Example - JSON AutoComplete Choices URL with a Universal Display File

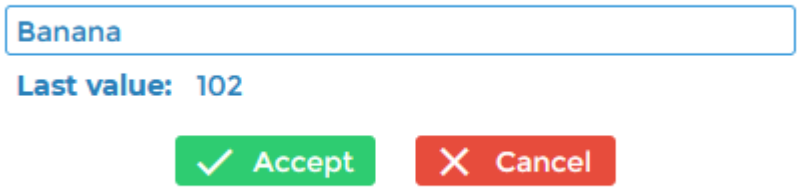
Since publishing the Example - Using Choices URL with a Universal Display File example several years ago, we've received comments that the example didn't use a true JSON file. This example provides a genuine JSON example that builds a REST API that provides full Auto-Complete Functionality on a text box.

This example requires Profound UI version 6 and fix pack 13.0 in order to work properly. If you'd like to try this feature prior to the release of fix pack 13.0, you can contact support@profoundlogic.com to get a patch for fix pack 12.

You can download a .ZIP file containing the example here: JsonChoicesUrl.zip. View the README.txt file in the .ZIP file to learn how to set up the example so you can try it.

Once you have set up the program by following the steps in the README.txt file you can run it from Genie (or from an Atrium Launcher or Rich Display initial program) by setting up your library list properly and running **CALL FRUITSR**

The example will demonstrate doing auto-complete with a textbox by providing a screen that looks like this:



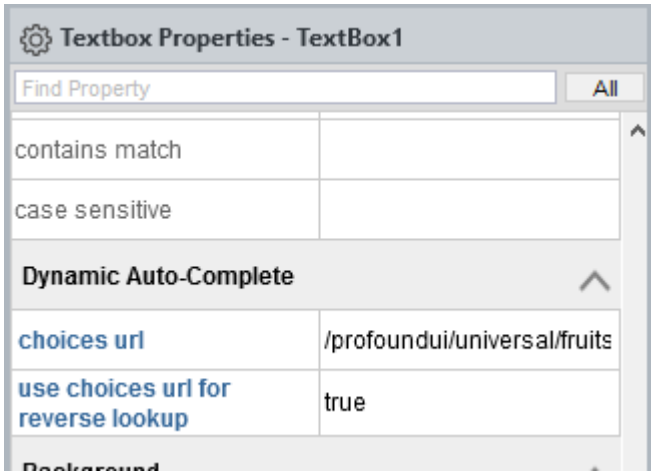
If you erase part of the word 'Banana' and/or start typing characters from a different fruit, it will offer autocomplete choices that show any fruits that match the letters you've typed. Under the covers, it is calculating these fruit names by running the choices url. The remainder of this page will help you understand how the choices url works as a Universal Display file.

Notes About The Display Files

The FRUITSD display file is a normal rich display file. (The D at the end is short for "display.") The FRUITSUD display file is a universal display file (the UD at the end is short for "universal display.") Using D for display file and UD for universal display file is a common naming convention, but it is not necessary for the displays to work properly.

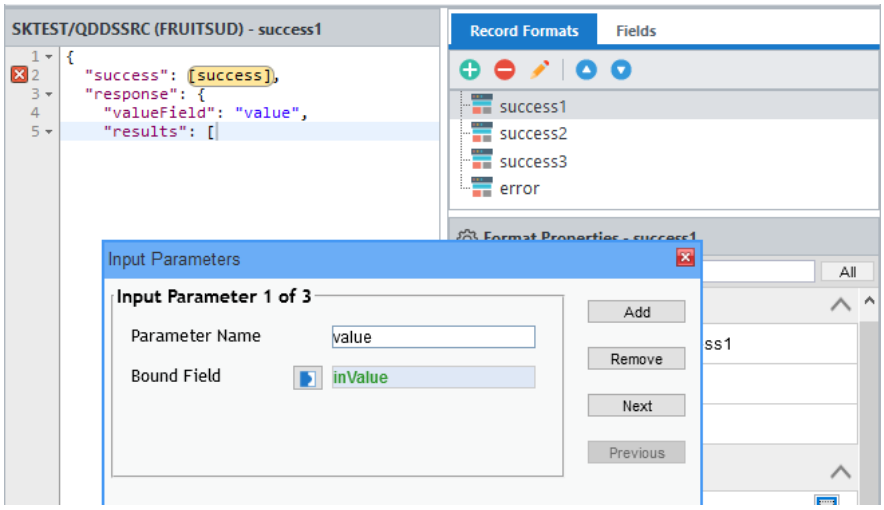
Things you should notice in the FRUITSD display file.

