type ToggleButton has been removed

The default control template for AutoCompleteBox never has made use of this template part. To improve the templating experience in Blend, this template part has been removed.

**How do I update an existing application?**

In the file with the custom control template definition, add a Click event handler in XAML and code behind. Then, locate the auto complete control in the visual tree hierarchy and toggle its IsDropDownOpen dependency property.

Anyone who derived from AutoCompleteBox should still be able to define their own ToggleButton template part and flip the IsDropDownOpen property.

From an updated sample application, here is part of the custom control template:

|  |  |
| --- | --- |
| Before (XAML) | After (XAML) |
| <ToggleButton x:Name="DropDownToggle" Style="{StaticResource ComboToggleButton}" /> | <ToggleButton ~~x:Name="DropDownToggle"~~ Style="{StaticResource ComboToggleButton}" **Click=”DropDownToggle\_Click”** /> … |

And the code behind file:

|  |  |
| --- | --- |
| Before (C#) | After (C#) |
| (n/a) | private void DropDownToggle\_Click(object sender, RoutedEventArgs e){     FrameworkElement fe = sender as FrameworkElement;     AutoCompleteBox acb = null;     while (fe != null && acb == null)     {         fe = VisualTreeHelper.GetParent(fe) as FrameworkElement;         acb = fe as AutoCompleteBox;     }     if (acb != null)     {         acb.IsDropDownOpen = ! acb.IsDropDownOpen;     }} |

## DataGrid RowGroupExpanding and RowGroupCollapsing events removed

These events did not completely work in the Beta, and have been removed.

## DataGrid.SortDescriptions and GroupDescriptions properties removed

The SortDescriptions and GroupDescriptions properties were removed from DataGrid.  You can use the code behind to setup the group descriptions.

Silverlight 3 Beta

[Xaml]

<dataGrid.GroupDescriptions>

      <windata:PropertyGroupDescription PropertyName=”State” />

</dataGrid.GroupDescriptions>

Silverlight 3 RTM

[c#]

pagedCollectionView.GroupDescriptions.Add(new PropertyGroupDescription(“State”));

## DataGrid Entity level validation errors are shown using a ValidationSummary instead of a ListBox

Who Is Affected: Silverlight 3 Beta managed applications that re-templated the DataGrid to a custom template that includes a ListBox for validation errors.

Summary: To have a standard look and feel for entity level validation errors, the DataGrid now uses a ValidationSummary similar to the DataForm. Previously, the DataGrid used a ListBox.

Fix Required: Users with custom DataGrid templates that include a ListBox for validation errors need to update their templates.

Silverlight 3 Beta

[Xaml]

<ListBox Name="ErrorsListBox" Grid.Row="3" Grid.ColumnSpan="3" MaxHeight="91" Background="#FFEFEFEA" BorderThickness="0,1,0,0">

…

Silverlight 3 RTM

[Xaml]

<dataInput:ValidationSummary Name="ValidationSummary" Filter="ObjectErrors" Grid.Row="3" Grid.ColumnSpan="3" MaxHeight="90"/>

## DataGrid no longer selects the first item by default

Who Is Affected: Silverlight 2 applications that depend on the DataGrid selecting the first item by default.

Summary

Based on customer feedback, the DataGrid no longer selects the first item by default, which also matches that of other list based controls like the ListBox.  Now, the DataGrid’s current item is by default the first item, but that item is not selected.  Tabbing in to the DataGrid causes the current item to be selected.  The DataGrid will, however, select the an ICollectionView’s current item by default if its ItemsSource is set to an ICollectionView implementation.  This is consistent with the ListBox and consistent with other consumers of ICollectionView, and it also supports the master/details scenario via CollectionView.

Fix Required

Silverlight 2 users depending on the DataGrid to select the first item by default now need to explicitly set the SelectedItem or the SelectedIndex after setting the ItemsSource.

## DataForm Changes

The largest change related to **DataForm** impacts anyone using the **Fields** collection in the **DataForm**, and is as follows:

**DataFormFields** have all been removed and **DataForm.Fields** collection has been removed. They have been replaced with the new **DataField** control, which is a control that wraps content with a label and **DescriptionViewer**. Also, note that you must now explicitly specify **Mode=TwoWay** on bindings in the **DataField**, whereas before you could just specify the path and the **DataFormField** would inject TwoWay into the binding. For example, if you have the following in the Fields collection:

<dataControls:DataFormTextField Binding="{Binding FirstName}" />

...you would want to instead place this in the EditTemplate:

<dataControls:DataField>

 <TextBox Text="{Binding FirstName, Mode=TwoWay}" />

</dataControls:DataField>

Note that since these are now controls, **DataFormFieldGroups** can be replaced just with regular panels.

Also note that since these are now controls unto themselves, the **DataForm** no longer handles all of the alignment of the labels in these fields. The **DataForm** will group these fields together at the top level, but if you want more fine-tuned grouping (for example, two columns of fields), you can use the attached property **DataField.IsFieldGroup** on a **Panel** to specify that the **DataFields** within the **Panel** should have their labels grouped together in width. The following shows an example:

<StackPanel Orientation="Horizontal">

 <StackPanel dataControls:DataField.IsFieldGroup="True">

 <dataControls:DataField>

 <TextBox Text="{Binding FirstName, Mode=TwoWay}" />

 </dataControls:DataField>

 <dataControls:DataField>

 <TextBox Text="{Binding LastName, Mode=TwoWay}" />

 </dataControls:DataField>

 </StackPanel>

 <StackPanel dataControls:DataField.IsFieldGroup="True">

 <dataControls:DataField>

 <TextBox Text="{Binding StreetAddress, Mode=TwoWay}" />

 </dataControls:DataField>

 <dataControls:DataField>

 <TextBox Text="{Binding State, Mode=TwoWay}" />

 </dataControls:DataField>

 </StackPanel>

</StackPanel>

This would cause the FirstName/LastName DataFields to have their labels the same width, and similarly for StreetAddress/State DataFields.

