

AMC BRIDGE

Autodesk Vault Professional



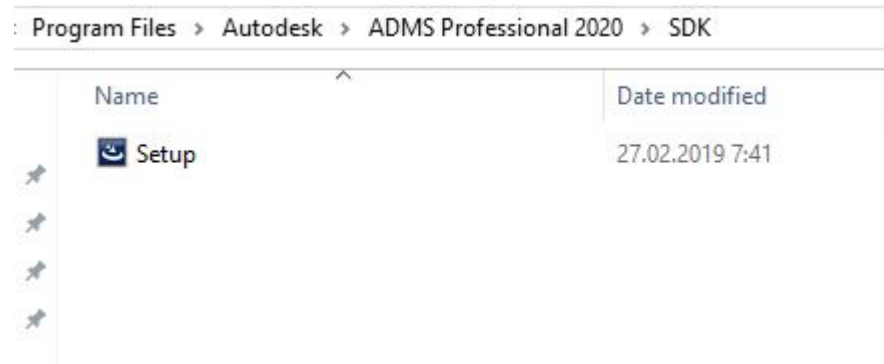
- ✓ Vault SDK
- ✓ Installing Vault SDK
- ✓ Vault Developer Framework (VDF)
- ✓ Adapt Vault with API
- ✓ Three interface Vault API
- ✓ How to customize Vault Explorer
- ✓ The Vault API assembly/DLL
- ✓ Examples of the use of Vault API
- ✓ Autodesk Developer Network

Vault isn't just a single program. It's a framework, composed of many pieces working together. Some of these pieces are customizable and some are not. The rules for customizing one piece may be different than the rules for another piece.

The SDK component contains useful tools, such as documentation on all classes and functions, application examples, knowledge base articles, utilities, error code descriptions, and changes from previous versions. It has everything you need to start creating your own applications for Vault.

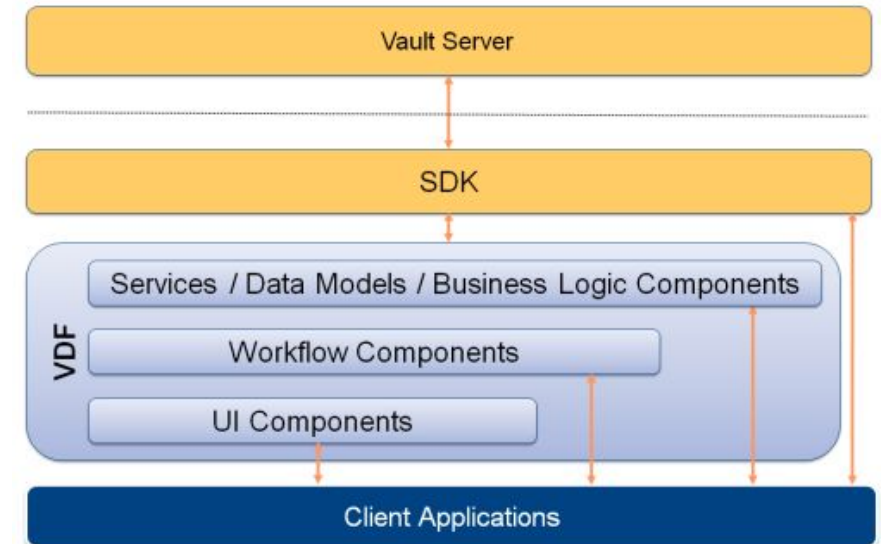


This component is automatically installed when installing the server part of Vault. It is located in the SDK folder inside the ADMS folder in the location specified during installation.



