Base Language - Feature #1781

add missing support for the WIDGET-HANDLE builtin function

10/30/2012 09:31 AM - Greg Shah

Status:	Closed	**Start date:**	09/18/2013
Priority:	Normal	**Due date:**	10/04/2013
Assignee:	Hynek Cihlar	**% Done:**	100%
Category:		**Estimated time:**	40.00 hours
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to Base Language - Bug #2183: fix resource save/restoration when used...	**Closed**	**09/18/2013**	**10/04/2013**

History

#1 - 10/31/2012 02:56 PM - Greg Shah

- Target version set to Milestone 12

#2 - 02/20/2013 04:28 PM - Greg Shah

- Subject changed from add missing support for the WIDGET-HANDLE attribute to add missing support for the WIDGET-HANDLE builtin function

#3 - 02/20/2013 06:20 PM - Greg Shah

The use cases (and even the description of the function) seem to be identical to the HANDLE builtin function. A testcase should be checked to determine if these are synonyms. If so, then the conversion can be identical.

The following will probably be sufficient:

```
def var h1 as handle.
def var h2 as handle.
def var h3 as handle.
def var txt-hndl as character.

h1 = this-procedure.
txt-hndl = string(h1).
h2 = handle(txt-hndl).
h3 = widget-handle(txt-hndl).

if (h1 eq h2) and valid-handle(h2) and (h2 eq h3) and valid-handle(h3) then
   message "HANDLE() and WIDGET-HANDLE() are synonyms!".
else
   message "YUCK!".
end.
```

#4 - 02/20/2013 06:20 PM - Greg Shah

- Target version changed from Milestone 12 to Milestone 7

#5 - 02/20/2013 06:57 PM - Greg Shah

After a quick review of the handle.java class and the builtin_functions.rules, I can see the following:

1. WIDGET-HANDLE is already supported as handle.fromString() in both conversion and runtime.
2. HANDLE is not yet supported. Let's add that.
3. The implementation is OK, but I think it could be a bit better:

a. We currently use the hashcode of each instance as the key in a HashMap. The javadoc says this is a "weak hashmap" but that is not the case.

b. The map gets cleaned only on client exit, memory problems may appear if the client is long-running. This should probably be dealt with better, but how?

c. handle just uses the Object.hashCode() (if the contained value is null) or the value's hashcode (if not null) as its hashcode. However, using hashcode as a key is not safe since it is possible (though very unlikely) to have more than 1 object with the same hashcode. In particular, the hashcode is a 32-bit int and when Object.hashCode() uses 64-bit memory addresses as their hashcodes, the result is lossy and can conflict. It would be better to just register a new int for this mapping purpose when a handle is turned to a string.

d. The 4GL needs to be checked to see if the format string used can make a difference in the mapping. For example, does a zero padded format string map a handle differently than a non-padded format string. If so, then the string itself should be used as the key instead of an Integer instance.

e. There is a TODO about "Should it be maintained by the DELETE OBJECT/PROCEDURE statements too?". The answer is probably no, since the 4GL docs suggest that you can delete a handle but then still use widget-handle to get a reference back BUT then the VALID-HANDLE would return FALSE. So the DELETE just affects the content of the handle, it doesn't make the handle go away. At least the javadoc needs update for this, if true.

#6 - 04/26/2013 07:43 AM - Greg Shah

- Project changed from User Interface to Base Language

#7 - 04/26/2013 09:49 AM - Greg Shah

- Start date set to 08/07/2013

- Due date set to 08/09/2013

- Assignee set to Greg Shah

#8 - 05/02/2013 12:22 PM - Greg Shah

- Due date changed from 08/09/2013 to 08/16/2013

- Start date changed from 08/07/2013 to 08/14/2013

#9 - 09/18/2013 09:41 AM - Greg Shah

- Target version changed from Milestone 7 to Milestone 11

- Assignee changed from Greg Shah to Hynek Cihlar

#10 - 09/18/2013 09:42 AM - Greg Shah

- Start date changed from 08/14/2013 to 09/18/2013

- Estimated time changed from 16.00 to 40.00

- Due date changed from 08/16/2013 to 09/30/2013

#11 - 09/18/2013 09:43 AM - Greg Shah

- Due date changed from 09/30/2013 to 10/04/2013

#12 - 09/18/2013 07:09 PM - Hynek Cihlar

- Status changed from New to WIP

#13 - 10/01/2013 05:59 PM - Hynek Cihlar

Ad 2
handle method is already recognized and parsed. Only emitting of handle.toString seems to be missing.

Ad 3
Because the semantics of Object.hashCode allows two distinct objects to have equal hash codes, hashCode should not be used for the purpose of resource identification. Instead IDs should be generated from a number sequence.
When a resource is deleted, the corresponding entry from the ID->handle map should be removed to prevent memory leak.
The above probably leads to the need to store the resource ID directly in the resource (WrappedResource class?).