That represents the bulk of this breaking change. In addition, there are other breaking changes on the **DataForm** which are as follows:

* **FieldEditEnding** and **FieldEditEnded** have been removed.
* **GetFieldElement()** has been removed.
* **ItemEditEnding**, **ItemEditEnded**, **CommitItemEdit()** have been changed to **EditEnding**, **EditEnded**, and **CommitEdit()** respectively.
* CanMoveToXXXXItem have been removed, as they can be derived from the collection given to the **DataForm**.
* **CanUserAdd**/**DeleteItems** have been removed, as they are implied through **CommandButtonsVisibility**.
* **IsEditing** and **IsAddingItem** have been removed, as they are implied through **Mode**.
* **FieldLabelPosition** and **FieldLabelStyle** have become **LabelPosition** and **LabelStyle**.
* **WrapAfter** and **Orientation** have been removed.
* The type names **DataFormFieldLabelPosition** and **DataFormDescriptionViewerPosition** have been changed to **DataFieldLabelPosition** and **DataFieldDescriptionViewerPosition**.

## Breaking Changes to System.ComponentModel.dll and the DataPager

These breaking changes are summarized below:

1. **System.ComponentModel.dll** has been deleted.
2. From **System.ComponentModel**, the following classes/interfaces have been moved to the new **System.Windows.Data.dll** (namespaces have remained unchanged)
	1. PagedCollectionView
	2. CollectionViewGroup
	3. IEditableCollectionView
	4. IPagedCollectionView
	5. PageChangingEventArgs
	6. PropertyGroupDescription
3. The following classes/interfaces will no longer be part of the SDK
	1. EntityAction
	2. IEditableCollection
	3. IIndexableCollection
4. **IPagedCollection** has been removed.  **PagedCollectionView** no longer has special-cases implementers of **IPagedCollection**.
5. **DataPager.Source** is no longer of type **IPagedCollectionView**, but is now an **IEnumerable** that may also be an **IPagedCollectionView**.  When non-IPagedCollectionViews are given to the DataPager, it behaves as though a collection with a single large page was bound to the control.  To invoke **IPagedCollectoinView** APIs, cast the **Source** property to an **IPagedCollectionView**.

## PagedCollectionView

PagedCollectionView no longer supports arbitrary remote paging. As a result, PagedCollectionView.NewItemPlaceholder has been removed, and PagedCollectionView.ItemCount and TotalItemCount properties are now read-only.

## DataPagerAutomationPeer

The DataPagerAutomationPeer now implements IRangeValueProvider explicitly rather than implicitly. As a result, the following members have been removed from the DataPagerAutomationPeer class:

        public bool IsReadOnly { get; set; }

        public double LargeChange { get; set; }

        public double Maximum { get; set; }

        public double Minimum { get; set; }

        public double SmallChange { get; }

        public double Value { get; }

        public void SetValue(double value);

## BindableAttribute and BindingDirection have been removed

In Silverlight 3 Beta, the Bindable attribute was used to annotate the properties of a business object allowing the UI to react in two ways:

         To treat the property as read only in the UI.

         To tell controls that perform auto-generation, such as DataGrid and DataForm, if they should generate UI for the property.

In the Silverlight 3 release, the Editable and Display attributes should be used for these purposes.

Marking a property as ReadOnly in the UI:

|  |  |
| --- | --- |
| Before | After |
| **[Bindable (true, BindingDirection.OneWay)]** public string foo {get;set;} | **[Editable (false)]**public string foo {get;set;} |

Marking a property to not have controls auto-generate UI for it:

|  |  |
| --- | --- |
| Before | After |
| **[Bindable (false)]** public int bar {get;set;} | **[Display (AutoGenerateField=false)]**public int bar {get;set;} |

## Changes to ValidationAttribute

Any classes that derive from the Beta of ValidationAttribute will likely need to be updated. Instead of returning a bool from IsValid, implementations should only override the ValidationContext-based IsValid method and return ValidationResult. ValidationContext can be ignored if it is not needed.

The ValidationAttributeAPI has been changed to the following:

public abstract class ValidationAttribute: Attribute {

 ~~public void Validate(object value, string name);~~

 public void Validate(object value, ValidationContext validationContext);

 ~~public bool TryValidate(object value, ValidationContext validationContext, ref ValidationResult validationResult);~~

 public ValidationResult GetValidationResult(object value, ValidationContext validationContext~~, out ValidationResult validationResult~~);

 ~~public abstract bool IsValid(object value);~~

 protected virtual ~~bool~~ ValidationResult IsValid(object value, ValidationContext validationContext~~, out ValidationResult validationResult~~);

}

Representing Success and Failure: ValidationAttribute implementations need to return ValidationResult to represent both success and failure.  In order to represent success, public static readonly ValidationResult Success was added to ValidationResult.  For failure scenarios, ValidationResult has two constructors for providing an error message and optionally a list of member names.  These constructors did not change, but the Success field was an addition.

Here is a very simple implementation of a ValidationAttribute, illustrating both success and failure.

    public class IsTrueAttribute : ValidationAttribute {

        protected override ValidationResult IsValid(object value, ValidationContext validationContext) {

            if (value is bool && (bool)value) {

                return ValidationResult.Success;

            }

            else {

                return new ValidationResult("The value provided was not true");

            }

        }

    }

The Validator class was updated to call GetValidationResult instead of TryValidate. Anything else that was calling TryValidate will also need to be updated, although there is no other framework code affected by this change.

