Design Document

McMaster-Carr Add-In for Autodesk Inventor
Version 1.0.2 • October 28 2019

Nicholas Quandt
nicholas.quandt@marquette.edu

Marquette University Masters Student of
Computational Mathematical and Statistical Sciences

1 Overview

This project aims to implement an embedded web-browser into Autodesk Inventor 2019 [1]. The purpose being to allow a user to browse the McMaster-Carr online catalog of products, in order to eventually include a product within their Inventor files [2]. Nearly every product in the catalog has a 3D model available for download, along with properties pertaining to the part. The McMaster Add-In for Autodesk Inventor (MAFI) will reduce time needed to implement these 3D models into an Inventor project.

2 Context

This docuemnt should give a user or other developer ample knowledge of the current state of MAFI, including software packages implemented, code repositories, and languages used, along with goals for the future, challenges faced, etc.

2.1 Document Conventions and Definitions

The following describes all naming conventions used throughout the document:

- MAFI: The McMaster-Carr Add-In for (Autodesk) Inventor software.
- Inventor: The computer application "Autodesk Inventor 2019" developed by Autodesk, Inc. [1].
- McMaster: The online catalog available at www.mcmaster.com [2].
- API: Application programming interface.
- 3D model: A three dimensional representation of a physical object, as shown within a computer program such as Inventor.
- Add-in: A software that relies on a parent application in order to execute, typically embedded within the parent application, i.e. non-standalone.
- IProperties: A set of attributes embedded within each Inventor file such as part number, description and physical material.
- Button: A user interface element that executes a process of an application from a mouse click.
- .NET: A software framework developed by Microsoft that includes a large class library and provides language interoperability across several programming languages [3].
- CefSharp: A fast, fully embedded web browser for .NET applications [4].

3 Scope

MAFI is an application add-in built for Inventor as a method for more efficiently connecting to and retrieving models from, the McMaster database of 3D models that often are used by mechanical designers. It will allow for a, direct, in-program access to the online catalog. Along with, "fast" conversion of the database file-type (.stp) into something usable by an Inventor user (.ipt). This is a software that should speed up design projects for engineers who use 3D models available from McMaster, in their Inventor projects.

4 Overview of Requirements

These requirements are a summary of the user stories gathered from the developer and potential users of the add-in. See Appendix A for a collection of user stories.

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. Replace "add to cart" button in HTML with "add to assembly" or "open as part" buttons.
6. Take product information from online catalog and automatically import into Inventor part file.
7. Implement a bill of materials export to send all McMaster items onto clipboard with appropriate QTY per assembly.

5 Languages and Packages

- C#
- XAML
- HTML
- Autodesk Inventor Interop
- CEFSharp v57.0 [4]
- .NET 4.7.2 [3]

6 User Interface

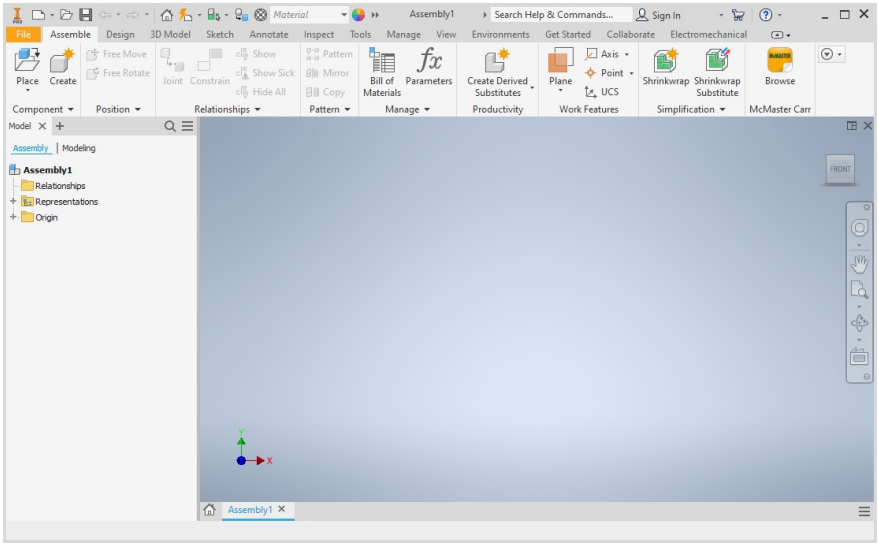

Figure 1: The button(far right) addition to the ribbon interface within Autodesk Inventor.

