Architecture

# Overview:

There’re 4 main components in the solution: A Windows Phone client application, a Web Role, a Worker Role, and a native component.

Windows Phone

Web Role

Worker Role

Native Component

Blob

Queue

As illustrated in the above diagram, the Windows Phone client application connects to the web role (to upload stories to the cloud). The web role and the worker role do not communicate with each other directly. Instead, they communicate via a queue. Blob storage is used heavily in the solution. The web role and worker role access blob directly using the storage account key, while the phone client accesses it via SAS.

Table storage is also used briefly in web/worker role. We did not illustrate it to save some space. Finally, the worker role uses a native component to encode the video.

In the future, we may create a separate managed transition library project and a separate native transition library project. But in this version, we put the managed transition library inside the Windows Phone project, and put the native transition library inside the native component project.

# Windows Phone client:

The Windows Phone client application is the main application end user uses. Below is a list of its main components:

## Pages

* MainPage: The home page. It contains buttons that allow users to create new stories, or open existing stories.
* ComposePage: The most important page in the application. Here users can compose the story.
* PreviewPage: Users can use this page to preview the story before it’s uploaded to the cloud and encoded into video. This page uses Silverlight animation to provide the preview. It also delegates transition preview to the corresponding transition class.
* ChooseStoryPage: A simple page that displays a list of saved stories, and allow users to select one to work on.
* ChooseMediaPage: This page is displayed when users want to add new photos to the current story. It uses Windows Phone MediaLibrary API to query the phone’s photo.

## Transition components:

* TransitionFactory: This component creates concrete transition classes from their names. But it returns ITranstion rather than the concrete class.
* ITransition: The interface that is used by the main application components and the transition library.
* TransitionBase: An abstract class that provides some default implementation of ITransition.
* FadeTransition: A sample transition that implements a simple fade effect.
* FlyInTransition: A sample transition that implements a fly-in effect.
* FlyInTransition\_Design: A sample design surface that corresponds to the fly-in transition. It provides additional design surface which will be blended in the compose page.

## Other components:

* StoryServiceLocator: A service locator that is responsible for invoking the cloud service.
* IsolatedStorageHelper: A class that interacts with Windows Phone isolated storage.
* PersistenceHelper: A class that is responsible for serializing/deserializing the stories.
* BitmapHelper: A class that helps to resize bitmaps.

There’re also various value converters, models, and view models.

# Web Role:

The web role contains two main components:

* StoryService: A REST service built with WCF Web API. It provides features to upload the stories, as well as list the encoded videos.
* Story viewer: A simple HTML5 page that invokes the REST service to list encoded stories, and use HTML5 video tag to render the encoded videos.

# Worker Role:

The worker role is used to encoded videos. While it delegates the actual encoding task to the native component, it handles all interactivities with Azure storage.

# Native component:

The native component is responsible for encoding the video. It contains the following main components:

* VideoEncoder: The main component that encodes the video. It is also responsible for loading the photo bitmap files. We use Media Foundation to encode the video. In particular, we use the new Media Foundation feature shipped with Windows 7 and Windows Server 2008 R2: sink writer.
* XmlParser: A simple homebrew xml parser. It is used to parse the story configuration file.
* Transition related components: Similar to the transition library in the Windows Phone project. But they’re implemented in native code.