## Signatures for [CustomValidation] methods have changed

Anywhere that a custom validation method was implemented, the signature will have to be updated. Conflicts will not be discoverable at runtime. If you have implemented any custom validation methods, you must find those implementations and update the method signatures to match one of the new supported signatures.

CustomValidationAttribute is used to provide a Type and a Method to invoke for validation.  This is a simple mechanism for extending the validation framework to call custom validation routines, which uses a convention for the method signatures so that the referenced method can be called.

In Silverlight 3 Beta, [CustomValidation] attributes required one of the following method signatures:

public static bool MethodName([type] value) {…}

public static bool MethodName([type] value, ValidationContext context, out ValidationResult result) {…}

In the final version of Silverlight 3, the supported method signatures have been changed to the following:

public static ValidationResult MethodName([type] value) {…}

public static ValidationResult MethodName([type] value, ValidationContext context) {…}

The return type has changed, and the "out ValidationResult" parameter has been removed.

## Frame only supports navigating to Pages

Frame only supports navigation to XAML for controls derived from Page.  Attempts to navigate to items other than pages will result in an exception.

In Silverlight 3 Beta, it was possible to navigate to XAML for a UserControl, which increased complexity and required users to know how to reach the Navigation service to get navigation-related information within that UserControl.  If this is still necessary, wrap the UserControl in a Page or use a Page as the parent class.

## Frame breaking change related to UriMapper

Previously, the way to provide a UriMapper to a Frame control was by adding it as a resource with a key of “uriMapper”:

**Prior to this change:**

<navigation:Frame.Resources>

    <nav:UriMapper x:Key=”uriMapper”>

    …

    </nav:UriMapper>

</navigation:Frame.Resource>

**After the change:**

The Frame control has a property named UriMapper that takes a UriMapperBase:

    public static readonly DependencyProperty **UriMapperProperty**;
    public UriMapperBase **UriMapper** { get; set; }

Now, you can explicitly declare and specify a UriMapper, for example by binding to a resource:

<navigation:Frame UriMapper=”{StaticResource uriMapper}” …>

</navigation:Frame>

Or, you can specify it directly:

<navigation:Frame …>

    <navigation:Frame.UriMapper>

     <nav:UriMapper>

       …

       </nav:UriMapper>

    </navigation:Frame.UriMapper>

</navigation:Frame>

The API for UriMapper itself is unchanged.

## Effects Files (.ps) can no longer be loaded as Content

In Silverlight 3 Beta, effects files (pixel shaders) could be packaged as ‘Content’ or as ‘Resource’ (Build Action = Content or Resource in the Visual Studio Properties window). Silverlight now only supports effects files packaged as Resource. (There is no change to the built-in DropShadow and Blur effects.)

To use a PS file, set its build action in Visual Studio to "Resource" and access it via an assembly URI of the form "<assemblyName>;component/<filename>.ps". The following shows an example of the SilverlightCustomEffect application loading GrayScaleEffect.fx.ps:

pshader = new PixelShader() { UriSource = new Uri("/SilverlightCustomEffect;component/GrayScaleEffect.fx.ps", UriKind.Relative) };

## WritableBitmap constructor parameter removed

The PixelFormat parameter for the WritableBitmap constructor has been removed:

public [WriteableBitmap](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Windows:2.0.5.0:7cec85d7bea7798e/System.Windows.Media.Imaging.WriteableBitmap/.ctor(Int32,Int32,System.Windows.Media.PixelFormat))([int](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.5.0:7cec85d7bea7798e/System.Int32) pixelWidth, [int](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://mscorlib:2.0.5.0:7cec85d7bea7798e/System.Int32) pixelHeight**~~,~~** [**~~PixelFormat~~**](http://www.aisto.com/roeder/dotnet/Default.aspx?Target=code://System.Windows:2.0.5.0:7cec85d7bea7798e/System.Windows.Media.PixelFormat) **~~format~~**);

The only supported PixelFormat is now Pbgra32.

Similarly, the PixelFormat and PixelFormats type has been removed.

## RenderOptions.TextRenderingOptions is now TextOptions.TextHintingMode

**Prior to this change:**

 public static class RenderOptions {

 public static DependencyProperty TextRenderingMode TextRenderingModeProperty;

 }

 public enum TextRenderingMode {

 Default = 0,

 RenderForAnimation = 1

 }

**After the change:**

 public static class TextOptions {

 public static DependencyProperty TextHintingMode TextHintingModeProperty;

 }

 public enum TextHintingMode {

 Fixed = 0,

 Animated = 1

 }

## Transparent platform extensions are not hosted on Microsoft.com

Transparent platform extensions will not be hosted on Microsoft.com. Instead they will be hosted by default on the application’s site of origin. Application developers can override this setting by editing the assembly-to-extension mapping file (<assemblyShortName>.extmap.xml) located with the assembly.

## Out-of-browser API name changes

Change to the Detach method in the System.Windows.Application type.

|  |  |  |
| --- | --- | --- |
|  | From | To |
| Signature | public bool Detach();public bool RunningOffline;public event EventHandler ExecutionStateChanged; | public bool Install();public bool RunningOutOfBrowser;public event EventHandler InstallStateChanged; |

Changes to the ExecutionStates enumeration.

|  |  |  |
| --- | --- | --- |
|  | From | To |
| Signature | public enum ExecutionStates {    RunningOnline,    Detaching,    Detached,    DetachedUpdatesAvailable,    DetachFailed} | public enum InstallState{    NotInstalled,    Installing,    Installed,    (removed, see below)    InstallFailed} |

Change to the ApplicationIdentity property in the Deployment element of the application manifest (AppManifest.xaml).

|  |  |  |
| --- | --- | --- |
|  | From | To |
| XAML | <Deployment.ApplicationIdentity … | <Deployment.OutOfBrowserSettings … |
| Signature | public ApplicationIdentity ApplicationIdentity; | public OutOfBrowserSettings OutOfBrowserSettings ; |