1. Open the FRUITSD display file in the Visual Designer. Look at the "choices url" property of the textbox. The value "/profoundui/universal/fruits" corresponds to the "/fruits" mapping that you set up in the steps of the README.txt file. You added a record to the PUIMAPP file that mapped /fruits to a program named FRUITSUCL which runs the FRUITSUR RPG program. That means any time you access that URL, it will run that RPG program.
2. Notice that the textbox also has the "use choices url for reverse lookup" property set to true. This tells the textbox that it should load the text that corresponds to the 'value' property when the screen first loads. Normally we look up data by searching for the characters you have typed, but when the screen first loads it needs to do the opposite and search for the number that corresponds to the characters, so this is called a "reverse lookup".
3. When a reverse lookup is performed, variables reverse=1 and value=XXX (where XXX is the number bound to the value property of the textbox) will be submitted to the choices url so it knows the text to look for.
4. During run-time whenever you type a character into the text box, it will run the "choices url" again to get a new value to display on the screen. In this case, it is a normal (forward) lookup, and a variable query=XXX (where XXX is the characters typed into the textbox) will be sent to the choices url.



Things you should notice in the FRUITSUD universal display file.

1. Since running the /profoundui/universal/fruits URL will run the FRUITSUCL program, which runs th FRUITSUR rpg program, any time the "choices url" is called, it will in fact run FRUITSUR. FRUITSUR uses the FRUITSUD universal display file to process the input and output to implement a REST API.
2. Open the FRUITSUD universal display in the universal display designer (the URL is http://your-system:port/profoundui/universal) you must use this designer (instead of the normal Visual Designer) when working with Universal Displays.
3. Under the "Format Properties" for the "success1" format, you'll notice that the "input parameters" property has been used. If you click on it, there's a "..." button that can be used to view those properties.
4. The input properties are defined as follows. Notice that while its commonplace to name the input parameters the same as the corresponding RPG variables, its not necessary to do so. In this example, *value* will be renamed to *inValue* in the RPG program, but the others match the name of the RPG variable.
    - **value** is bound to the **inValue** RPG variable.
    - **reverse** is bound to the **reverse** RPG variable.
    - **query** is bound to the **query** RPG variable.
5. Refer back to the discussion of forward and reverse lookups in points 2-4 under the FRUITSD display file. These correspond to the input parameters above. After reading the *success1* record format in the RPG program, the following will be true:
    - When a reverse query is performed, the RPG variable named *reverse* will be **on*
    - When a reverse query is performed, the RPG variable named *inValue* will be set to the value that was bound to the textbox.
    - When a forward query is performed, the RPG variable named *query* will be set to the letters the user has typed into the textbox.
6. The output of the **success1** record format contains the portion of the JSON response that should be written at the top of the JSON document. The "valueField" is special and matches one of the JSON fields specified in the *success2* format. This is how the Rich Display knows which field contains the numeric fruit id vs. which field contains the text representation to show to the user.
7. The output of the **success2** record format provides one row of the choices to be returned. This format can be written in a loop to build the entire list of choices. The separator property has been set to a comma to denote that each time this format is written, it should be separated from the next one by writing a comma.
8. The output of the **success3** record format provides the portion of the JSON response that should be written at the end of the JSON document to finish it.
9. The output of the **error** record format can be written in lieu of the 3 *success* formats to write an error message.



Notes About the RPG Programs

Things you should notice about the FRUITSUR rpg program.

1. This program is the main logic for the REST API described above. It is called both for the reverse lookup and for each forward lookup (on every key press) so should be relatively fast.
2. You can calculate the values in the program any way you wish. In this example, to keep it simple, the options are predefined and hard-coded in an array in the program. You may do it differently, however, such as reading the options from a file or any other method that makes sense to your application.
3. The program must read a format that has its *input parameters* property set up in order to get input. Notice that the first step in the code (after hard-coding the array values) is to read the success1 format. This corresponds to the information explained in items 3-5 under the FRUITSUD universal display, above. After reading *success1*, the *inValue*, *reverse* and *query* variables will be set to whatever was sent by the textbox widget.
4. The RPG code uses the *reverse* variable to determine whether a reverse lookup is being performed. When it is on, it searches the hard-coded array for a value that matches the numeric code for a given fruit. The success1, success2 and success3 formats are written to build a json document with the matching value, or if no value is found, the error format is written to indicate the error.
5. When the *reverse* variable isn't on, it a forward query is performed by searching the array for any fruit name that contains the characters that were typed. The characters are provided in the *query* variable. The program will create a temporary copy of both *query* and the fruit names in all-lowercase so that the search is not case-sensitive. The *success1* format is written at the start, and then *success2* is written for each matching fruit. *success3* is written at the end to finish the JSON document.
6. The *inlr variable is set at the end of the program which forces RPG to close the FRUITSUD universal display file. This is important because if left open, the universal display would continue where it left off on the next call instead of starting fresh with a new document.

Things you should notice in the FRUITSR rpg program.

1. FRUITSR is the interactive RPG program that you run to demonstrate this feature. It is a regular RPG program that uses a Rich Display file. (The universal display is, subsequently, called from that Rich Display.)
2. FRUITSR is simple. It declares a Rich Display, then executes the FRUITS1 display format in a loop until the cancel button is pressed.
3. FRUITSR copies the value that was provided in the textbox to another variable named "lastval" that shows the previous value on the screen. This helps you see the difference between what is shown on the screen by the universal display and what is submitted when the Rich Display returns to FRUITSR.