Intel Unite®

SDK

Reference Design for Displaying Images

Guide

August 2019

Revision 1.0

Intel Confidential

Revision History

|  |  |  |
| --- | --- | --- |
| Revision | Description | Date |
| 1.0 | Initial release | August 2019 |
|  |  |  |
|  |  |  |
|  |  |  |

**Legal Disclaimers and Copyrights**

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel, the Intel logo, Intel® Core™ vPro™ processor family are trademarks of Intel Corporation in the United States and/or other countries.

\*Other names and brands may be claimed as the property of others

Contents

[1 Terminology 4](#_Toc19702271)

[2 UC Requirements 5](#_Toc19702272)

[2.1 Display WCs 5](#_Toc19702273)

[2.1.1 MVVM files 5](#_Toc19702274)

[2.1.2 Implement Factory Strategy 8](#_Toc19702275)

[2.1.4 Allocate HubView.Type 9](#_Toc19702276)

Figures

[Figure 2 Add View 6](#_Toc19702277)

[Figure 3 View Design Example 7](#_Toc19702278)

Code

[Code 1 WCsDisplay View Model 6](#_Toc19702279)

[Code 2 WCsView xaml 6](#_Toc19702280)

[Code 3 WCsDisplay.cs 7](#_Toc19702281)

[Code 4 enum HubView.Type 8](#_Toc19702282)

[Code 5 WCsDisplayFactory 9](#_Toc19702283)

[Code 6 Allocate HubView Example 9](#_Toc19702284)

# Terminology

Table 1. Terminology

| Term | Description |
| --- | --- |
| UI | User Interface |
| WPF | Windows\* Presentation Foundation |
| Hub | Intel® Core™ vPro™ processor family-based PC running the Intel Unite Hub application |
| Client | Device used to connect to the hub |
| Server | Device that manages access and configuration of Hub and Client Devices |
| Partial Background View | Area on the background layer where applications can reside |
| WC | Web Conference |
| UC | Unified Communications |
| UC User | Unified Communication User |

## Displaying Images within Unite

Images can be worth a 1000 words but getting them displayed correctly can be a chore. This primer will cover a few basics when wanting to use Images with Unite.

Unite has and uses a unique image type called a UniteImage – this object has some differences when attempting to create. In this primer we will be using PNG extensions of our images, but UniteImage supports other image extensions such as JPG.

Status and Auth Views – require using an UniteImage. Attempting to use a normal Image regardless of its extension will result in no image being displayed in these views.

Clients – Inside Unite’s abstract class: Module Base is a UniteImage named ModuleImage and this is the Image used by Intel Unite Clients. If this UniteImage is not set during the Load method in your plugin’s ModuleHandler, then it is possible that a Client could have an empty Image in their application.

### Image as an Embedded Resource

When setting Images to be an Embedded Resource, it is important to pay attention to the physical directory structure inside Visual Studio of the images because the path uses a period (.) as a separator instead of slashes (/).

Looking at the target path, you can see that we are referring to the Project Name. Then “Folder Name.Name” of resource as PNG.

Code - Using as an Embedded Resource

|  |
| --- |
| /// <summary>  /// creates a UniteImage from and Embedded Resouce  /// </summary>  /// <param name="resoucePath"></param>  /// <param name="uniteImageType" cref="UniteImageType"></param>  /// <returns cref="UniteImage"></returns>  /// <example>  /// <code>  /// UniteImage image = GetUniteImageFromEmbeddedResource("ProjectName.FolderName.recording-icon.png", UniteImageType.Png);  /// </code>  /// </example>  protected UniteImage GetUniteImageFromEmbeddedResource(string resoucePath, UniteImageType uniteImageType)  {  return Intel.Unite.Common.Utils.BytesHelper.SetImageFromResource(Guid.NewGuid(),  uniteImageType, resoucePath, Assembly.GetExecutingAssembly());  } |

### Image as a Resource

When setting Images to be a Resource, it is also important to pay attention to the directory structure of the images because the path uses a slash (/) as one would expect.