Moved ApplicationIdentity.Title to OutOfBrowserSettings.WindowSettings.Title.

|  |  |  |
| --- | --- | --- |
|  | From | To |
| XAML | <ApplicationIdentity                        ShortName=”…”                        Title=”…”> | <OutOfBrowserSettings ShortName=”…”>    <OutOfBrowserSettings.WindowSettings>          <WindowSettings Title=”…”     /> <OutOfBrowserSettings.WindowSettings /> |

## ExecutionStates.DetachedUpdatesAvailable is removed

The ExecutionStates.DetachedUpdatesAvailable value has been removed from the ExecutionStates enum (which is now the InstallState enum). Consider instead using one of the following new APIs:

* Application.CheckAndDownloadUpdateAsync() method
* Application.CheckAndDownloadUpdateCompleted event
* CheckAndDownloadUpdateCompletedEventArgs class
* CheckAndDownloadUpdateCompletedEventArgs.UpdateAvailable property

## Icon.Size type change and <Icon Size=”…”> format changed

This type is typically used inside appmanifest.xaml to specify icon information for out-of-browser applications. Changes affect both XAML and programmatic usage:

* The Icon.Size property has changed from type string to type Size.
* Values “small”, “medium”, and “large” are no longer accepted for Icon.Size
* The XAML format is now the same as other properties of type Size. For example, “16,16” instead of “16x16”.
* Icon sizes are now validated. Valid icon sizes are “16,16”; "32,32"; "48,48"; "128,128"; "256,256"; "512,512".

**Example:**

**Before:**

<Icon Size="16x16">16.png</Icon>

<Icon Size="32x32">32.png</Icon>

<Icon Size="48x48">48.png</Icon>

<Icon Size="128x128">128.png</Icon>

**After:**

<Icon Size="16,16">16.png</Icon>

<Icon Size="32,32">32.png</Icon>

<Icon Size="48,48">48.png</Icon>

<Icon Size="128,128">128.png</Icon>

## XmlConvert.IsNCNameSurrogatePair() and IsStartNCNameSurrogatePair() have been removed

The following methods have been removed:

class XmlConvert {

  public static bool IsNCNameSurrogatePair(char lowChar, char highChar);

  public static bool IsStartNCNameSurrogatePair(char lowChar, char highChar);

}

## SaveFileDialog.File -> SaveFileDialog.SafeFileName

The SaveFileDialog.File property has been removed. It has been replaced with the SaveFileDialog.SafeFileName property:

public String SafeFileName { get; }

A String that only contains the file name for the selected file. The default is Empty, which is also the value when either no file is selected or a directory is selected.

## WriteableBitmap[] is now WritableBitmap.Pixels[]

The indexer on the WriteableBitmap class has been replaced with the Pixels property:

 public sealed class WriteableBitmap: BitmapSource {

 ~~public int this[int index] { get; set; }~~

 public int[] Pixels { get; }

 }

In typical usage:

myBitmap[index] = pixel;

becomes:

myBitmap.Pixels[index] = pixel;

## WriteableBitmap.Lock() and Unlock() have been removed; calling Invalidate() is now required

In Silverlight 3 beta, one had to call Lock() before modifying pixels and Unlock() afterwards. Those calls are now unnecessary, instead one calls Invalidate() after modifications are complete. E.g.:

myBitmap.Lock();

… modify pixels …

myBitmap.Unlock();

Becomes:

… modify pixels …

myBitmap.Invalidate();

# Breaking Changes since Silverlight 2

The changes described in this section could potentially break Silverlight 2 applications. Many of these changes are actually bug fixes; nevertheless, if your Silverlight 2 application employed workarounds for these bugs, it is possible that you will have to remove these workarounds for your application to work correctly.

## System.Web.Silverlight.dll has been removed from the SDK

The ASP.NET Silverlight and MediaPlayer controls have been removed from the Silverlight 3  SDK. These are contained in a server assembly named System.Web.Silverlight.dll that typically gets copied to the test Web site’s BIN folder.

*<asp:silverlight ../>*

*<asp:mediaplayer ../>*

Existing Silverlight 2 web sites will continue to work as they have the System.Web.Silverlight assembly in their Web application’s BIN folder (even if the Silverlight 2 SDK is uninstalled).

When upgrading existing projects to Silverlight 3, in most cases, you should replace these controls with the equivalent <object> tag. For <asp:silverlight>, that is:

      <div id="silverlightControlHost">

            <object data="data:application/x-silverlight-2," type="application/x-silverlight-2" width="100%" height="100%">

                <param name="source" value="{XAP\_FILE}"/>

                <param name="onerror" value="onSilverlightError" />

                <param name="background" value="white" />

                <param name="minRuntimeVersion" value="{BUILD}" />

                <param name="autoUpgrade" value="true" />

                <a href="http://go.microsoft.com/fwlink/?LinkID=149156&v={BUILD}" style="text-decoration: none;">

                     <img src="http://go.microsoft.com/fwlink/?LinkId=108181" alt="Get Microsoft Silverlight" style="border-style: none"/>

                </a>

            </object><iframe id="\_sl\_historyFrame" style='visibility:hidden;height:0;width:0;border:0px'></iframe></div>

**Existing** Silverlight applications will continue to have the System.Web.Silverlight assembly in their Web application’s BIN folder, even if the Silverlight 2 SDK is uninstalled. Therefore upgrading to Silverlight 3 SDK will not remove this local assembly and applications should continue to work. Notionally, a developer will set the relevant version flag for Silverlight 3.

* In the client, the older fwlinks rendered should now point to Silverlight 3.
* Developers who added the System.Web.Silverlight assembly to the GAC would have to manually remove it from the GAC per the install.

