User Interface - Bug #3560

widget handles in the 4GL will compile with any valid attribute name even if it is not valid for that widget type

05/03/2018 11:24 AM - Greg Shah

Status:	New	**Start date:**	05/03/2018
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 05/03/2018 11:25 AM - Greg Shah

4GL allows (I think) any attribute/method to be specified for a widget reference, i.e.:

```
def var ch as char.
form ch with frame f1.
ch:first-buffer in frame f1.
```

will compile in 4GL.

Of course the runtime for this code will be just a raised 'not a queryable attribute' error in 4GL, but FWD will fail compilation of any such code, because we restrict the ch widget reference to that widget's type - FWD doesn't treat this expression as a handle, we are working directly with the resource.

There might be cases where automatically-generated code may emit code like this (maybe with a NO-ERROR clause to disregard unqueryable attributes/methods), where it will not be that simple to comment-out the code. A solution would be to rely on widget.asWidgetHandle() to access the attributes/methods in all cases (but this will incur a performance hit in some tightly-ran reports/jobs), or to emit these only if we know (at conversion time) the widget's type and if that attr/method is not available for the widget.

This was originally seen in #3523.

#3 - 05/03/2018 11:29 AM - Greg Shah

We do intend to implement some handle type-analysis in the future. This would enable us to find and warn about these cases. Then we could convert the broken cases differently so that it would fail at runtime in the same way.

That should limit the broken compilation to a very small set, since it is neither recommended nor common for the same handle to be used for multiple resource types. In that scenario, we still might be able to detect if one of the possible types was allowed to use that attribute. At a minimum, we

could warn about this and let the user clean it up.

More aggressive runtime solutions should be limited so that we don't affect performance.