VDF is a high-level structure on top of an existing API that provides:

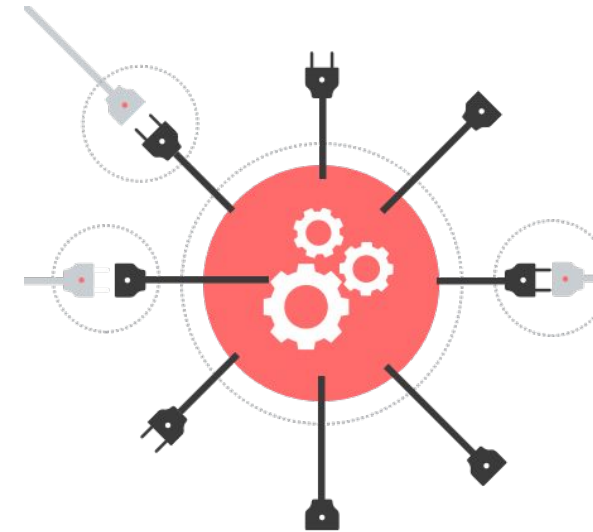
- Reusable business logic for common storage algorithms
- Reusable GUI controls for common workflows
- Expandable components that can be customized to meet host requirements



```
}  
VDF.Vault.Currency.Connections.Connection connection = e.Context.Application.Connection;
```

Adapt Vault with API

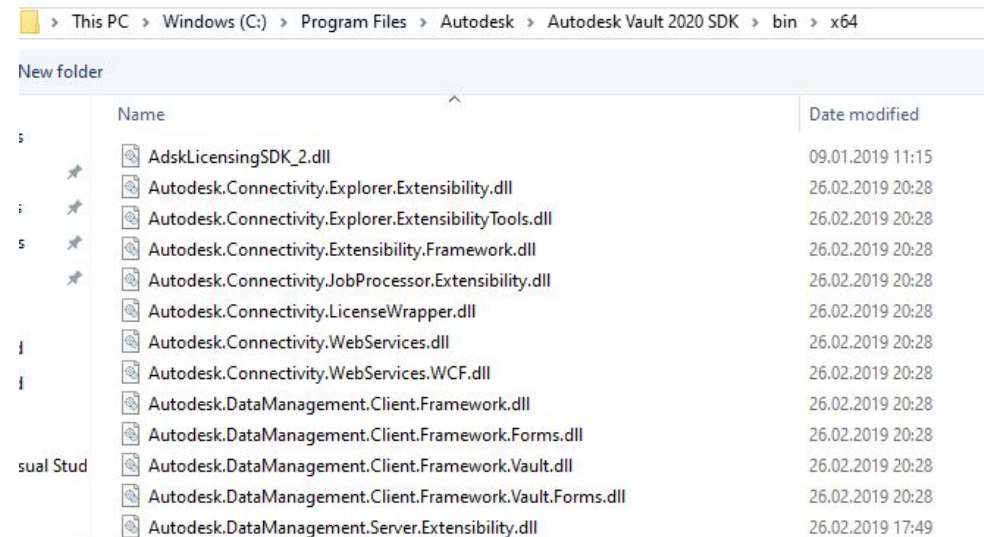
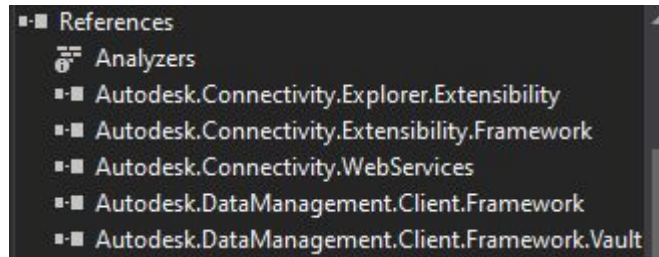
Possessing programming skills, the user gets a large number of opportunities to adapt the system using the application programming interface (API) Vault. Common adaptations include utilities, automatic processing and integration with other systems, commands and data views.



- ❑ **The Web Services API** comprised of the Autodesk.Connectivity.WebServices DLL. The DLL enables you to communicate with the server.
- ❑ **The Vault Development Framework (VDF)** comprised of the Autodesk.DataManagement DLLs. The VDF is a high-level framework that provides reusable business logic for common vault algorithms as well as reusable GUI controls for common workflows, simplifying add-in and client development.
- ❑ **The Extensibility Libraries** are comprised of the Autodesk.Connectivity.Explorer.Extensibility and Autodesk.Connectivity.JobProcessor.Extensibility DLLs. They allow you to plug into the Vault Explorer and Job Processor applications respectively.