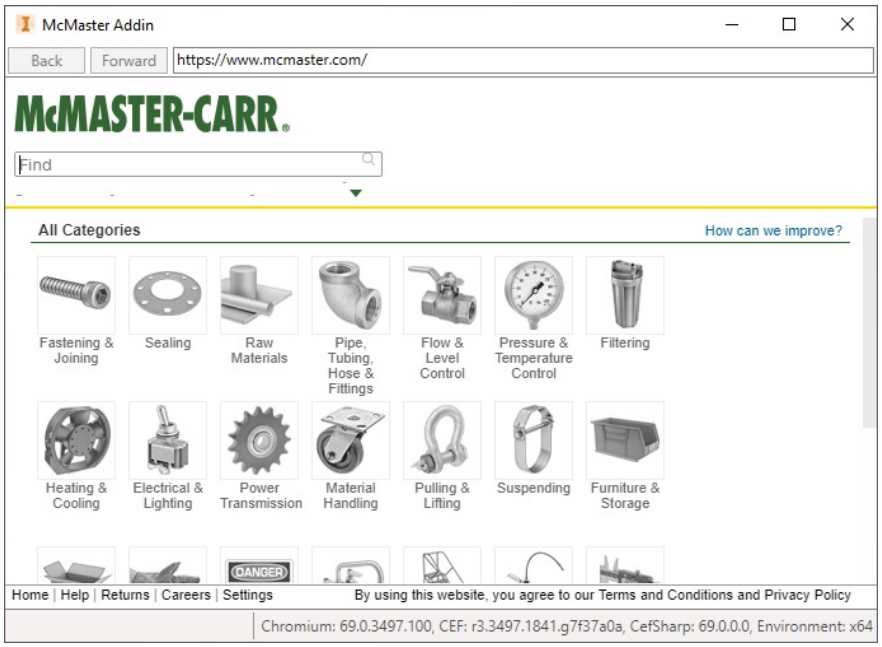

Figure 2: The browser instance following the button execute event.

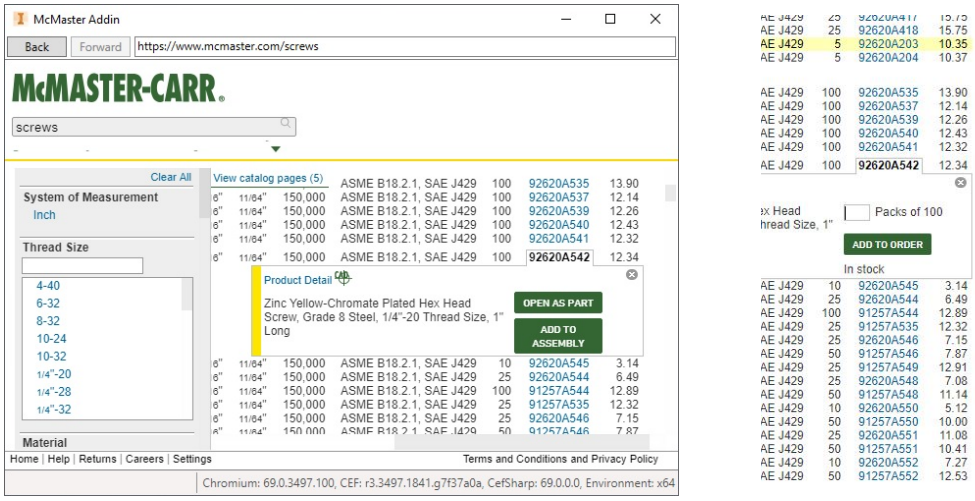

Figure 3: The alteration of the McMaster webpage for new model saving interactions(new on left, old on right)

7 Classes

7.1 StandardAddInServer

This class is responsible for all Add-In initialization, such as hooking into the primary application and calling for a new instance of the McMasterButton to be added into the ribbon interface within Inventor. Currently it contains the method GetSource() as well which stores webpage html to memory in order to parse for a file location, of which can be returned to an instance of the McMasterImporter class for conversion from a .stp file to .ipt, something Inventor can work with. In order to obtain the html from a dynamically created javascript enabled webpage, an instance of HeadlessWebBrowser is used to render the webpage silently.

7.2 HeadlessWebBrowser

This class is a background process version of the CEFSharp utility. It allows for a silent operation of visiting webpages. This enables the software to obtain the html of the "part" page, which contains the url of the 3D model location.