Developers creating **new applications** will want to use the latest templates that contain the OBJECT tag and other markup.

* Developers ‘could’ add the System.Web.Silverlight assembly manually and use the server controls. However, these controls will not provide the latest installation logic, or be able to render iframes for Silverlight history support for example.

## Silverlight Applications now respond to Browser Zoom

Silverlight applications (old and new) will have the opportunity to respond to Browser Zoom depending on their current behavior. Silverlight applications will now zoom by default if the application does not handle the Resized event. To the application, zooming is equivalent to a RenderTransform = new ScaleTransform() being applied to the imaginary parent of the root element.

Applications may customize this behavior by using the following new APIs:

* System.Windows.Interop.Content's Zoomed event and ZoomFactor property. (Content class is usually used as App.Current.Host.Content)
* The EnableAutoZoom parameter on the Silverlight plugin

## Popup Tab key Change

In Silverlight 2 if you have focus inside a popup and you press the TAB key, focus leaves Silverlight entirely and goes to the browser; ignoring other tab stops within the popup. The new and correct behavior in Silverlight 3 is to tab around inside the popup first. If you want the old behavior you can set **TabNavigation=”Cycle”** on the root of your popup.

## ComboBox Popup Clipping Fixed

In Silverlight 2 if there is a vertical scrollbar in the **ComboBox** popup, and the popup is wider than the **ComboBox** itself, the popup is too narrow by the width of the scrollbar:



If you worked around this Silverilght 2 bug by artificially increasing the size of the **ComboBox** items, in Silverlight 3 there will be extra space and you may want to remove your workaround.

## Silverlight always redraws Popups when PopupRenderTransform changes

In Silverlight 2, Silverlight did not always redraw popups when **Popup.RenderTransform** was changed. This has been fixed in Silverlight 3.

## ContentPresenter.Content = "string" no longer changes the ContentTemplate property

Previously, if you set **ContentPresenter.Content = string**, and then queried **ContentPresenter.ContentTemplate** you would see a non-null value, the template used to display the string. Now, setting ContentPresenter.Content = string no longer changes the **ContentTemplate** property. This change was done so that **ContentPresenter.Content** could not be set again after it has been set to string.

Note that this is **ContentPresenter**, not **ContentControl**. **ContentPresenter** is typically used only inside control templates like so:

 <ContentPresenter

 Content="{TemplateBinding Content}"

 ContentTemplate="{TemplateBinding ContentTemplate}"

 HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"

 VerticalAlignment="{TemplateBinding VerticalContentAlignment}"

 Margin="{TemplateBinding Padding}"/>

## ContentControl now ignores Content when using an empty ContentTemplate

In Silverlight 2, if you give a ContentControl (such as Button) an empty **ContentTemplate**, the control will ignore the template and displays the content anyway using a **ContentPresenter**. For example, the following displays “ABC” in Silverlight 2:

 <ContentControl Content="ABC">

 <ContentControl.ContentTemplate>

 <DataTemplate></DataTemplate>

 </ContentControl.ContentTemplate>

 </ContentControl>

This has been fixed in Silverlight 3 and so the previous code now displays no content.

## Setting ContentControl.Content does not cause unnecessary OnApplyTemplate calls

In Silverlight 2, setting ContentControl.Content will call OnApplyTemplate every time, even if ContentControl.ControlTemplate has been set. In Silverlight 3, OnApplyTemplate will only be called the first time if ContentControl.ControlTemplate has been set.

## ComboBox and ContentControl now displays strings rather than numbers when ItemsSource is set to enum with typeconverter

In Silverlight 2, if you put enums into a **ComboBox** (its **ItemsSource** property), the **ComboBox** will display the numeric value of the enum rather than the name of the enum. For example:

 enum MyEnum { First, Second, Third }

 …

 MyComboBox.ItemsSource = new MyEnum[] { MyEnum.First };

Displays "1” instead of "First". This has been fixed in Silverlight 3 so now the string value of the enum is displayed.

## ComboBox with no selected item takes an extra keystroke to select second item

In Silverlight 2, when you open a ComboBox that has no selected item, the first item in the list is treated as having focus (even though it is not selected), so the first down arrow key moves both focus and selection to the second item. In Silverlight 3, when that ComboBox is opened, the first item is not considered focused. The first keydown moves selection and focus to the first item, and the second keydown moves selection and focus to the second item.

## HorizontalScrollBarVisibility and VerticalScrollBarVisibility can now be set in a style for ComboBox and ListBox

In Silverlight 2, these properties were set locally in the constructor, overriding any value from the style. This has been fixed.

## ListBoxItem.HorizontalContentAlignment is now respected

In Silverlight 2, when using the default template for a ListBoxItem, the HorizontalContentAlignment was ignored. This has been fixed.

## ListBoxItem.Style now takes precedence over ListBox.ItemContainerStyle

In Silverlight 2, if you set ListBoxItem.Style and then later (in code) set ListBox.ItemContainerStyle, that would erroneously override the ListBoxItem.Style you set earlier. This has been fixed.

## Tabbing inside a ListBox now moves focus to next control rather than next ListBox Item

In Silverlight 2, pressing TAB when focus is inside a **ListBox** moves to the next **ListBox** item, rather than moving focus to the next control. Now the tabbing behavior is consistent with other platforms.

## Removed TemplatePart declarations from PasswordBox/TextBox

For Silverilght 2, **PasswordBox** and **TextBox** had a number of [TemplatePart] attributes describing parts the control did not actually have. These attributes have been removed in Silverlight 3.

