Converting an ID9 Custom UI in APEX

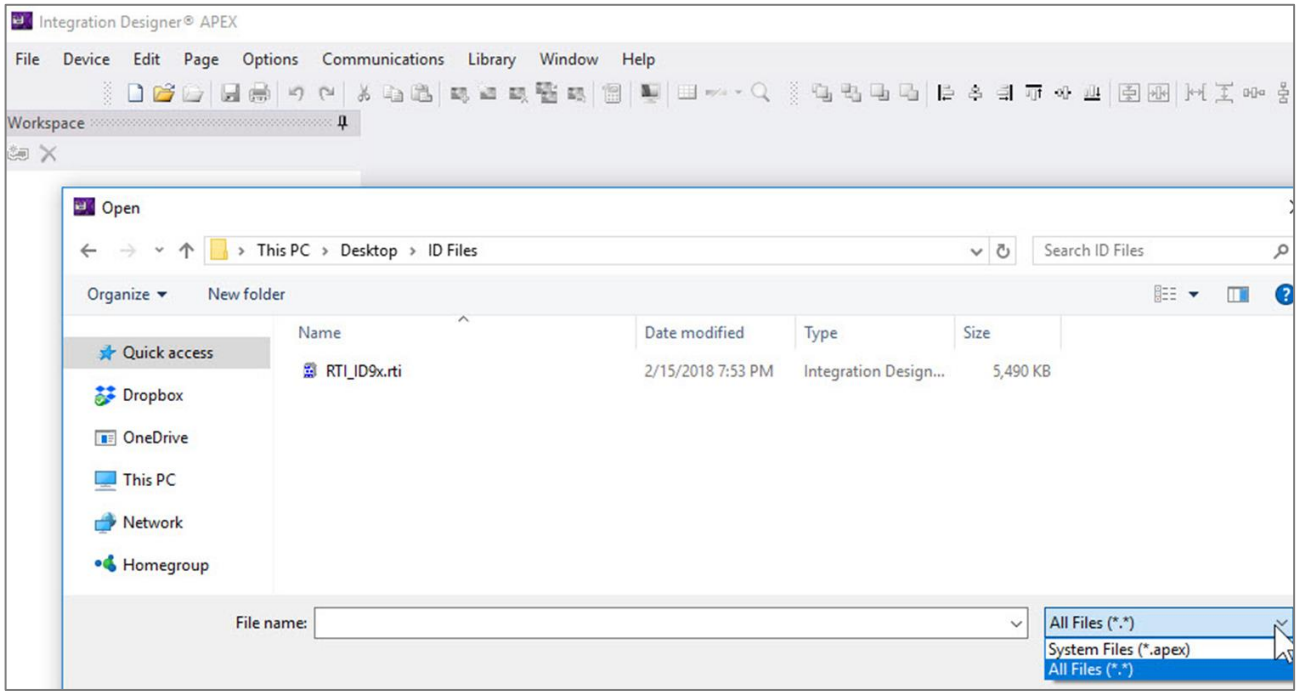
A. Overview

Many dealers have devoted time to creating and refining graphical interfaces over the years using RTI's Integration Designer software. Converting a custom interface project originally programmed in Integration Designer 9.x can be converted to an Apex file and used for future projects, taking full advantage of the tags utilized in Apex.

