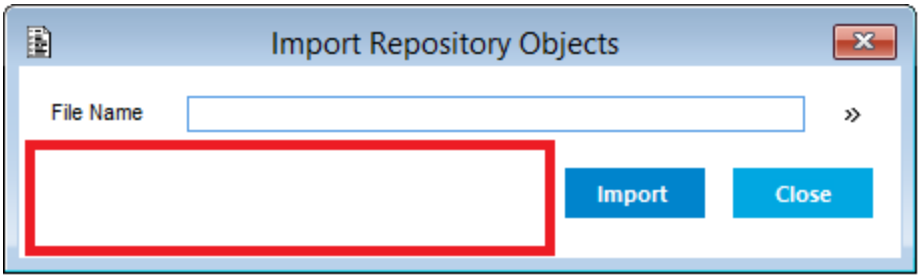
Drop it like it's hot

(Original creator: a4dvo)

What very few people know is that in the Import Repository Objects component there is a small section where you can drag and drop your export files from Windows Explorer. After you drop them, the File name box is updated with the list of files that you dropped.

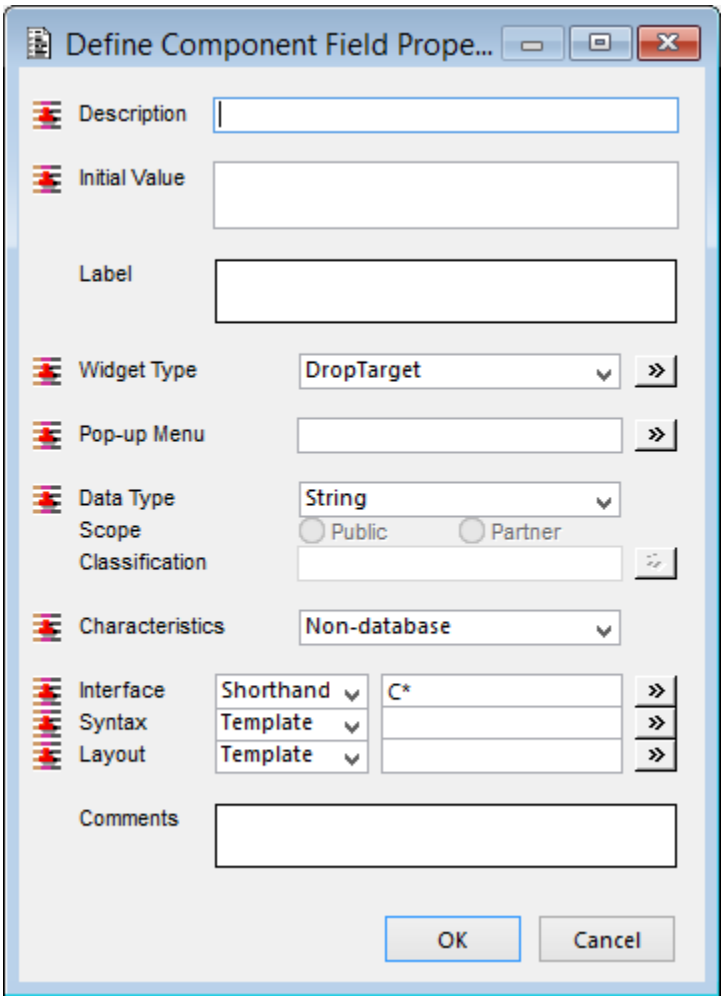


The red section is where the drag-and-drop field is located.