How to customize Vault Explorer

The first thing is to set up your project in Visual Studio.
Add library project (DLL) and it has to use the .NET
framework.



The next step is to create a public class which implements the `IExplorerExtension` interface. This step requires you to fill in the logic for a bunch of functions. Depending on what you want to do you may be able to leave some of the functions empty or just return null.

```
...public interface IExplorerExtension
{
    IEnumerable<CommandSite> CommandSites();
    IEnumerable<CustomEntityHandler> CustomEntityHandlers();
    IEnumerable<DetailPaneTab> DetailTabs();
    IEnumerable<string> HiddenCommands();
    void OnLogOff(IApplication application);
    void OnLogOn(IApplication application);
    void OnShutdown(IApplication application);
    void OnStartup(IApplication application);
}
```

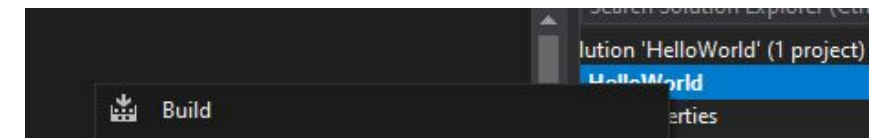
There are 5 assembly attributes you need to have in your code:

- Assembly Company - The name of your company.
- Assembly Product - The name of the product.
- Assembly Description - A description of your product.
- Extension Id - A unique GUID for your product. Visual Studio has a tool that can generate one for you. Do not copy a GUID from another extension.
- API Version - A string indicating which API schema your product works with. Enter in "13.0" to work with Vault 2020.

```
[assembly: AssemblyCompany("Autodesk")]  
[assembly: AssemblyProduct("HelloWorldCommandExtension")]  
[assembly: AssemblyDescription("Sample App")]  
[assembly: ExtensionId("7ADC0766-F085-46d7-A2EB-C68F79CBF4E7")]  
[assembly: ApiVersion("13.0")]
```

How to customize Vault Explorer

Once you have everything coded and built, it's time to deploy it. To deploy you create a folder for your extension under %ProgramData%/Autodesk/Vault 2020/Extensions/. You can name your folder whatever you want, but it's probably best to name it the same as your product name.



> This PC > Windows (C:) > ProgramData > Autodesk > Vault 2020 > Extensions >

	Name	Date modified	Type	Size
	ACADEExtensionHandler	11.06.2019 11:24	File folder	
	ACADSynergyExtensionHandler	11.06.2019 11:24	File folder	
	Autodesk	26.06.2019 12:08	File folder	
	InsertFromVault	11.06.2019 11:24	File folder	
	InventorExtensionHandler	11.06.2019 11:24	File folder	
	NavisworksExtensionHandler	11.06.2019 11:24	File folder	
	RevitExtensionHandler	11.06.2019 11:24	File folder	

[Autodesk.Connectivity.Extensibility.Framework](#)

This assembly contains common classes and utilities which are shared across the entire Vault client framework.

[Autodesk.Connectivity.WebServices](#)

Provides a set of functions used to communicate with the Vault Server. The server functions are organized into services based on functionality.

[Autodesk.Connectivity.Explorer.Extensibility](#)

This assembly contains the needed hooks to write your own extensions to Vault Explorer

[Autodesk.Connectivity.Explorer.ExtensibilityTools](#)

This assembly provides access to Vault Explorer business logic. It provides a simple interface for calling complex functionality by re-using Vault Explorer components.

[Autodesk.Connectivity.JobProcessor.Extensibility](#)

This assembly contains the needed hooks to write your own extensions to Job Processor.

[Autodesk.DataManagement.Client.Framework](#)

Provides fundamental data management features and services.