7.3 McMasterButton

This class is a template for the button that StandardAddInServer adds to the Inventor interface. Upon clicking the button, an embedded web-browser will be open, MainWindow, to allow for a user to browse the McMaster catalog for a part they wish to obtain the 3D model of.

7.4 MainWindow

This class is the primary interface a user will see while using MAFI. The McMasterButton will create an instance of this web-browser form. Javascript is injected into the web-browser that dynamically adds "Open as Part" and "Add to Assembly" buttons within the webpage. Clicking these buttons will execute methods to begin the importing process, which is handled by the McMasterImporter, owned by the StandardAddInServer.

7.5 McMasterImporter

This class locates the Inventor translator feature that allows for .stp to .ipt conversion. Then when called will silently convert a file that was obtained from the McMaster catalog, from MainWindow.

8 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, possible copyright or anything legal.
  - Possibly develop for use with Solidworks, a similar program by Dassault Systèmes.

- Current Issues:
  - "Add to Assembly"/"Open as Part" import time is too long, possibly look into pre-conversion of possible part download, such as when a user hover's the button, begin conversion, but if user does not click, delete from memory.
  - Web browser seems to lag on initial startup.

9 References

[1] Autodesk, Inc., "Inventor | Mechanical Design And 3D CAD Software | Autodesk," https://www.autodesk.com/products/inventor/overview.

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

[3] Microsoft Corporation, ".NET Documentation | Microsoft Docs," https://docs.microsoft.com/en-us/dotnet/.

[4] A. Maitland, "CefSharp," https://cefsharp.github.io, v57.0.

10 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 want to be able to choose whether to add a part to my open assembly or as an individual part.
Acceptance Criteria: Two individual buttons for "Add to Assembly" "Open as Part".

107 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.

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

109 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.

110 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.

11 Appendix B: Code

StandardAddInServer.cs

```
using System;
using System.IO;
using reflect = System.Reflection;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using System.Drawing;
using Inv = Inventor;//two using statements to allow for
   disambiguation
//between Inventor and System.IO namespace at times
using Inventor;
using CefSharp.Wpf;
using CefSharp;

#pragma warning disable IDE1006 //naming rules
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
  {
    #region ObjectInitialization
    private static readonly GuidAttribute m_ClientID =
      (GuidAttribute)System.Attribute.GetCustomAttribute(
        typeof(StandardAddInServer), typeof(GuidAttribute));
    public static readonly string m_ClientIDstr =
      "{" + m_ClientID.Value + "}";

    private bool wasOff = false;

    // Inventor application object.
    public Inv.Application m_invApp;
    private HeadlessWebBrowser _headlessBrowser;

    public static readonly string urlBase =
       "https://www.mcmaster.com/";
    public System.Collections.Generic.List<string> fileList =
       new System.Collections.Generic.List<string>();


    //single button for McMaster Catalog
```

```
    private McMasterButton m_Button;
    private McMasterImporter m_Importer;
    //user interface event
    private UserInterfaceEvents m_UIEvents;

    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;
        m_Importer = new McMasterImporter(this);

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

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

        m_Button = new McMasterButton(this);

        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();
            }
          }

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

      public void Deactivate()
      {
        //Need to call on main thread
        if (wasOff){
          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;
      }
    }

    #endregion

    #region Event Handlers

    private void CreateOrUpdateRibbonUserInterface()
    {
      m_Button.AddToUI();
    }

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

    #endregion

    public void GetSource(string url, bool isAssembly)
    {
      //Load Offscreen browser to partNumber webpage, to extract
         file locations
      _headlessBrowser.OpenUrl(url);
      System.Threading.Thread.Sleep(1000);
      string source = "";
      var tS = _headlessBrowser.Page.EvaluateScriptAsync(
        @"document.getElementsByTagName('html')[0].innerHTML");
      tS.Wait();
      JavascriptResponse response = tS.Result;
      var result = response.Success ? (response.Result ??
         "null") : response.Message;
      source = (string)result;
      //Reverse version of REGEX match pattern, to get shortest
         match due to non-greddy algorithm difficulty
      string exp = @">il/" + "<PETS
         D-3>\\\"PETS.+?\\\"=noitpo-dac-mcm-atad
         \\\"dac--il\\\"=ssalc il<";
      string[] matchEnds = new string[] { ">il/<PETS D-3>\"",
         "/\"=noitpo-dac-mcm-atad \"dac--il\"=ssalc il<" };
      source = ReverseString(source);
```

```
    System.Text.RegularExpressions.Regex rx = new
       System.Text.RegularExpressions.Regex(exp);
    System.Text.RegularExpressions.MatchCollection mS =
       rx.Matches(source);
    string fileName = "";
    foreach (System.Text.RegularExpressions.Match x in mS)
    {
      fileName = ReverseString(x.Groups[0].Value
        .Substring(matchEnds[0].Length))
        .Substring(matchEnds[1].Length);
    }
    if (fileName.Length > 0)
    {
      if (Directory.Exists(System.IO.Path.GetTempPath()))
      {
        using (var client = new System.Net.WebClient())
        {
          string filePath = System.IO.Path.Combine(
            System.IO.Path.GetTempPath(),
              url.Substring(urlBase.Length) + ".STEP");
          System.Diagnostics.Debug.WriteLine(filePath);
          System.Net.ServicePointManager.SecurityProtocol =
            System.Net.SecurityProtocolType.Tls |
            System.Net.SecurityProtocolType.Tls11 |
            System.Net.SecurityProtocolType.Tls12;
          client.DownloadFile(new System.Uri(
            System.IO.Path.Combine(urlBase, fileName)),
              filePath);
          fileList.Add(filePath);
          m_Importer.Import(filePath,
             url.Substring(urlBase.Length),isAssembly);
        }
      }
    }
  }
  /// <summary>
  /// A method to reverse a string type by
  /// converting to charArray and using char[].Reverse method
  /// </summary>
  /// <param name="s">The string that you want to reverse the
     order of</param>
  /// <returns>A string with reversed character order</returns>
  public static string ReverseString(string s)
  {
    char[] arr = s.ToCharArray();
    Array.Reverse(arr);
    return new string(arr);
  }
```

```
    private void InitializeCEF()
    {
      //Keep CEF on until INVENTOR exits, not just the WPF form.
      if (!Cef.IsInitialized) {
        CefSharpSettings.ShutdownOnExit = false;

        var settings = new CefSettings { RemoteDebuggingPort =
            8088 };
     //   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);
        wasOff = true;
      }
      CefSharpSettings.LegacyJavascriptBindingEnabled = true;
      _headlessBrowser = new HeadlessWebBrowser();
    }
    public void DeleteTempFiles()
    {
      try
      {
        foreach (string s in fileList)
        {
          if (System.IO.File.Exists(s))
          {
            System.IO.File.Delete(s);
          }
        }
      }
      catch { }
    }
  }

  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