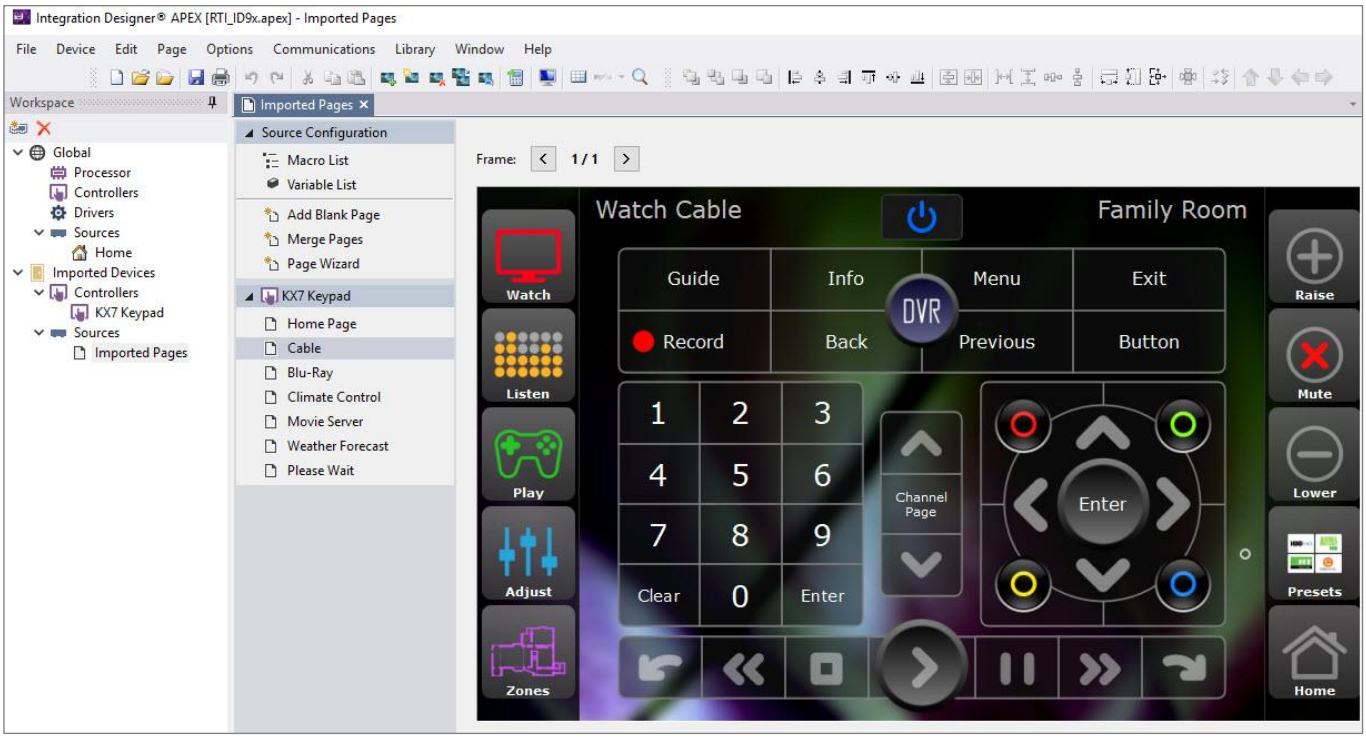
It is important to know that Apex follows a room structure and tagging methodology that is not a standard in ID9.x. For this reason, you will not retain your control processor, page links or any functionality as it relates to driver, IR or any other command structure. Apex will group all your pages in a single room called "imported devices" and again, will not retain your master control processor in the global area.

B. Converting your ID9 Custom UI in APEX

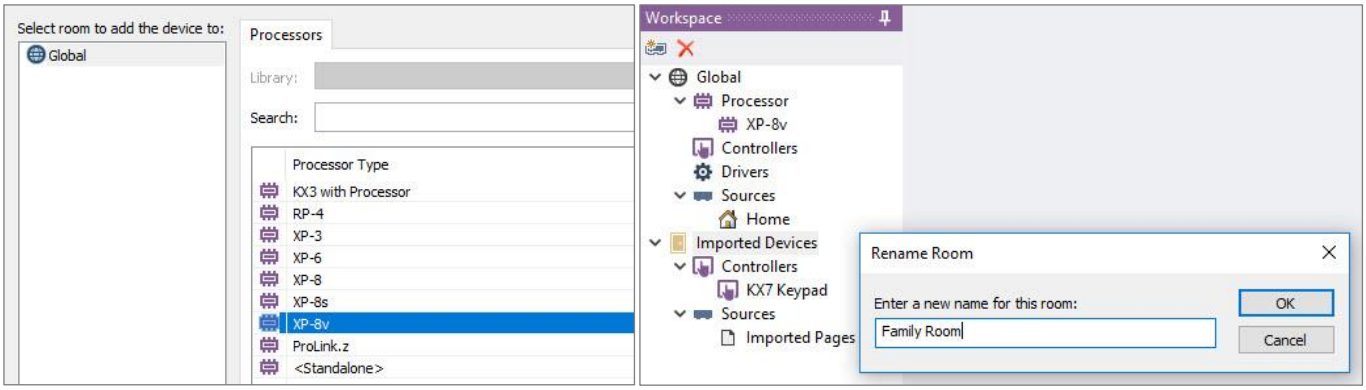
Launch Apex, navigate to the directory where your ID9.x file is located, and select File/Open from the top left toolbar. Next, toggle the file type to All Files (*.*) and click on your ID9.x file to open it in Apex.