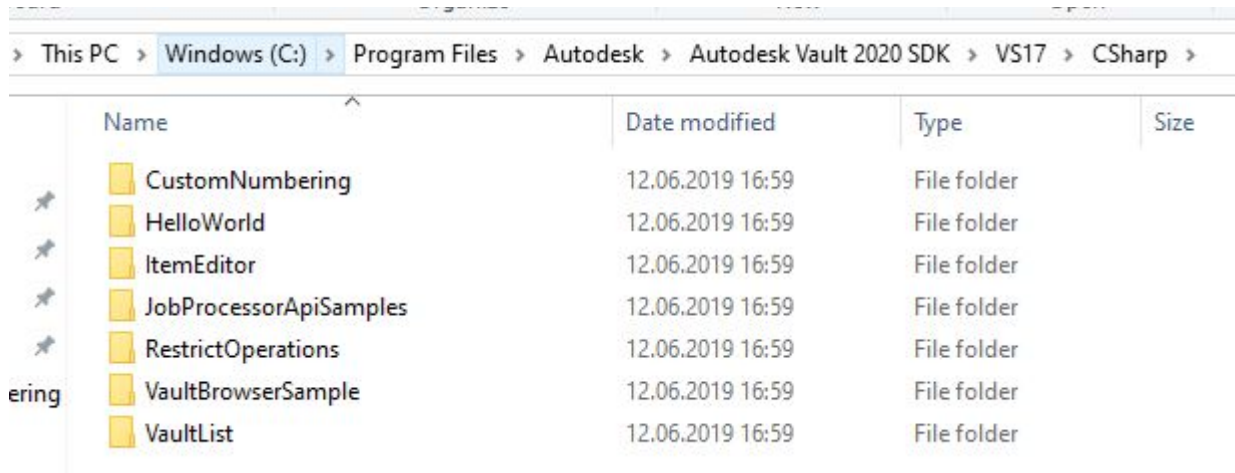
[Autodesk.DataManagement.Client.Framework.Forms](#)

Provides data management features and services which use GUI's to interact with users

[Autodesk.DataManagement.Client.Framework.Vault.Forms](#)

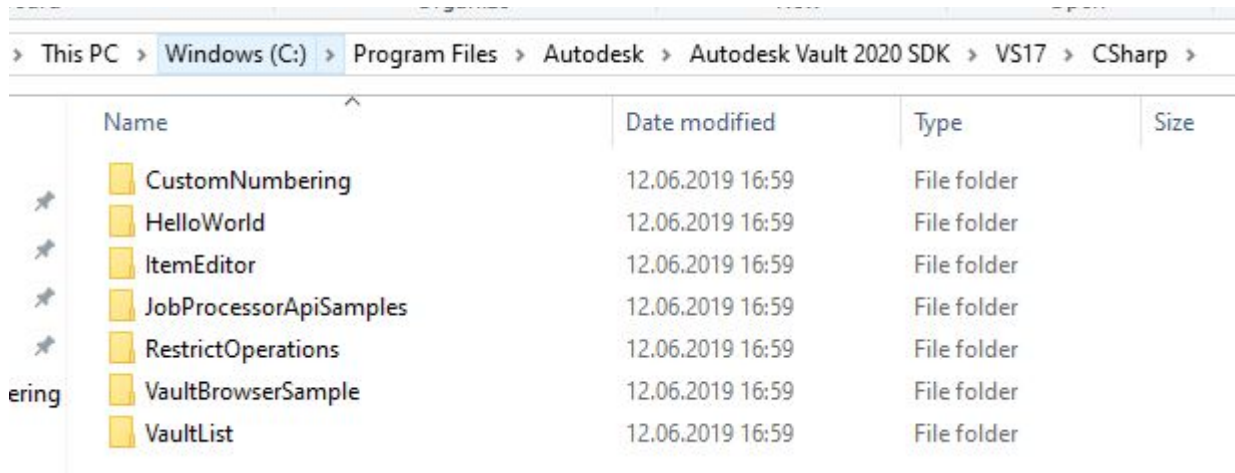
Provides data management features and services that are specific to Vault and use GUI's to interact with users.

For a simple example of how to configure Vault Explorer, SDK provides examples.



This PC > Windows (C:) > Program Files > Autodesk > Autodesk Vault 2020 SDK > VS17 > CSharp >				
	Name	Date modified	Type	Size
✦	CustomNumbering	12.06.2019 16:59	File folder	
✦	HelloWorld	12.06.2019 16:59	File folder	
✦	ItemEditor	12.06.2019 16:59	File folder	
✦	JobProcessorApiSamples	12.06.2019 16:59	File folder	
✦	RestrictOperations	12.06.2019 16:59	File folder	
ering	VaultBrowserSample	12.06.2019 16:59	File folder	
	VaultList	12.06.2019 16:59	File folder	

For a simple example of how to configure Vault Explorer, SDK provides examples.