Code - Using as a Resource

|  |
| --- |
| /// <summary>  /// creates a UniteImage from and Resource  /// </summary>  /// <param name="resoucePath"></param>  /// <param name="uniteImageType" cref="UniteImageType"></param>  /// <returns cref="UniteImage"></returns>  /// <example>  /// <code>  /// var Image = GetUniteImageFromResouce("/Namespace;component/Images/recording-icon.png", UniteImageType.Png);  /// </code>  /// </example>  protected UniteImage GetUniteImageFromResouce(string resourcePath, UniteImageType uniteImageType)  {  var resourceLocater = new Uri(resourcePath, System.UriKind.Relative);  var resourceInfo = Application.GetResourceStream(resourceLocater);  using (var memoryStream = new MemoryStream())  {  resourceInfo.Stream.CopyTo(memoryStream);  var imgObj = System.Drawing.Image.FromStream(memoryStream);  var size = new UniteDisplayRect { Height = imgObj.Size.Height, Width = imgObj.Size.Width };  return new UniteImage  {  Id = Guid.NewGuid(),  Data = memoryStream.ToArray(),  DataType = uniteImageType,  Size = size,  };  }  } |

### ImageViewBase

Unite Hub requires the Status and Auth Views to be Allocated differently in that the UniteImage is to be used as a parameter. When using a StatusView or AuthView ensure that UI components inherit from the ImageViewBase class.

1. In the **UI** folder, create a file name ImageViewBase.cs

Code 3 - ImageViewBase.cs contents

|  |
| --- |
| public class ImageViewBase : ViewBase  {  [field: NonSerialized]  private UniteImage \_image;  protected UniteImage Image { get => \_image; set => \_image = value; }  protected ImageViewBase(IHubModuleRuntimeContext runtimeContext, PhysicalDisplay display, Dispatcher currentUiDispatcher, Func<FrameworkElement, MarshalNativeHandleContract> createContract) :  base(runtimeContext, display, currentUiDispatcher, createContract)  { }  public override void Allocate()  {  RuntimeContext.DisplayManager.AllocateUiInHubDisplayAsync(  this.Image,  HubAllocationInfo,  AllocatedCallBack  );  }  public override void DeAllocate()  {  RuntimeContext.DisplayManager.DeallocateUiFromHubDisplayAsync(  DisplayView,  DeallocateCallBack  );  }  public override bool Show()  {  if (!IsAllocated) Allocate();  CurrentUiDispatcher.BeginInvoke(DispatcherPriority.ApplicationIdle, new Action(() => { })).Wait();  return IsAllocated && RuntimeContext.DisplayManager.ShowAllocatedUi(DisplayView);  }  public override void AllocatedCallBack(HubAllocationResult hubAllocationResult)  {  if (hubAllocationResult.Success)  {  AllocatedSuccess(hubAllocationResult.AllocatedView);  }  else  {  AllocatedFail();  throw new Exception(hubAllocationResult.ResultType.ToString());  }  }  public override bool AllocatedFail()  {  lock (this)  {  }  return true;  }  public override void AllocatedSuccess(DisplayView allocatedView)  {  lock (this)  {  DeepCopy.CopyDisplayView(DisplayView, allocatedView);  }  }  public override void DeallocateCallBack(HubAllocationResult hubAllocationResult)  {  if (hubAllocationResult.Success)  {  lock (this)  {  SetDeallocate();  }  }  else  {  throw new Exception(hubAllocationResult.ResultType.ToString());  }  }  private void SetDeallocate()  {  DisplayView.Id = new Guid();  }    // Removed the two GetUniteImage helper methods from above to save space  } |

Note that the second parameter of the Allocate method is the class UniteImage, instead of a DisplayView.

Code 4 - Allocate method

|  |
| --- |
| public override void Allocate()  {  RuntimeContext.DisplayManager.AllocateUiInHubDisplayAsync(  this.Image,  HubAllocationInfo,  AllocatedCallBack  );  } |