Design Document

McMaster-Carr Add-In for Autodesk Inventor
Version 1.0.0 • September 30 2019

Nicholas Quandt
nicholas.quandt@marquette.edu

Marquette University Masters Student of
Computational Mathematical and Statistical Sciences

1 Introduction

The goal of this project is to create an interface within Autodesk Inventor 2019 that allows for more efficient usage of McMaster Carr's online catalog of 3d Models.

2 Languages and Packages

- C#
- XAML
- HTML
- Autodesk Inventor Interop
- CEFSharp v57.0

3 Overview of Requirements

These requirements are a summary of the user stories gathered from the developer and potential users of the add-in.

1. Runs in Windows 64-bit for Inventor 2019 software.
2. Use Inventor API to add Button for interface.
3. Use McMaster online catalog that has links to 3d models of the majority of their items. Primarily hardware,fasteners etc.
4. Best to keep online interface from McMaster, and somehow open the "form" in inventor to access.
5. Drag and drop from "McMaster interface" into inventor, for either immediate drop into Assembly, or open as individual part.
6. Replace "add to cart" button in HTML with "add to assembly" or "open as part" buttons.
7. Take product information from online catalog and automatically import into Inventor part file.
8. Implement a bill of materials export to send all McMaster items onto clipboard with appropriate QTY per assembly.

4 Implementation Plan

McMaster.com is a dynamically generated HTML page that only changes via user exposure to the javascript. A basic HTML scrape of the webpage will not be feasible due to the JS. Thus to gain exposure to the underlying HTML tags needed for properties and file locations, a headless copy of the browser will work to trick the webpage into generating the needed information while the user is interacting with the saving procedure for each part. So far it seems CEFSharp will be the best plan of attack as they offer a headless wrap of google chrome within their package. All processing on the Inventor side of things will be done through the built in .NET API. The browser and add-in will be fully stand alone within Inventor, with no outside installations needed. As is noted in the Notes section, CEFSharp is already built into Inventor 2019, making implementation easier.

5 User Interface

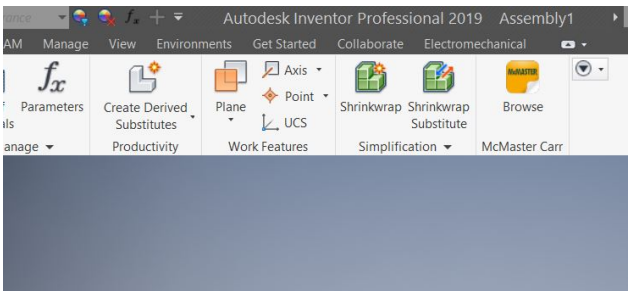

Figure 1: The button(far right) addition to the ribbon interface within Autodesk Inventor.

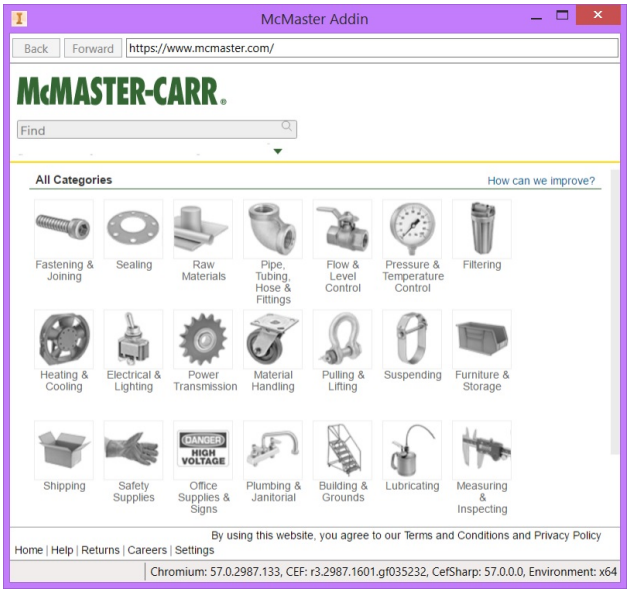

Figure 2: The browser instance following the button execute event.

Iteration 1 allows for the part number to be extracted from the browser form through the mouse hovering link. Iteration 2 aims to take this part number and capture the correct file from the mcmaster database for saving into project.(See Listing 1)

Listing 1: Excerpt from MainWindow.xaml.cs

```
currentPartNumber =
   HoverLinkBehaviour.HoverLink.Substring(urlBase.Length);
System.Diagnostics.Debug.WriteLine(currentPartNumber);
```

6 Extra Notes

- CEFSharp v57.0 specifically. v57.0 is built into Inventor 2019, though .WPF package is not, so that needs to be included in release of .dll's.
- Possible revisions after all other features:
  - Incorporate a "no model" warning for items that do not have corresponding 3d models.
  - Choose folder to save models per project, or per item.
  - Always on top.
  - Comments throughout code for future revising, if ever McMaster online changes.
  - If final release available, contact McMaster to check usability.
  - Solidworks Version??