This PC > Windows (C:) > Program Files > Autodesk > Autodesk Vault 2020 SDK > VS17 > CSharp >				
	Name	Date modified	Type	Size
✦	CustomNumbering	12.06.2019 16:59	File folder	
✦	HelloWorld	12.06.2019 16:59	File folder	
✦	ItemEditor	12.06.2019 16:59	File folder	
✦	JobProcessorApiSamples	12.06.2019 16:59	File folder	
✦	RestrictOperations	12.06.2019 16:59	File folder	
ering	VaultBrowserSample	12.06.2019 16:59	File folder	
	VaultList	12.06.2019 16:59	File folder	

```
0 references
public IEnumerable<CommandSite> CommandSites()
{
    // Create the Hello World command object.
    CommandItem helloWorldCmdItem = new CommandItem("HelloWorldCommand", "Hello World...")
    {
        // this command is active when a File is selected
        NavigationTypes = new SelectionTypeId[] { SelectionTypeId.File },

        // this command is not active if there are multiple entities selected
        MultiSelectEnabled = false
    };

    // The HelloWorldCommandHandler function is called when the custom command is executed.
    helloWorldCmdItem.Execute += HelloWorldCommandHandler;

    // Create a command site to hook the command to the Advanced toolbar
    CommandSite toolbarCmdSite = new CommandSite("HelloWorldCommand.Toolbar", "Hello World Menu")
    {
        Location = CommandSiteLocation.Advanced, CommandSite.CommandSite(string uniqueId, string label)
        DeployAsPulldownMenu = false
    };
    toolbarCmdSite.AddCommand(helloWorldCmdItem);

    // Create another command site to hook the command to the right-click menu for Files.
    CommandSite fileContextCmdSite = new CommandSite("HelloWorldCommand.FileContextMenu", "Hello World Menu")
    {
        Location = CommandSiteLocation.FileContextMenu,
        DeployAsPulldownMenu = false
    };
    fileContextCmdSite.AddCommand(helloWorldCmdItem);

    // Now the custom command is available in 2 places.

    // Gather the sites in a List.
    List<CommandSite> sites = new List<CommandSite>();
    sites.Add(toolbarCmdSite);
    sites.Add(fileContextCmdSite);

    // Return the list of CommandSites.
    return sites;
}
```

```
1 reference
void HelloWorldCommandHandler(object s, CommandItemEventArgs e)
{
    try
    {
        VDF.Vault.Currency.Connections.Connection connection = e.Context.Application.Connection;

        // The Context part of the event args tells us information about what is selected.
        // Run some checks to make sure that the selection is valid.

        if (e.Context.CurrentSelectionSet.Count() > 1)
            MessageBox.Show("This function does not support multiple selections");
        else
        {
            // we only have one item selected, which is the expected behavior

            ISelection selection = e.Context.CurrentSelectionSet.First();

            // Look of the File object. How we do this depends on what is selected.
            File selectedFile = null;
            if (selection.TypeId == SelectionTypeId.File)
            {
                // our ISelection.Id is really a File.MasterId
                selectedFile = connection.WebServiceManager.DocumentService.GetLatestFileByMasterId(selection.Id);
            }
            else if (selection.TypeId == SelectionTypeId.FileVersion)
            {
                // our ISelection.Id is really a File.Id
                selectedFile = connection.WebServiceManager.DocumentService.GetFileById(selection.Id);
            }

            if (selectedFile == null)
            {
                MessageBox.Show("Selection is not a file.");
            }
            else
            {
                // this is the message we hope to see
                MessageBox.Show(String.Format("Hello World! The file size is: {0} bytes",
                    selectedFile.FileSize));
            }
        }
    }
}
```

Examples of the use of Vault API



The Autodesk Developer Network (ADN) provides full API support in Vault.

AMC BRIDGE

303 Wyman Street, Suite 300
Waltham, MA 02451, USA