Download for this drag n drop sample

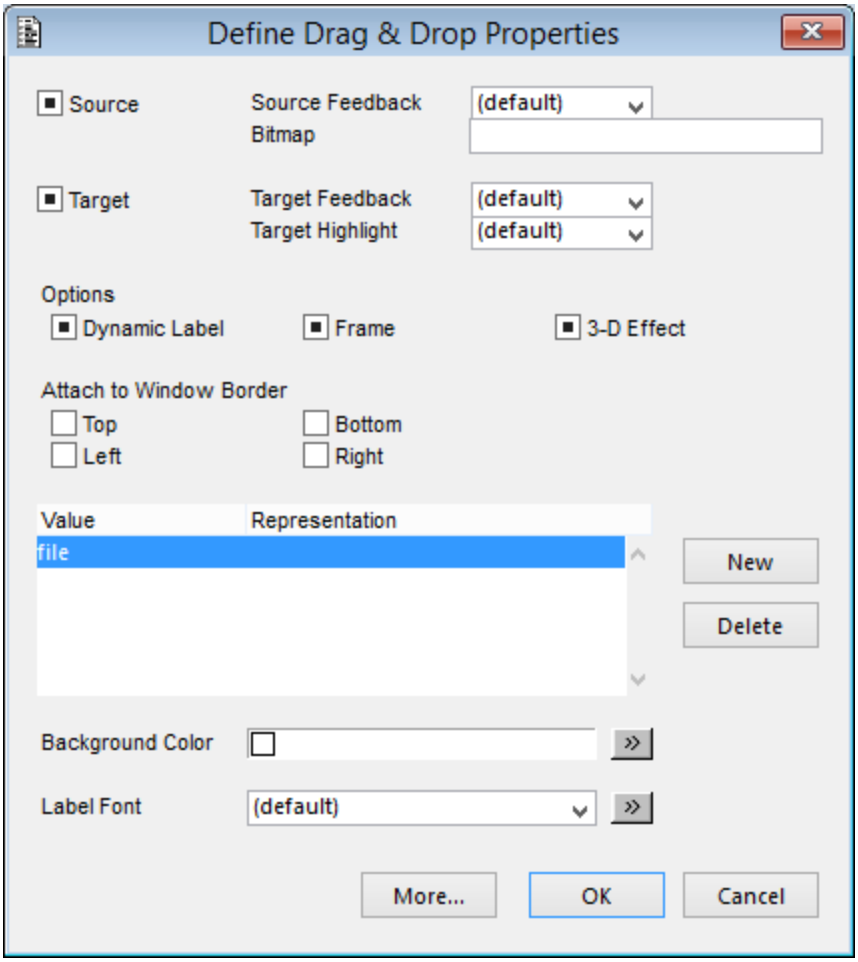
Creating this element in your own component is pretty easy. Just follow these steps:

1. Create a component. In this example I have created a component named DragNDrop
2. Paint an entity. I used a non-database entity named DE in the DM application model.
3. Paint a field. DROPTARGET
4. Now set the properties according to the image below.



We set the widget type to DropTarget indicating that this element will be able to receive elements using a drop event. Make sure you painted the widget large enough so that you can find it very easy on the component. In this setting the widget will have the same gray color as the rest of the component, making it impossible to find if you paint it too small. I painted it across the entire component so it is impossible to miss. I have set the datatype to string