## TextAlignment in TextBox now works relative to the document instead of the TextBox

In Silverlight 2, text in **TextBox** did not center based upon the "document", but based upon the width property of the **TextBox**, This caused center rendering to be incorrect. For example, suppose that you start with centered text like this:



In Silverlight 2, if you type a long string, the **TextBox** looks like the following:



In Silverlight 3, this behavior was corrected. The correct behavior is for "ABC" to be centered for the entire document, not the center of the visible portion of the document. This change does not affect the overall size of the **TextBox**, and if the user needs to, they can scroll the viewport to see the centered text. Nevertheless, if you counted on the previous behavior, you may want to adjust your application.

## Default colors for read-only TextBoxes have changed

In Silverlight 2, read-only TextBoxes looked almost identical to editable text boxes.

## ENTER key will now respects Button.ClickMode which defaults to Release (KeyUp)

If the application changes ClickMode to Press, the click handler will not be able to execute code that causes reentrancy. (Examples of code that could cause reentrancy include OpenFileDialog and SaveFileDialog.)

## Slider control's hit test area now matches what is visible

In Silverlight 2, if you made the slider really tall, you could click in an area that does not look like it is part of the slider, even though it is.

## Malformed URLs inside clientaccesspolicy.xml now cause that URL to be ignored, not the rest of the file

In Silverlight 2, an invalid URL caused the remainder of the clientaccesspolicy.xml to be ignored. This has been fixed.

## Bug fix where changing Opacity did not always redraw

In certain circumstances, Silverlight 2 would not redraw content when Opacity was changed. This has been fixed.

## DataBinding requires property setter to be public

Silverlight data binding generally requires properties and types to be public. In Silverlight 2, there was a bug where a property with a public getter but a private setter could be data bound to. This has been fixed.

## Changing Binding.Path after SetBinding() is not allowed

In Silverlight 2, it's possible to set the Binding.Path programmatically after using that Binding in a SetBinding call, but before DataContext is set, eg:

 Binding binding = new Binding();

 myElement.SetBinding(HeightProperty, binding);

 binding.Path = new PropertyPath("propertyname");

 myElement.DataContext = …;

In Silverlight 3, once SetBinding has been called, the Path property may no longer be changed, so the above code needs to be changed to:

 Binding binding = new Binding();

 binding.Path = new PropertyPath("propertyname");

 myElement.SetBinding(HeightProperty, binding);

 myElement.DataContext = …;

If Path is changed after SetBinding() is called, the Silverlight 3 behavior depends on what version of Silverlight the application was compiled against. If compiled against Silverlight 3, setting Path after SetBinding() will throw an exception. However, to reduce the chance of breaking Silverlight 2 applications, if the .xap is built against Silverlight 2 setting Path after SetBinding() will be silently ignored and the old Path will be used (see also discussion of "quirks mode" in section 5).

# Upgrade Breaking Changes

The Silverlight team wanted to fix a number of Silverlight 2 bugs in Silverlight 3. However, by fixing some of these bugs, it is possible that some existing Silverlight 2 applications will be broken. In order to avoid this, Silverlight developers made these potentially problematic changes “quirk mode changes”. A quirk mode change is one that the Silverlight 3 runtime will not enforce if the runtime detects that the application being run was designed for Silverlight 2. In this way, Silverlight 3 is made “bug compatible” with Silverlight 2 when a Silverlight 2 application is being run. However, these changes will impact you if you recompile your application for Silverlight 3.

As shown by the following illustration, if the **RuntimeVersion** for your application specifies Silverlight 2, the runtime runs in "quirks mode". When in quirks mode, the runtime falls back to the Silverlight 2 behavior associated with quirks mode changes.



From the illustration, you can see that the Silverlight 3 runtime detects whether the application was designed for Silverlight 2 by using the **RuntimeVersion**. The **RuntimeVersion** is set as an attribute inside the .xap’s AppManifest.xaml:

<Deployment xmlns="http://schemas.microsoft.com/client/2007/deployment"

 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" EntryPointAssembly="ElevatorSilverlight"

 EntryPointType="ElevatorSilverlight.App"

 RuntimeVersion="2.0.31005.0">

 <Deployment.Parts>

 <AssemblyPart x:Name="ElevatorSilverlight" Source="ElevatorSilverlight.dll" />

 </Deployment.Parts>

</Deployment>

**RuntimeVersion** reflects the build of Silverlight that was on the developer’s machine when the application was compiled.

## OpenFileDialog.ShowDialog() can now only be opened from a user initiated event

For security reasons, OpenFileDialog.ShowDialog() may only be called during a user initiated event (MouseLeftButtonDown/Up and KeyDown/Up) or derived user event like clicking a button or checking a check box. This makes Silverlight behave in the same way as other major browsers and plug-ins.

## Tabbing processing and associated callbacks changed from asynchronous to synchronous

In Silverlight 2, there was a bug associated with tabbing over the last control on the page. As a side effect of fixing this bug, the tabbing processing and associated callbacks, like the **GotFocus** event, were changed from asynchronous to synchronous. (Where "asynchronous" really means PostMessage. It happens on the same thread after the current message has been processed.) This is a breaking change because during an asynchronous event in Silverlight, you cannot call into the HTML DOM. This is the Silverlight version of WPF's unexpected reentrancy guard. In quirks mode, applications continue to get asynchronous tab processing that does not work very well for the last control on the page.

## Mouse events inside a Popup now give positions relative to the pop-up not the entire application

Suppose you have an element inside a pop-up that listens to mouse events:

[xaml]

 <Popup>

 <Border Name="border" MouseLeftButtonDown="OnMouseLeftButtonDown"/>

 </Popup>

[c#]

void OnMouseLeftButtonDown(object sender, MouseButtonEventArgs e)

{

 … e.GetPosition(border) ...

