INTERFACING PRIME 3 WITH WINDOWS

1. OVERVIEW

This project helps us process input entered in a field in a webpage that is a part of the application and upon verifying the input, performs one or more of the following actions:

1) Restart the computer gracefully and start Chrome in Kiosk mode after the restart.
2) Disable/Enable the Windows home button on the tablet (Requires restart to take effect).
3) Setting printer margins to zero.
4) Deleting local printers.

2. STRUCTURE

The project consists of two major parts that help us accomplish the goals outlined in the overview:

1. Chrome Extension (Written in Javascript)
2. Native Host Application (Written in C#)

File Structure: C:\Projects\Prime_3

- Prime3_Chrome_Extension (Contains the Chrome Extension that needs to be unpacked)
- Prime3_Native_App (Contains Code for C# native application)
- Prime3_Original (Original Prime 3 Code)
- README

2.1 Chrome Extension

2.1.1 Description

The Chrome extension allows us to obtain and verify information from the webpage and using the information retrieved from the fields in the webpage and interacts with the Native Host application (described in detail later).

2.1.2 Setting up the Chrome Extension

Follow the steps detailed below to setup the Chrome Extension

1.Open Chrome and navigate to chrome://extensions.

2.Check the Box that reads “Developer mode”

3.Click on load unpacked extension.

4.Browse to the folder containing the chrome extension
C:\<YOURPATH>\Prime_3\Prime3_Chrome_Extension, and click 'ok'.

5. Make sure the checkbox that reads “Enabled”, near the extension that was just loaded is checked.

2.1.3 How the Chrome extension works

2.1.3.1 The Content.js content script

The content.js file is the part of the Chrome extension which can interact and obtain information from the current webpage. In our project, the code detailed in the content.js file allows us to read and verify the access code that is entered in the webpage. The content.js page also interacts with another component of our chrome extension, Background.js which is described [here] below in section 2.1.3.2.

For more information on the content.js file, what actions can be performed using it, please refer to the Chrome extension documentation here: https://developer.chrome.com/extensions/content_script

2.1.3.2 The Background.js script

This is a background script in our chrome extension that can inform us of browser actions, the state of the browser and can interact with the chrome native host application (described below). This file receives messages from the content script and sends them over to the chrome native host application.

For more information on the background scripts, what actions can be performed using it, please refer to the Chrome extension documentation here:
https://www.developer.chrome.com/extensions/background_pages

2.1.3.3 The manifest.json manifest file

This is the file that describes the Chrome extension to the browser. It contains details about the content and the background scripts and provides permissions to the extension regarding the pages that the content script can match (i.e. perform actions on through URL matching) and whether the extension can send messages to chrome's native host applications.

For more information on the manifest file and how to structure it and use it, refer to the documentation provided here: https://developer.chrome.com/extensions/manifest

2.2 Native Host Application

2.2.1 Description

The native host application allows us to interface with the Chrome extension and Windows. Since we cannot communicate and perform actions directly from Chrome to Windows (as it would be a huge security loophole). Native host applications are a way to achieve this type of communication. Messages can be sent to and from the extension and the native host application through "native messaging". The messages are sent and received in the form of standard input and output. For more information on Native host applications and native messaging, please refer to the following documentation:
https://developer.chrome.com/extensions/nativeMessaging

2.2.2 Setting up the Native Host Application

The following steps need to be followed to install the native host application on the tablet.

(This setup can be replaced by an installation script that can run with on a single click).

1. Open command prompt using the search function on windows.

2. Add the following registry entries using REG ADD command:

REG ADD "HKCU\Software\Google\Chrome\NativeMessagingHosts\com.prime3.interface" /ve /t REG_SZ /d " C:\<YOURPATH>\Prime_3\Prime3_Native_App\manifest.json" /f

Copy and paste the line above and insert your path where it says <YOURPATH>

Go to the manifest file (C:\<YOURPATH>\Prime_3\Prime3_Native_App\manifest.json) of the native host application and change the chrome extension id to the current unpacked extension. You can look it up under Chrome >> Extensions >> Prime 3 Extension ID

This manifest file contains a hardcoded path to the exe file (NativeHostApplication.exe). Change this to the current path of this exe **Where the path is on your computer** (In Prime3 Tablet: "path": "C:\\Projects\\Prime_3\\Prime3_Native_App\\NativeHostApplication\\NativeHostApplication\\bin\\Debug\\NativeHostApplication.exe")

2.2.3 How the Native Host Application works

The native host application is written in C#. The host application is started automatically by Chrome, when a request to connect to it is received from the Chrome extension. Upon receiving the request to connect, the Native Host can communicate through standard input and standard output.

We have the following functions

1. readFromExtension() : This function reads the input from standard input and uses the StreamReader object to obtain the data in chunks.

2. sendToStandardOuput(): This function is used to process and send a response back to the Chrome extension through standard output.

3. extractMessage(): This function takes the JSON value obtained from the readFromExtension() function and extracts the string message that was sent from the extension.

The main program, based on the message received from the extension performs the desired action. There actions that are currently supported are

1.Performing a restart and starting with the Chrome application running in Kiosk mode. (reboot_and_startChrome.bat). Note: In Prime_3\Prime3_Native_App\BatchCommands\startChrome.bat, Change file:///C:/Projects/Prime_3/Prime3_Original/Prime3_Sample_Webpage.html to Path to Prime III homepage.

2. Enabling or disabling the Windows key. (disableWindowsKey.bat, enableWindowsKey.bat) This requires a restart to take effect.

3. Deleting the local printers (Implemented, requires testing). (deleteLocalPrinters.bat) To test: Go to control panel and add dummy printer (http://support.esri.com/technical-article/000008077)

4.To set printer margins to zero, we manipulate the CSS @page element under the media tag which controls the size and the orientation of the page to be printed and set it to zero.

Unfortunately, Chrome's print preview module can and does on occasion override these changes and there doesn't exist an option currently through the developer tools provided by Chrome to change this. Currently, the only tool available is the 'chrome.printProvider' module, which can send jobs to the printer, get information on the printers on the network etc. but unfortunately, does not provide an option to modify the dimensions of the margin.