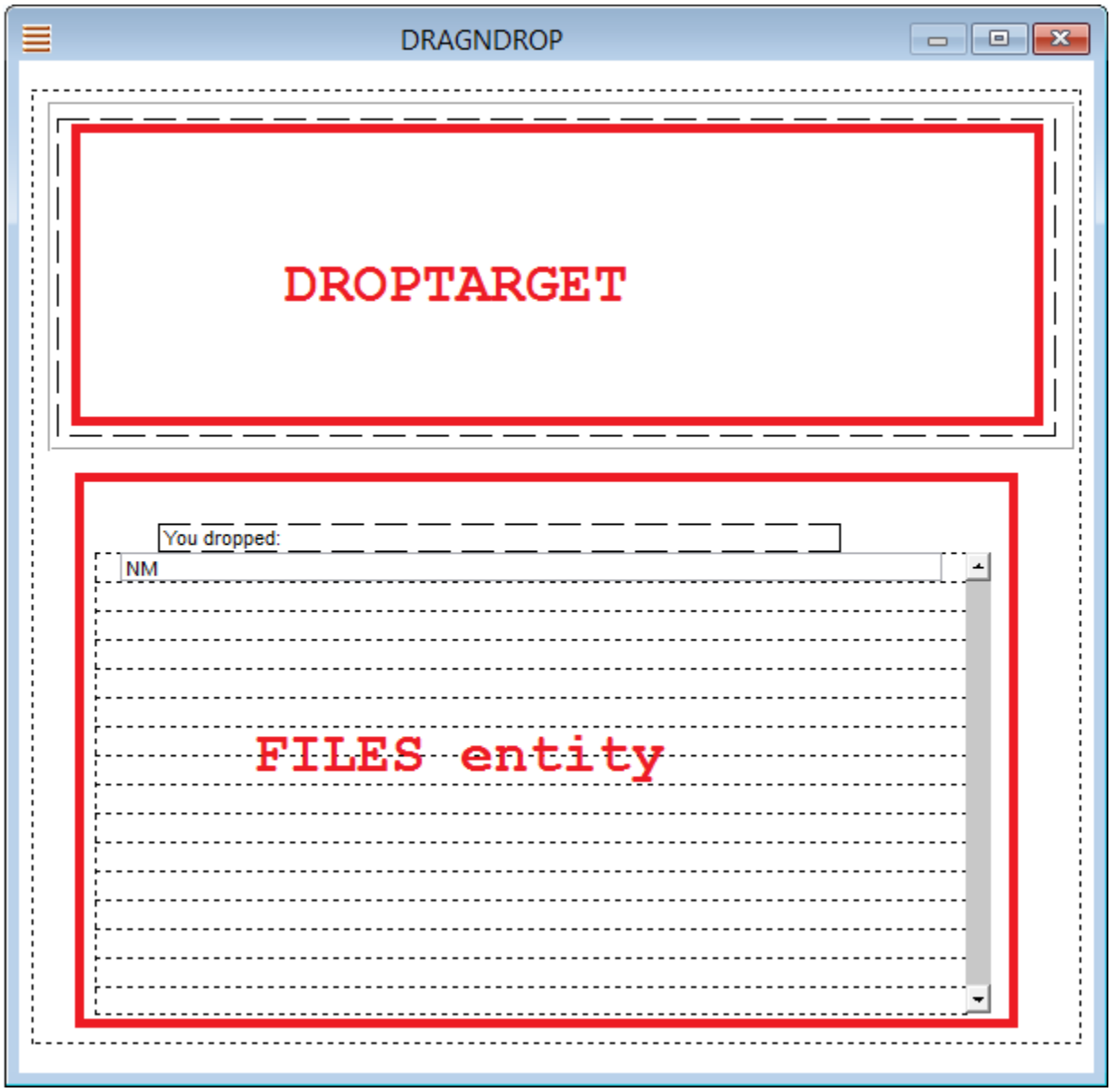
since the result will of a type string. Made the interface a shorthand C* just because I do not know how long the string of filenames will be and the default of 40 characters will probably not cut it. Now compile the component and test it by dragging something from Windows Explorer to the droptarget. As you may have noticed that did not work at all. No worries you need to make one small change and it will work. DoubleClick the droptarget widget to open the properties and then navigate to the widget properties. In the valrep list enter a value named “file” and leave the representation empty.



When you compile it and test it again you will notice that the mouse pointer changes from the no-entry sign to a arrow with a plus sign when you hover over the drop target. You can drop it, but it will not do very much.

How to get to the goodies

Dropping something on the drop target will fire the value changed trigger. Here you can put the code to verify if the dropped elements are actually the files that you want and process them further. In my component I have created an entity FILES in the DM application model and added a field named NM of type string and set its interface to C*. The component looks like this:



Next I opened the value changed trigger of the DROPTARGET field and added the following:

```
variables
   string v_nm
endvariables

while <$fieldname>.<$entname> != ""
   getitem v_nm, <$fieldname>.<$entname>, 1
   delitem <$fieldname>.<$entname>, 1

   if ($lfileexists(v_nm))
      creocc "FILES"
      NM.FILES = v_nm
   endif
endwhile

sort "FILES", "NM"

; reset the background image
<$fieldname>.<$entname> = "image"
```

In the first line I declared a variables block holding a string variable named v_nm. In the fifth line the forlist command extracts a value from my droptarget field and inserts it in the variable. After I checked if it the file exists, I add a record to my FILES entity and adds the variable value to the NM field in the entity.