}

In Silverlight 2, the coordinates from MouseButtonEventArgs.GetPosition will be relative to the entire Silverlight plug-in, even though the application asked for coordinates relative to the border control (the parameter to GetPosition is supposed to indicate the frame of reference). This has been fixed in Silverlight 3.

## Popup can now only have one parent

In Silverlight 2, under certain circumstances a **Popup** can actually have two parents. Silverlight 3 allows only a single parent for **Popup**.

## Namescopes inside Popup now work correctly

Silverlight 2 had several bugs where named items inside a Popup were put in the wrong namescope when Popup.Child was its own namescope (for example, child was created with XamlReader.Load). This resulted in FindName() and Storyboard.TargetName finding elements that should not have been found (because Silverlight looked in the wrong name scope), and not finding elements that should be found. In some cases FindName() would find elements that had been removed from the tree. This has been fixed in Silverlight 3. The correct name scope behavior is when Popup is added to the live tree, and Popup.Child is not a namescope owner, the popup children become part of the popup parent's name scope. Otherwise, popup’s children live in a namescope attached to Popup.Child.

## FindElementsInHostCoordinates now searches Popup's

In Silverlight 2, FindElementsInHostCoordinates would not look inside Popup’s. This has been fixed.

## Removed extra events associated with replacing Item in ItemsControl

In Silverlight 2, replacing items in **ItemsControl** generates three events: **Remove**, **Add**, and **Replace**. This is not compatible with WPF and it is hard to code against. When you receive the remove and add events, you do not know if it is really a replace that is about to follow. In Silverlight 3, only the **Replace** event occurs.

## ScrollViewer now clamps scrollbar positions to ScrollableWidth/ScrollableHeight

As a result, the Width and Height of content inside a ScrollViewer will be zero until the layout pass is complete.

## Rectangle and Ellipse now appear when Stretch is set and Height and/or Width are not set

In Silverlight 2, if **Stretch != UniformToFill** and **Height** or **Width** was not set on a **Rectangle** or **Ellipse** within a **ContentControl**, then the object was not rendered. In Silverlight 3, if the **Stretch** property of a **Rectangle** or **Ellipse** is other than None, the object will be rendered.

## Width of TextBlock, TextBox, and PasswordBox now rounded up

In Silverlight 2, if you let **PasswordBox**, **TextBlock** and **TextBox** autosize themselves, they will truncate the final character if the natural width of the text is not a whole number and the fractional part of that is less than .5. For example, text which measures to 96.34 units wide will get rounded down to 96.00, and the **TextBlock** does not have enough space to display the text. Now the width is rounded up but this could potentially affect the layout of your applications.

## textblock.FontFamily = null throws ArgumentNullException

In Silverlight 2, **textblock.FontFamily = null** throws a **NullRefException**, now it throws **ArgumentNullException**.

## Spaces no longer rendered between Run elements within template

Inside a template, Silverlight 2 implicitly inserted a space between elements, so <TextBlock><Run Text="Hel"/><Run Text="lo"/></TextBlock> would have a space between "Hel” and “lo”. In Silverlight 3, this space is no longer rendered.

## RadioButton.GroupName now searches outside template for RadioButtons

In Silverlight 2, if you use **RadioButton.GroupName** inside a template, Silverlight only looks inside that template for RadioButtons with the same group name. This is different from WPF and makes it difficult to perform a lot of useful scenarios. So in Silverlight 3, the **RadioButton** searches outside the template as well.

## ComboBox content now gets IsHitTestVisible="false"

In Silverlight 2, the selected item inside a ComboBox will receive mouse events. This interfered with ComboBox displaying correct visuals for mouse over state, so ComboBox will now set IsHitTestVisible="false" on the ContentPresenter inside its control template. ComboBox itself continues to get mouse events.

## TextBox now respects the Opacity property

In Silverlight 2, the Opacity property was ignored (always treated as 1). This has been fixed.

## FrameworkElementAutomationPeer ctor with null parameter throws NullReferenceException

In Silverlight 2, the **FrameworkElementAutomationPeer** ctor with a null parameter threw a **NullReferenceException**. Now this throws **ArgumentNullException**.

## HyperlinkButton now navigates the entire window and not just the IFRAME that Silverlight is in

In Silverlight 2, HyperlinkButton behaved inconsistently across browsers. In Internet Explorer, HyperlinkButton navigated the entire browser, not just the current IFrame.

In other browsers, HyperlinkButton navigates only the current IFrame, which is consistent with HTML behavior. In Silverlight 3 Beta, the behavior inside Internet Explorer was changed to match the behavior in other browsers.

## Correct base URI used when Silverlight v2+ application does not have XAP file extension

With the Silverlight 2 plug-in, if you renamed a .xap file to have a different extension, Silverlight erroneously thought the application was designed for Silverlight 1, and Silverlight ran it in version 1 compat mode. One impact was the base URI was considered to be the HTML page, not the (renamed) .xap. Another impact was that the default font was different from a .xap. This has been fixed.

## Application.Current.Host.Source now returns an absolute URL even when .xap URL has a query string

In Silverlight 2, if the URL the .xap was loaded from contained a query string, Application.Current.Host.Source could return a relative URL. In Silverlight 3, this has been fixed and Application.Current.Host.Source always returns an absolute URL. The URL of the .xap is typically specified as a <param> within the <object> tag, the following is an example of a .xap URL with a query string:

<param name="source" value="Bin/Debug/bug24857.xap**?a=b**"/>

## ResourceDictionary enum bug fixed

In Silverlight 2, if you put an enum into a **ResourceDictionary** and take it out again, you get back an object of type Integer32 rather than something of the expected enum type. This has been fixed for Silverlight 3.

## ReadOnlyObservableCollection moved from System.Windows.Controls.Data.dll to System.Windows.dll

The **ReadOnlyObservableCollection** class has been moved into **System.Windows.dll** so that other APIs in System.Windows.dll can use it. In order to minimize download size impact, some uncommonly used methods were removed.