7 References

- https://www.mcmaster.com[1]
- https://www.autodesk.com/products/inventor/overview
- https://docs.microsoft.com/en-us/dotnet/
- https://cefsharp.github.io/
- https://modthemachine.typepad.com/

[1] McMaster-Carr Supply Company, "McMaster-Carr," https://www.mcmaster.com.

8 Appendix A: Story Cards

101 As a user I want to be able to use Windows.
Acceptance Criteria: Runs on Windows 64-bit without errors.

102 As a user I want to use Autodesk Inventor 2019.
Acceptance Criteria: Uses Autodesk Inventor 2019 without errors.

103 As a user I want to have access to McMaster-Carr catalog from inside of Inventor.
Acceptance Criteria: No outside browser needed to access catalog.

104 As a user I don't want the interface to change much.
Acceptance Criteria: Uses original online interface.

105 As a user I should be able to add parts from mcmaster right into my project. Acceptance Criteria: No outside program or steps needed to import file.

106 As a user I would like properties like the material added to my file for me.
Acceptance Criteria: Properties are auto-populated on file addition.

107 As a developer I would like to use C#
Acceptance Criteria: All code written with C# on .NET 4.7.2 or newer

108 As a developer I shouldn't need to install too many files or packages on the final computer.
Acceptance Criteria: Keep total file count below 3.

109 As a developer I should well document code to allow for future updates.
Acceptance Criteria: An outside individual can proof read code and follow along without difficulty.