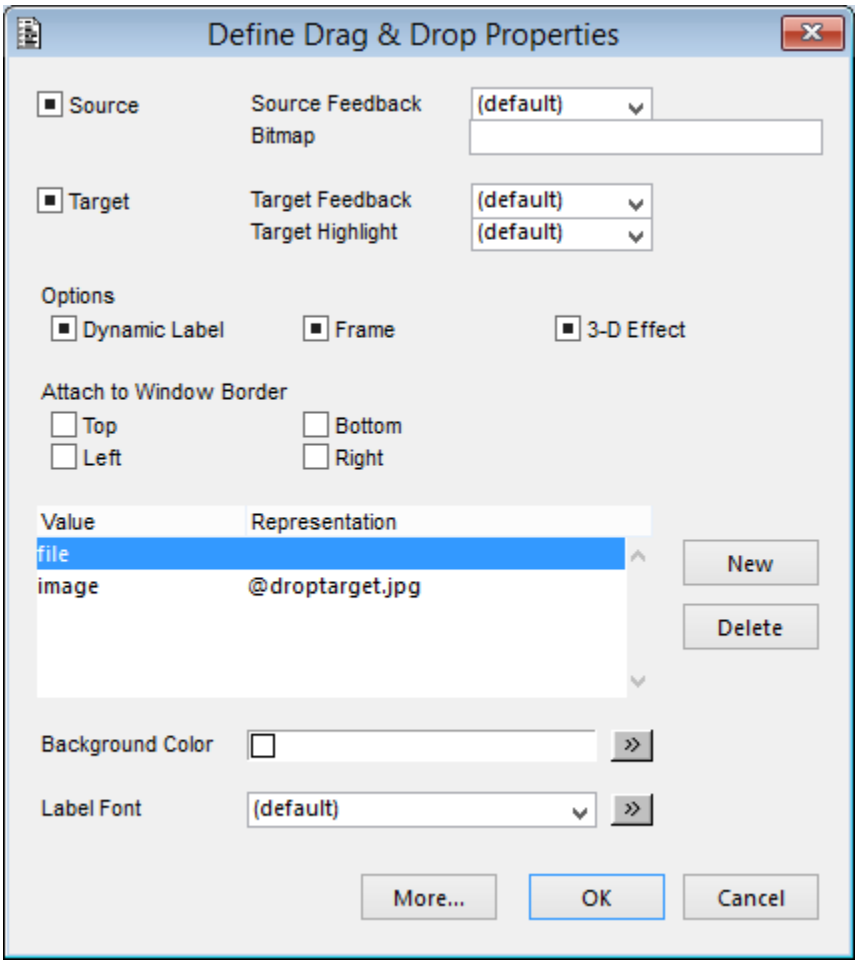
The reference to <$fieldname>.<$entname> will be replaced during compilation with the actual fieldname. In my case DROPTARGET.DE.

If you are using Uniface prior to 9.5.01 you cannot use the forlist command. You should use a combination of getitem, delitem. I am assuming you know how to do this, if not here is an example:

```
variables
   string v_nm
endvariables

while <$fieldname>.<$entname> != ""
   getitem v_nm, <$fieldname>.<$entname>, 1
   delitem <$fieldname>.<$entname>, 1

   if ($lfileexists(v_nm))
      creocc "FILES"
      NM.FILES = v_nm
   endif
endwhile

sort "FILES", "NM"

; reset the background image
<$fieldname>.<$entname> = "image"
```