```
using System.Threading;
using System;
using CefSharp;
using CefSharp.OffScreen;

namespace McMasterAddin
{
  public class HeadlessWebBrowser
  {

    /// <summary>
    /// The browser page
    /// </summary>
    public ChromiumWebBrowser Page { get; private set; }
    /// <summary>
    /// The request context
    /// </summary>
    public RequestContext RequestContext { get; private set; }

    // chromium does not manage timeouts, so we'll implement one
    private ManualResetEvent manualResetEvent = new
       ManualResetEvent(false);

    public HeadlessWebBrowser()
    {
      RequestContext = new RequestContext();
      Page = new ChromiumWebBrowser("", null, RequestContext);
      PageInitialize();
    }

    /// <summary>
    /// Open the given url
    /// </summary>
    /// <param name="url">the url</param>
    /// <returns></returns>
    public void OpenUrl(string url)
    {
      try
      {
        Page.LoadingStateChanged += PageLoadingStateChanged;
        if (Page.IsBrowserInitialized)
        {
          Page.Load(url);

          //create a 60 sec timeout
          bool isSignalled =
             manualResetEvent.WaitOne(TimeSpan.FromSeconds(60));
          manualResetEvent.Reset();
```

```
            //As the request may actually get an answer, we'll
               force stop when the timeout is passed
            if (!isSignalled)
            {
              Page.Stop();
            }
          }
        }
        catch (ObjectDisposedException)
        {
          //happens on the manualResetEvent.Reset(); when a
             cancelation token has disposed the context
        }
        Page.LoadingStateChanged -= PageLoadingStateChanged;
      }

      /// <summary>
      /// Manage the IsLoading parameter
      /// </summary>
      /// <param name="sender"></param>
      /// <param name="e"></param>
      private void PageLoadingStateChanged(object sender,
         LoadingStateChangedEventArgs e)
      {
        // Check to see if loading is complete - this event is
           called twice, one when loading starts
        // second time when it's finished
        if (!e.IsLoading)
        {
          manualResetEvent.Set();
        }
      }