9 Appendix B: Code

StandardAddInServer.cs

Listing 2: Excerpt from MainWindow.xaml.cs

```
using System;
using System.IO;
using reflect = System.Reflection;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using System.Drawing;
using Inv = Inventor;
using Inventor;
using CefSharp.Wpf;
using CefSharp;

namespace McMasterAddin
{
  /// <summary>
  /// This is the primary AddIn Server class that implements the
  /// ApplicationAddInServer interface that all Inventor AddIns
  /// are required to implement. The communication between
     Inventor
  /// and the AddIn is via the methods on this interface.
  /// </summary>
  [GuidAttribute("4989fc73-4710-47df-9034-98d770e68fbb")]
  public class StandardAddInServer : ApplicationAddInServer
  {
    // Inventor application object.
    private Inv.Application m_invApp;

    #region ObjectInitialization

    //user interface event
    private UserInterfaceEvents m_UIEvents;
    private static GuidAttribute m_ClientID =
      (GuidAttribute)System.Attribute.GetCustomAttribute(
        typeof(StandardAddInServer), typeof(GuidAttribute));
    private static readonly string m_ClientIDstring =
      "{" + m_ClientID.Value + "}";

    //single button for McMaster Catalog
    private ButtonDefinition m_butDefinition;
    private ButtonDefinitionSink_OnExecuteEventHandler
      m_butDefinition_OnExecute_Delegate;

    private
       UserInterfaceEventsSink_OnResetCommandBarsEventHandler
        UIESink_OnResetCommandBarsEventDelegate;
```

```
    private
       UserInterfaceEventsSink_OnResetEnvironmentsEventHandler
        UIESink_OnResetEnvironmentsEventDelegate;
    private
       UserInterfaceEventsSink_OnResetRibbonInterfaceEventHandler
        UIESink_OnResetRibbonInterfaceEventDelegate;

    #endregion

    public StandardAddInServer()
    {
      //Keep empty
    }

    #region ApplicationAddInServer Members

    public void Activate(ApplicationAddInSite addInSiteObject,
       bool firstTime)
    {
      try
      {
        //the Activate method is called by Inventor when it
           loads the addin
        //the AddInSiteObject provides access to the Inventor
           Application
        //object the FirstTime flag indicates if the addin is
           loaded for the
        //first time

        //initialize AddIn members
        m_invApp = addInSiteObject.Application;

        ControlDefinitions controlDefs =
          m_invApp.CommandManager.ControlDefinitions;

        //initialize event delegates
        m_UIEvents =
           m_invApp.UserInterfaceManager.UserInterfaceEvents;

        UIESink_OnResetCommandBarsEventDelegate = new
          UserInterfaceEventsSink_OnResetCommandBarsEventHandler(
            UIE_OnResetCommandBars);
        m_UIEvents.OnResetCommandBars +=
          UIESink_OnResetCommandBarsEventDelegate;

        UIESink_OnResetEnvironmentsEventDelegate = new
          UserInterfaceEventsSink_OnResetEnvironmentsEventHandler(
            UIE_OnResetEnvironments);
        m_UIEvents.OnResetEnvironments +=
```

```
    UIESink_OnResetEnvironmentsEventDelegate;

UIESink_OnResetRibbonInterfaceEventDelegate = new
  UserInterfaceEventsSink_OnResetRibbonInterfaceEventHandler(
    UIE_OnResetRibbonInterface);
m_UIEvents.OnResetRibbonInterface +=
  UIESink_OnResetRibbonInterfaceEventDelegate;

Stream myStream = reflect.Assembly.GetExecutingAssembly()
  .GetManifestResourceStream("McMasterAddin.Resources.mcmaster.ico")

stdole.IPictureDisp largeImage =
  PictureDispConverter.ToIPictureDisp(new
     Icon(myStream));

//Button definition
m_butDefinition =
   controlDefs.AddButtonDefinition("Browse",
  "BrowseButton",
  CommandTypesEnum.kQueryOnlyCmdType, m_ClientIDstring,
  "Browse McMaster-Carr Inventory", "Use this to find " +
  "hardware and other products available on
     McMaster.com",
  largeImage, largeImage,
     ButtonDisplayEnum.kAlwaysDisplayText);

m_butDefinition_OnExecute_Delegate =
  new
     ButtonDefinitionSink_OnExecuteEventHandler(m_but_OnExecute);
m_butDefinition.OnExecute +=
   m_butDefinition_OnExecute_Delegate;
m_butDefinition.Enabled = true;


if (firstTime == true)
{
  //access user interface manager
  UserInterfaceManager UIManager =
     m_invApp.UserInterfaceManager;

  //create the UI for classic interface
  if (UIManager.InterfaceStyle ==
     InterfaceStyleEnum.kClassicInterface)
  {
    //For first iterations assume RibbonInterface
  }
  //create the UI for ribbon interface
  else if (UIManager.InterfaceStyle ==
    InterfaceStyleEnum.kRibbonInterface)
```

```
        {
          CreateOrUpdateRibbonUserInterface();
        }

      }

    }
    catch (Exception e)
    {
      MessageBox.Show(e.ToString());
    }

    //Keep CEF on until INVENTOR exits, not just the WPF form.
    CefSharpSettings.ShutdownOnExit = false;

    var settings = new CefSettings();

    //Example of setting a command line argument
    //Enables WebRTC
    settings.CefCommandLineArgs.Add("enable-media-stream",
       "1");
    //Must call once on main thread, and shutdown on main
       thread.
    Cef.Initialize(settings);
  }

  public void Deactivate()
  {
    //Need to call on main thread
    Cef.Shutdown();
  }

  public void ExecuteCommand(int commandID)
  {
    // Note:this method is now obsolete, you should use the
    // ControlDefinition functionality for implementing
       commands.
  }

  public object Automation
  {
    // This property is provided to allow the AddIn to expose
       an API
    // of its own to other programs. Typically, this  would be
       done by
    // implementing the AddIn's API interface in a class and
       returning
    // that class object through this property.
```

```
    get
    {
      // TODO: Add ApplicationAddInServer.Automation getter
         implementation
      return null;
    }
  }

  private void CreateOrUpdateRibbonUserInterface()