Picture this

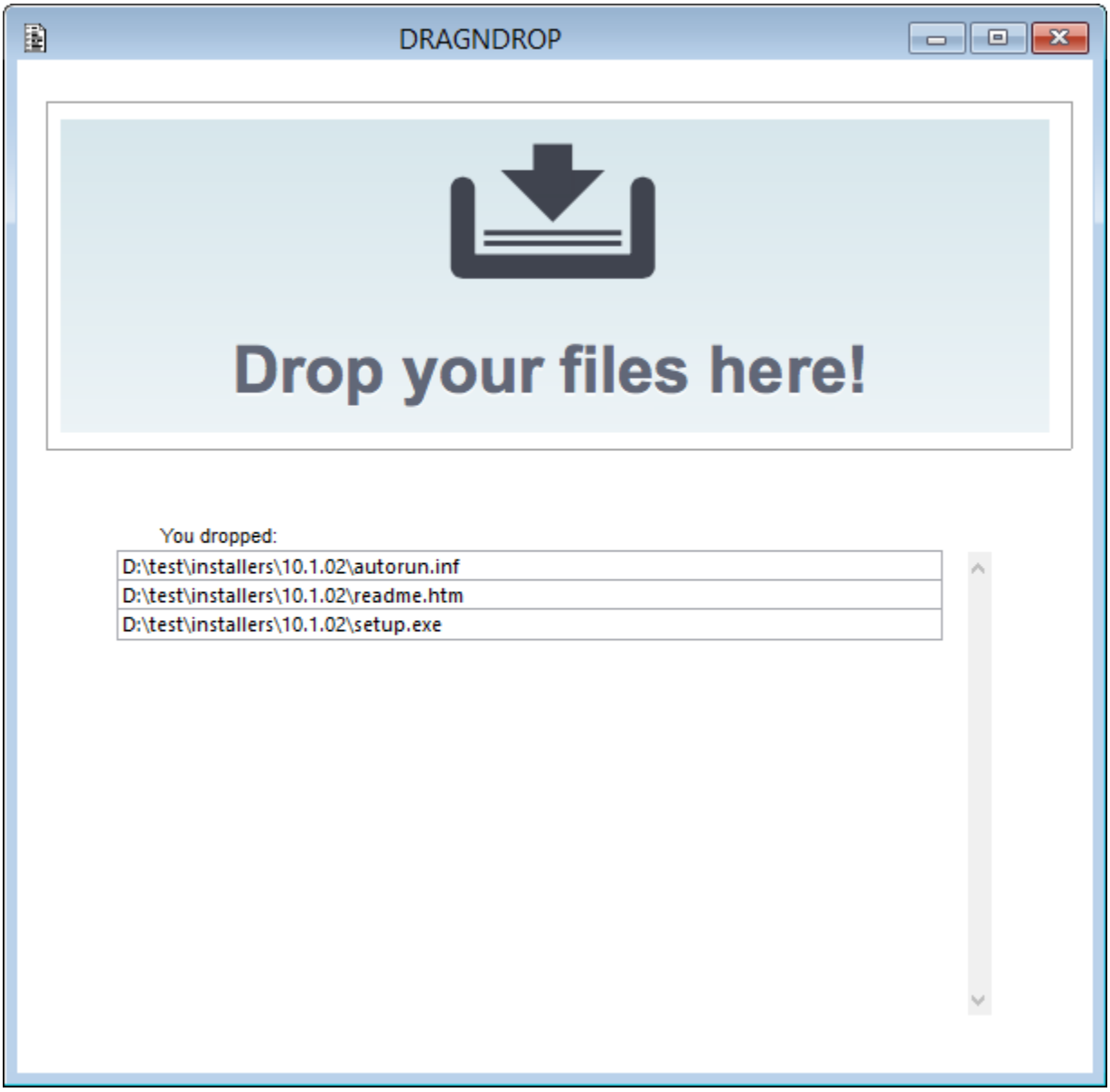
The last line in the previous code might have given it away, but I am using an image to highlight the droparea. On the drop event this value is being reset to the list of files, so after processing I need to reset it. Otherwise I will be left with a grey target or even worse, a target screaming “!format” at me. In the same location where we created the file valrep just minutes ago, I have added a value “image” and a representation “@droptarget.png”. The @ symbol implies that we are loading an image file named droptarget.png from the project folder and displaying it in the widget. The properties now look like this:



Just get your own cool looking image file and play around with the properties to get it right. To make sure the image is loaded on startup of the component we need to add two more lines to our component. In the EXECUTE trigger we add:

```
DROPTARGET.DE = "image"
edit
```

My component finally looks like this:



Download for this drag n drop sample