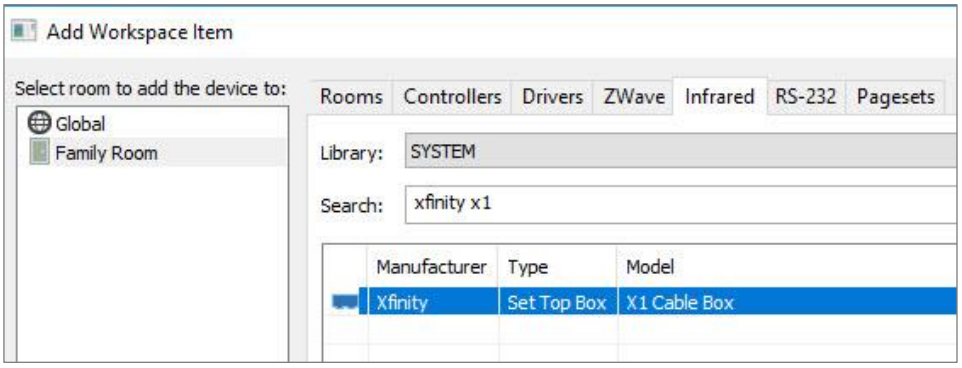
As you can see, your control processor will be missing from the global area. Apex will create a room named “imported devices” with a single page group of all your ID9.x pages. You will lose all your functionality in the template due to the lack of button tagging, but the graphics will port over without any compromise.



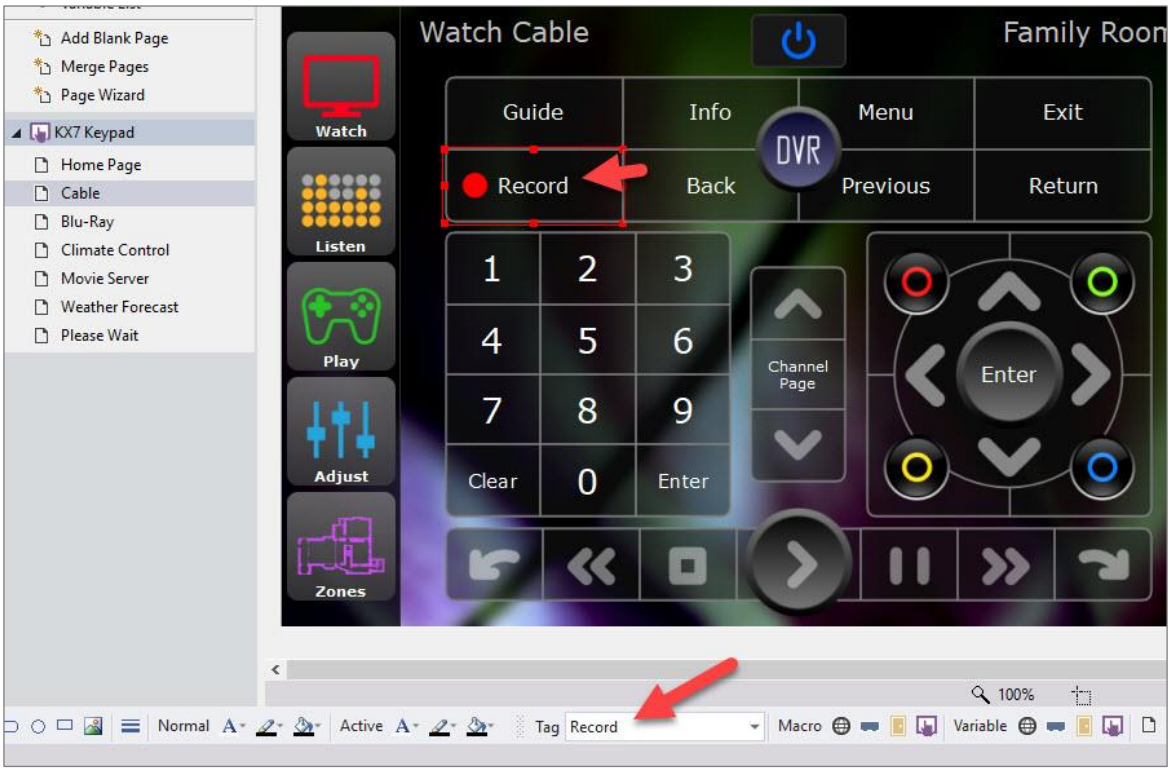
Click the “add workspace” icon in the upper left of the workspace and add a control processor to the global area in your system file. We are adding an XP-8v. Next, change the name of the default “imported devices” to a room name, in this case the family room.