      /// <summary>
      /// Wait until page initialization
      /// </summary>
      private void PageInitialize()
      {
        SpinWait.SpinUntil(() => Page.IsBrowserInitialized);
      }
    }
}
```

MainWindow.xaml.cs

```
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Inventor;
using System.Drawing;
using System;

namespace McMasterAddin
{
  class McMasterButton
  {
    private StandardAddInServer _stAddIn;

    public ButtonDefinition m_buttonDefinition;

    private ButtonDefinitionSink_OnExecuteEventHandler
      m_button_Definition_OnExecute_Delegate;

    public McMasterButton(StandardAddInServer s)
    {
      _stAddIn = s;

      Stream myStream = System.Reflection.Assembly.
        GetExecutingAssembly().GetManifestResourceStream(
        "McMasterAddin.Resources.mcmaster.ico");

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

      //Button definition
      m_buttonDefinition = _stAddIn.m_invApp.CommandManager.
        ControlDefinitions.AddButtonDefinition("Browse",
        "BrowseButton",
        CommandTypesEnum.kQueryOnlyCmdType,
           StandardAddInServer.m_ClientIDstr,
        "Browse McMaster-Carr Inventory", "Use this to find " +
        "hardware and other products available on McMaster.com",
        largeImage, largeImage,
           ButtonDisplayEnum.kAlwaysDisplayText);

      m_button_Definition_OnExecute_Delegate = new
        ButtonDefinitionSink_OnExecuteEventHandler(
        m_button_OnExecute);
      m_buttonDefinition.OnExecute +=
        m_button_Definition_OnExecute_Delegate;
      m_buttonDefinition.Enabled = true;
```

```
    }

    public void AddToUI()
    {
      UserInterfaceManager UIManager =
        _stAddIn.m_invApp.UserInterfaceManager;
      Ribbon assemblyRibbon = UIManager.Ribbons["Assembly"];
      RibbonTab assembleTab =
         assemblyRibbon.RibbonTabs["id_TabAssemble"];
      RibbonPanel mcMasterPanel =
        assembleTab.RibbonPanels.Add("McMaster Carr",
          "McMasterPanel", StandardAddInServer.m_ClientIDstr);

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

    /// <summary>
    /// This is the mcMaster-addin button execution
    /// </summary>
    /// <param name="Context"></param>
    private void m_button_OnExecute(NameValueMap Context)
    {
      //Refer to MainWindow.xaml for code of the browser
         extension
      var wpfWindow = new McMasterAddin.MainWindow(_stAddIn);
      //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)
      {
        Owner = new IntPtr(_stAddIn.m_invApp.MainFrameHWND)
      };

      wpfWindow.ShowDialog();
    }
  }
}
```

MainWindow.xaml.cs

```
using System.Threading;
using System.Windows;
using CefSharp;
using System.Text.RegularExpressions;

namespace McMasterAddin
{
  /// <summary>
  /// Interaction logic for MainWindow.xaml
  /// </summary>
  public partial class MainWindow : Window
  {
    private StandardAddInServer _stAddIn;
    // chromium does not manage timeouts, so we'll implement one
    private ManualResetEvent manualResetEvent = new
       ManualResetEvent(false);

    private string currentURL = "";
    private string currentSource = "";
    private string s = "";

    public MainWindow(StandardAddInServer a)
    {
      InitializeComponent();
      Browser.RegisterAsyncJsObject("mainWindowOBJ",
        new JavaScriptInteractionObj(a,this));
      _stAddIn = a;
      currentURL = (string)Browser.Address.Clone();

      using (System.IO.Stream myStream = System.Reflection
        .Assembly.GetExecutingAssembly()
         .GetManifestResourceStream(
          "McMasterAddin.Resources.addButtonScripts.js"))
      {
        using (System.IO.StreamReader sRdr =
          new System.IO.StreamReader(myStream))
        {
          s = sRdr.ReadToEnd();
        }
      }
    }

    /// <summary>
    /// Manage the IsLoading parameter
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void PageLoadingStateChanged(object sender,
```