  {
    UserInterfaceManager UIManager =
       m_invApp.UserInterfaceManager;
    Ribbon assemblyRibbon = UIManager.Ribbons["Assembly"];
    RibbonTab assembleTab =
       assemblyRibbon.RibbonTabs["id_TabAssemble"];
    RibbonPanel mcMasterPanel =
      assembleTab.RibbonPanels.Add("McMaster Carr",
        "McMasterPanel", m_ClientIDstring);

    mcMasterPanel.CommandControls.AddButton(m_butDefinition,
       true);
  }

  private void UIE_OnResetCommandBars(
    ObjectsEnumerator commandBars, NameValueMap context)
  {
    UserInterfaceManager UIManager =
       m_invApp.UserInterfaceManager;
    Inv.Environment asmEnvironment =
       UIManager.Environments["AMxAssemblyEnvironment"];

    foreach (CommandBar commandBar in commandBars)
    {
      if (commandBar ==
         asmEnvironment.PanelBar.DefaultCommandBar)
      {

      }
    }
  }

  private void UIE_OnResetEnvironments(
    ObjectsEnumerator environments, NameValueMap context)
  {
    try
    {
      Inv.Environment environment;
      for (int environmentCt = 1;
        environmentCt <= environments.Count; environmentCt++)
```

```
        {
          environment =
             (Inv.Environment)environments[environmentCt];
          if (environment.InternalName ==
             "PMxPartSketchEnvironment")
          {
            //make this command bar accessible in the
            //panel menu for the 2d sketch environment.
            environment.PanelBar.CommandBarList.Add(
              m_invApp.UserInterfaceManager.CommandBars[
                "Autodesk:SimpleAddIn:SlotToolbar"]);

            return;
          }
        }
      }
      catch (Exception e)
      {
        MessageBox.Show(e.ToString());
      }
    }

    private void UIE_OnResetRibbonInterface(NameValueMap context)
    {
      CreateOrUpdateRibbonUserInterface();
    }

    /// <summary>
    /// This is the mcMaster-addin button execution
    /// </summary>
    /// <param name="Context"></param>
    void m_but_OnExecute(NameValueMap Context)
    {
      //Refer to MainWindow.xaml for code of the browser
         extension
      var wpfWindow = new McMasterAddin.MainWindow();
      //This allows for a WPF control to be displayed without
      //the need of a fullfledge WPF Application.
      var helper = new
         System.Windows.Interop.WindowInteropHelper(wpfWindow);
      helper.Owner = new IntPtr(m_invApp.MainFrameHWND);

      //Show modal Even though button executions
      //seem to start their own threads.
      wpfWindow.ShowDialog();
    }

    #endregion
```

```
}

public sealed class PictureDispConverter

{
  [DllImport("OleAut32.dll",
      EntryPoint = "OleCreatePictureIndirect",
      ExactSpelling = true,
      PreserveSig = false)]

  private static extern stdole.IPictureDisp
      OleCreatePictureIndirect(
          [MarshalAs(UnmanagedType.AsAny)] object picdesc,
          ref Guid iid,
          [MarshalAs(UnmanagedType.Bool)] bool fOwn);

  static Guid iPictureDispGuid =
     typeof(stdole.IPictureDisp).GUID;

  private static class PICTDESC
  {
    //Picture Types
    public const short PICTYPE_UNINITIALIZED = -1;
    public const short PICTYPE_NONE = 0;
    public const short PICTYPE_BITMAP = 1;
    public const short PICTYPE_METAFILE = 2;
    public const short PICTYPE_ICON = 3;
    public const short PICTYPE_ENHMETAFILE = 4;

    [StructLayout(LayoutKind.Sequential)]
    public class Icon
    {
      internal int cbSizeOfStruct =
          Marshal.SizeOf(typeof(PICTDESC.Icon));
      internal int picType = PICTDESC.PICTYPE_ICON;
      internal IntPtr hicon = IntPtr.Zero;
      internal int unused1;
      internal int unused2;

      internal Icon(System.Drawing.Icon icon)
      {
        this.hicon = icon.ToBitmap().GetHicon();
      }
    }

    [StructLayout(LayoutKind.Sequential)]
    public class Bitmap
    {
      internal int cbSizeOfStruct =
```

```
            Marshal.SizeOf(typeof(PICTDESC.Bitmap));

        internal int picType = PICTDESC.PICTYPE_BITMAP;
        internal IntPtr hbitmap = IntPtr.Zero;
        internal IntPtr hpal = IntPtr.Zero;
        internal int unused;

        internal Bitmap(System.Drawing.Bitmap bitmap)
        {
          this.hbitmap = bitmap.GetHbitmap();
        }
      }
    }

    public static stdole.IPictureDisp ToIPictureDisp(
        System.Drawing.Icon icon)
    {
      PICTDESC.Icon pictIcon = new PICTDESC.Icon(icon);

      return OleCreatePictureIndirect(
          pictIcon, ref iPictureDispGuid, true);
    }

    public static stdole.IPictureDisp ToIPictureDisp(
        System.Drawing.Bitmap bmp)
    {

      PICTDESC.Bitmap pictBmp = new PICTDESC.Bitmap(bmp);

      return OleCreatePictureIndirect(pictBmp, ref
         iPictureDispGuid, true);
    }
  }
}
```

MainWindow.xaml.cs

Listing 3: Excerpt from MainWindow.xaml.cs

```
using System.Windows;

namespace McMasterAddin
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }


        private readonly string urlBase =
           "https://www.mcmaster.com/";
        private string currentPartNumber = "#";

        /// <summary>
        /// This will send the part number information,
        /// to the main code in order to
        /// download a .STEP file from mcmaster.com and
        /// translate into a .IPT to open into inventor.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Browser_PreviewMouseDown(object sender,
           System.Windows.Input.MouseButtonEventArgs e)
        {
            if (e.ChangedButton ==
               System.Windows.Input.MouseButton.Left)
            {
                try
                {
                    currentPartNumber =
                       HoverLinkBehaviour.HoverLink.Substring(urlBase.Length
                    System.Diagnostics.Debug.WriteLine(currentPartNumber);
                }
                catch { }
            }
        }
    }
}
```