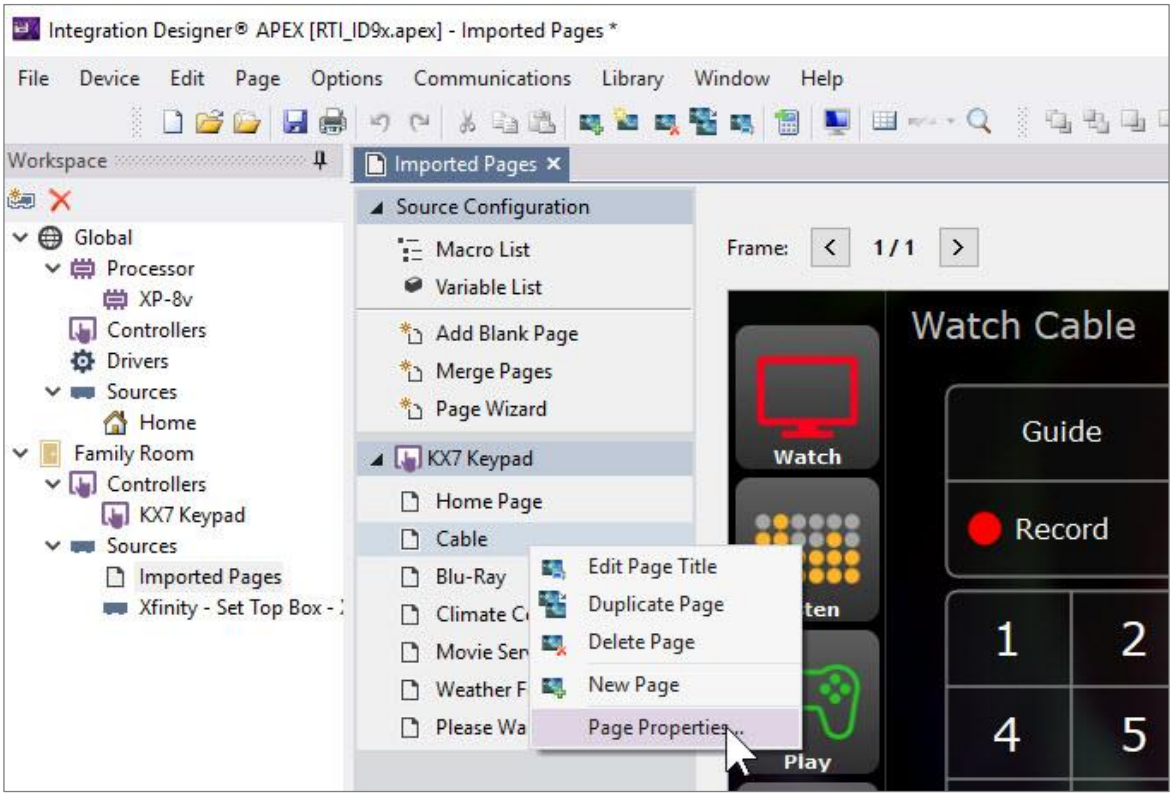
Click the “add workspace” button on in the workspace area and add a device to the room or global area, depending on the functionality of your system file. You may add a device from the driver, Infrared or RS-232 tab. We will add an Xfinity X1 cable box to the Family Room. Note that you will not add pages to your device since you already have custom graphics for the interface you will be using for this device library.



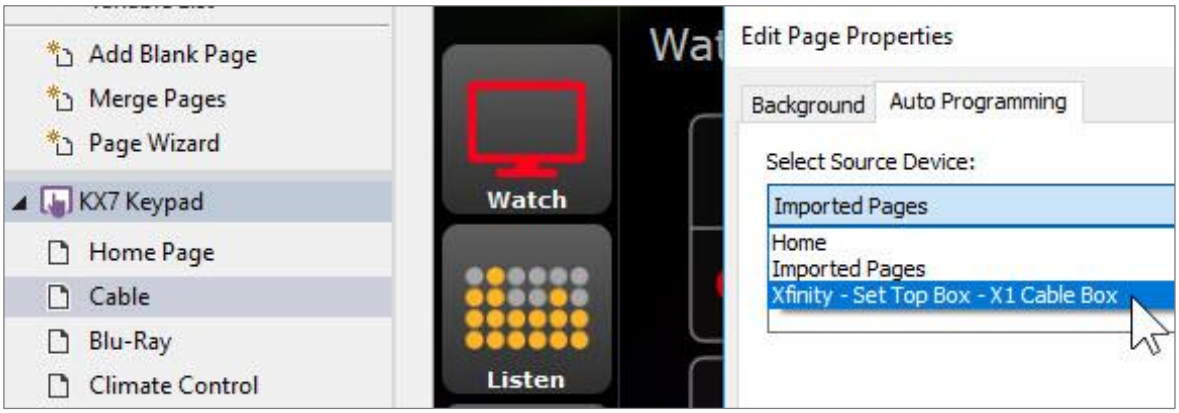
Close the “Add Workspace Item” interface and select the custom graphic page that corresponds to the device library you just added, which is called “Cable” in our test example. Begin tagging each button or element in your interface with tag designations that are typically found in a cable or satellite interface. Simply select the button and name the tag accordingly in the tag field in APEX.