Note that this will not affect applications compiled for Silverlight 2. If you use ReadOnlyObservableCollection in Silverlight 2, your .xap file will contain a copy of System.Windows.Controls.Data.dll. When Silverlight 3 Beta runs your .xap, it will use the .xap's copy of System.Windows.Controls.Data.dll and that version of ReadOnlyObservableCollection, not the version that ships in the System.Windows.dll of Silverlight 3 Beta.

## DataGridEndingEditEventArgs was removed from the SDK

There was no event that returns this **EventArgs**, so it was removed.

## PollingDuplexHttpBinding Changes

**PollingDuplexHttpBinding** now derives from **HttpBinding** instead of **BasicHttpBinding**. **PollingDuplexHttpBinding** now uses SOAP 1.2 instead of 1.1. As a result, Silverlight 3 **PollingDuplexHttpBinding** will not work with ASP.NET servers running the Silverlight 2 version of **System.ServiceModel.PollingDuplex.dll**.

## Bug fix with removing event handlers

Normally, when one removes an event listener (-= in C#), they remove a particular delegate for a particular event. Silverlight 2 ignored the event part – “foo.Bar -= Baz” would remove all event listeners tied to the Baz method, including events other than Bar. In Silverlight 3 this has been fixed.

## Bug fix allowing <Cursors> instead of <Cursor>

In Silverlight 2, the following was allowed:

 <DiscreteObjectKeyFrame.Value>

 <Cursors>Hand</Cursors> <-------note plural

 </DiscreteObjectKeyFrame.Value>

This was incorrect since the name of the type is Cursor not Cursors. Silverlight 3 only allows <Cursor>.

## Custom attached properties in XAML require xmlns are prefixed

In Silverlight 2, the following illegal XAML was accepted by Silverlight:

 <local:MyControl MyControl.DesignerID="12" />

This has been fixed in Silverlight 3. The correct XAML uses an XML prefix on the attached property:

 <local:MyControl **local:**MyControl.DesignerID="12" />

## Bug fix where subclass control could load wrong resource

In Silverlight 2, under certain conditions, if class A is defined in one user assembly, and markup inherits from class B in a different user code assembly, Application.LoadComponent(A) will try to load XAML from B's assembly instead of A's assembly. This has been fixed.

## Bug fix where ValueConverter is not called when leaf object on the path is null

Silverlight 2 data binding is inconsistent about how it handle nulls. If you specify Path=a.b.c.d and a, b, or c is null, Silverlight 2 calls the ValueConverter. However, if you specify Source=null, that does not call the value converter, even though it logically means the same thing. Silverlight 3 will always call the value converter.

## Thumb control now calculates drag relative to the absolute position rather than relative to its parent

This is most noticeable if the parent element is moving as a result of the drag.

## Setting Binding properties after SetBinding() no longer supported

In Silverlight 2, it's possible to programmatically set various properties on Binding between when SetBinding() is called but before DataContext is set, eg:

 Binding binding = new Binding();

 myElement.SetBinding(HeightProperty, binding);

 binding.Foo = value;

 myElement.DataContext = …;

Different properties had different behaviors in SL2:

* Converter, ConverterParameter, ConverterCulture, NotifyOnValidationError, and ValidatesOnExceptions properties would retroactively change the binding established with SetBinding().
* Mode and Source properties would be ignored (their values were copied when SetBinding() was called)

Silverlight 3 throws an exception if these properties are changed after SetBinding() is called. (Silverlight 2 will throw a similar exception if these properties are changed after DataContext is set)

## Child of a ContentPresenter can now only have one parent

In Silverlight 2, under certain circumstances the content (child element) inside a ContentPresenter can actually have two parents, because the content has been added to a second ContentPresenter without removing it from the first ContentPresenter. Silverlight 3 will throw an exception if an element is added to a second parent without removing it from the first parent.

## Relative URLs pointing to resources outside the .xap are now relative to the .xap file not the HTML page

Suppose the HTML page & the xap file are in different directories (e.g. http://example.com/ and http://example.com/ClientBin/), and you have a relative URL to a resource (eg images/foo.jpg) that is not contained within the xap file. SL2 would interpret that URL relative to the HTML page's location (http://example.com/images/foo.jpg). In order to make it easier to move Silverlight apps to different locations, Silverlight 3 interprets the URL relative to the .xap’s location (http://example.com/ClientBin/images/foo.jpg).

<section 5 -- upgrade breaking changes/quirks mode>

## Silverlight 3 automatically displays validation errors

In Silverlight 2, the default control templates were not aware of validation errors. In Silverlight 3, they are. Applications that display their own UI for validation errors (typically by listening to FrameworkElement.BindingValidationError) can turn off the built-in validation UI by re templating the affected control and removing the ValidationStates VisualStateGroup. You can do this in Blend by right clicking on the control, selecting “Edit Template” then "Edit a copy…", then give it a name, then go to xaml view and remove the following from the <Style>:

 <VisualStateGroup x:Name="ValidationStates">

 <VisualState x:Name="Valid">

 ...

 </VisualState>

 <VisualState x:Name="InvalidUnfocused">

 ...

 </VisualState>

 <VisualState x:Name="InvalidFocused">

 ...

 </VisualState>

 </VisualStateGroup>

## Image.Source and ImageBrush.Source now return null when not initialized

In Silverlight 2, getting Image/ImageBrush.Source before the property was set would create an empty BitmapImage. In Silverlight 3, if the property has not been set, it will return null. For example, if ImageBrush.Source has not been set, the following line of code will return null in Silverlight 3:

elementWithImageBrush.Fill.GetValue(ImageBrush.ImageSourceProperty)