```
    LoadingStateChangedEventArgs e)
  {
    if (!e.IsLoading)
    {
      Dispatcher.Invoke(new System.Action(() =>
      {
        if (currentURL != Browser.Address)
        {
          Browser.ExecuteScriptAsync(s);
          currentURL = (string)Browser.Address.Clone();
        }
      }));
    }
  }

  private void Window_Loaded(object sender, RoutedEventArgs e)
  {
    Browser.LoadingStateChanged += PageLoadingStateChanged;
  }

  private void Window_Closing(object sender,
    System.ComponentModel.CancelEventArgs e)
  {
    _stAddIn.DeleteTempFiles();
  }
}

public class JavaScriptInteractionObj
{
  private StandardAddInServer _stAddIn;
  private MainWindow _mW;

  public JavaScriptInteractionObj(StandardAddInServer s,
     MainWindow m)
  {
    _stAddIn = s;
    _mW = m;
  }

  public void AddToAssembly(string pNumber)
  {
    MessageBox.Show(pNumber);
    _stAddIn.GetSource(StandardAddInServer.urlBase +
      pNumber.Replace("partNumber",""),true);
  }

  public void OpenAsPart(string pNumber)
  {
    MessageBox.Show(pNumber);
```

```
        _stAddIn.GetSource(StandardAddInServer.urlBase +
          pNumber.Replace("partNumber", ""),false);
        _mW.Dispatcher.BeginInvoke(new System.Action(() =>
        {
          _mW.Close();
        }));
      }
    }

}
```

MainWindow.xaml.cs

```
using Inventor;

namespace McMasterAddin
{
  public class McMasterImporter
  {
    private StandardAddInServer _stAddIn;
    private string translatorID = "";
    private readonly static string tempDirectory =
      System.IO.Path.GetTempPath();

    public McMasterImporter(StandardAddInServer s)
    {
      _stAddIn = s;
      GetTranslatorAddInID("Translator: STEP");
    }

    /// <summary>
    /// A method for converting a .STEP to .IPT silently and
       adding to assembly.
    /// </summary>
    /// <param name="strFilePath">Path location of .STEP
       file</param>
    /// <param name="strFileName">Name of final file without
       suffix or file path</param>
    public void Import(string strFilePath, string
       strFileName,bool isAssembly)
    {
      ApplicationAddIns oAddIns =
         _stAddIn.m_invApp.ApplicationAddIns;
      TranslatorAddIn oTransAddIn =
         (TranslatorAddIn)oAddIns.ItemById[translatorID];
      oTransAddIn.Activate();

      TransientObjects transObj =
         _stAddIn.m_invApp.TransientObjects;

      DataMedium file = transObj.CreateDataMedium();
      file.FileName = strFilePath;

      TranslationContext context =
         transObj.CreateTranslationContext();
      context.Type = IOMechanismEnum.kFileBrowseIOMechanism;

      NameValueMap options = transObj.CreateNameValueMap();

      bool oHasOpt = oTransAddIn.HasOpenOptions[file, context,
         options];
```

```
      oTransAddIn.Open(file, context, options, out object oDoc);

      Document doc = (Document)oDoc;
      _stAddIn.m_invApp.SilentOperation = true;
      doc.SaveAs(tempDirectory + strFileName + ".ipt", false);
      _stAddIn.m_invApp.SilentOperation = false;

      //Add the converted .ipt file into my active assembly
      if (isAssembly)
      {
        doc.Close();
        //Create an operation matrix that contains information
        //about starting position of my part.
        Matrix oMatrix = _stAddIn.m_invApp
          .TransientGeometry.CreateMatrix();
        oMatrix.SetTranslation(_stAddIn.m_invApp
          .TransientGeometry.CreateVector(), true);
        ((AssemblyDocument)_stAddIn.m_invApp.ActiveDocument)
          .ComponentDefinition.Occurrences.Add(tempDirectory
          + strFileName + ".ipt", oMatrix);
      }
      else
      {
        doc.Views.Add();
      }
    }

    private void GetTranslatorAddInID(string translatorName)
    {
      ApplicationAddIns oAddIns =
        _stAddIn.m_invApp.ApplicationAddIns;
      TranslatorAddIn oTransAddIn;

      foreach (ApplicationAddIn a in oAddIns)
      {
        if (a.AddInType ==
          ApplicationAddInTypeEnum.kTranslationApplicationAddIn)
        {
          oTransAddIn = (TranslatorAddIn)a;
          if (oTransAddIn.DisplayName == translatorName)
          {
            translatorID = oTransAddIn.ClassIdString;
          }
        }
      }
    }
  }
}
```