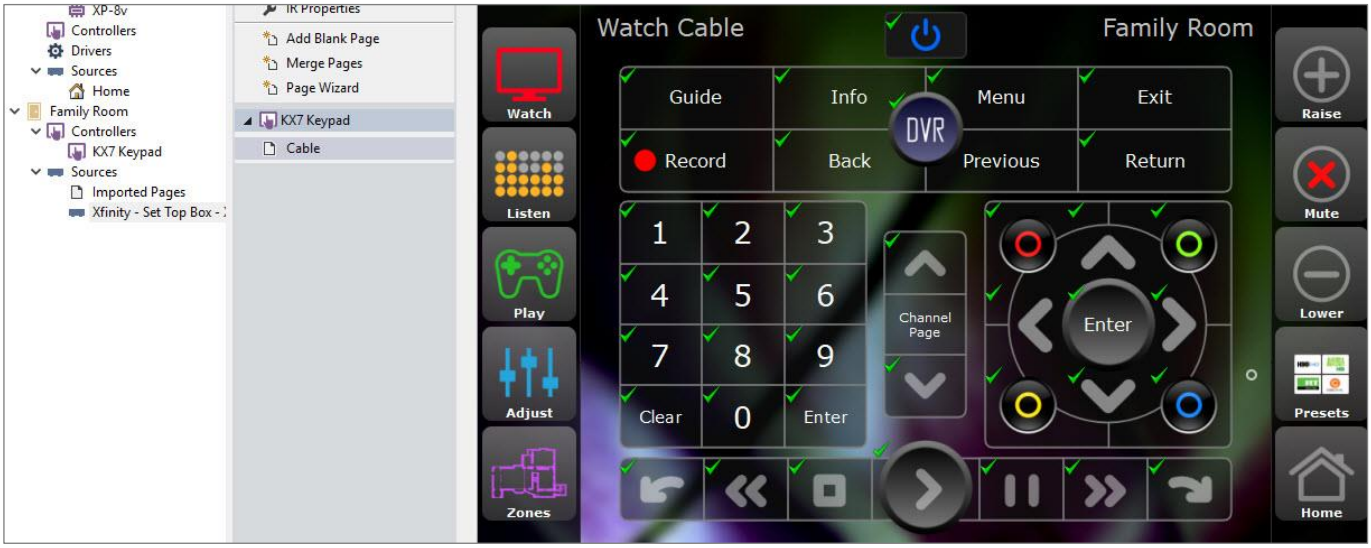
Once the interface is tagged, select the page set or user interface in your imported devices that corresponds to the device you added to the system file. Since the Xfinity X1 is a cable box, we will right-click the Cable custom page and select “Page Properties”



Click the “Auto Programming” tab and select the device library that corresponds to the custom page user interface you are programming.



Your page will now relocate from the “Imported Pages” device as a page set under your actual device, in this case Xfinity Set Top Box. Click on the Cable page under your device and observe how each button is now populated with the command from the library.



As observed, our user interface has been properly tagged and auto-programmed sufficiently. If there is any maintenance work to do or missing tags, you can investigate this and make any necessary changes. Each time you select the Auto Programming tab you can apply another round of auto-programming to any tags you added.

Continue to the next device and repeat the process. Once your custom interface has been tagged appropriately and auto-programmed, you may use it repeatedly even for other devices. If your tags are consistent with Apex tag naming methodology, you will have success. And while this takes some initial work up front, you can now easily auto-program various devices with your custom graphics.

In our example, we used a one-way IR library for auto-programming, but you can easily auto-populate two-way variables if they are properly named and present in the implemented system driver.