Other Progress observations, tested on OpenEdge 10.2B:
- the data types handle and widget-handle are functionally equal
- the methods handle and widget-handle are functionally equal
- handle("123") equals handle("   123   ") assuming "123" is a valid resource ID
- handle("123") equals handle("000123") assuming "123" is a valid resource ID
- handle("000 123") causes "Invalid character in numeric input"
- string(handle("123")) equals "0" assuming "123" is not a valid resource ID or an ID of a deleted resource
- max length of resource ID is cca 240 characters
- two distinct uninitialized handles are equal
- two distinct zero-handles (handles created from an invalid resource ID or an ID of a deleted resource) are equal
- an uninitialized and a zero-handle are not equal

With respect to the Progress semantics:
- handle.compareTo should not be implemented in terms of handle.hashCode or resource hashCode equality but in terms of resource ID equality
- a handle constructed from an invalid resource ID should construct a zero-handle (a handle with null resource reference returning "0" from its toString method)

#14 - 10/01/2013 06:03 PM - Greg Shah

This is a useful set of findings.

Don't forget to test out uninitialized handles and zero handles compared to handles that are the unknown value. Also, make sure to test what VALID-HANDLE returns for uninitialized handles and zero handles.

#15 - 10/01/2013 06:05 PM - Greg Shah

Also, for security purposes, we should not make the resource ID algorithm a simple sequence. It should be randomized such that malicious code cannot anticipate the resource IDs that will be generated and thus cannot obtain access to a resource that they should not have.

#16 - 10/02/2013 03:18 PM - Hynek Cihlar

Greg Shah wrote:

> This is a useful set of findings.
>
> Don't forget to test out uninitialized handles and zero handles compared to handles that are the unknown value. Also, make sure to test what VALID-HANDLE returns for uninitialized handles and zero handles.

The meaning of the mentioned "uninitialized" handle was in fact the same as a handle that is the unknown value. Still for the sake of completeness I did compare handles with the unknown value created in different ways (never assigned value; created valid and assigned ?; created valid, deleted and assigned ?). **All the handles with the unknown values I was able to come up with were equal.**

VALID-HANDLE returns False for zero-handles and unknown (or uninitialized) handles.

#17 - 10/02/2013 03:28 PM - Hynek Cihlar

Greg Shah wrote:

> Also, for security purposes, we should not make the resource ID algorithm a simple sequence. It should be randomized such that malicious code cannot anticipate the resource IDs that will be generated and thus cannot obtain access to a resource that they should not have.

Progress itself assigns resource IDs from a simple sequence. In addition, the following can be found in the HANDLE function reference documentation of OpenEdge 10.2B:
Caution:
Use this function only to convert a handle previously stored as a string value back to a valid handle. If you convert an arbitrary string to handle using this function and then reference the new handle, a system error will occur. If you use the VALID-HANDLE function to validate a handle generated from an arbitrary string value, a system error will occur.

Do we still want the randomized resource IDs?

#18 - 10/02/2013 03:38 PM - Greg Shah

> Do we still want the randomized resource IDs?

Yes, I think it still makes sense. We may find some 4GL code that somehow relies upon this simple sequence behavior. If so, we will have to change it. But it would be a very bad practice.

> Use this function only to convert a handle previously stored as a string value back to a valid handle. If you convert an arbitrary string to handle using this function and then reference the new handle, a system error will occur. If you use the VALID-HANDLE function to validate a handle generated from an arbitrary string value, a system error will occur.

I have read this before, but it is not clear how accurate this warning is. It seems to me that so long as there is an existing handle with that resource ID, it doesn't matter how the input string was created.

One important question: is this resource ID only allocated if the handle was previous "stored as a string"?

#19 - 10/02/2013 03:42 PM - Hynek Cihlar

I found the following paragraph on the VALID-HANDLE function reference page:

If a handle is valid, it can still point to an obsolete object. That is, ABL can recycle a previously used handle value to reference a new object instance, leaving previously set handles with the same value pointing to older objects. So you can recognize that seemingly identical handles are actually handle copies that point to different objects, ABL supports the UNIQUE-ID attribute on some types of object handles. For more information, see the UNIQUE-ID attribute reference entry.

(1) I am not sure I completely understand the above, and (2) I have a feeling that it contradicts with the following on the UNIQUE-ID attribute reference page:

The value of the HANDLE attribute is guaranteed to be unique among the HANDLE attributes for all object instances in an ABL session.

An expert comment is very welcome!

#20 - 10/02/2013 04:02 PM - Hynek Cihlar

Greg Shah wrote:

> Do we still want the randomized resource IDs?

Yes, I think it still makes sense. We may find some 4GL code that somehow relies upon this simple sequence behavior. If so, we will have to change it. But it would be a very bad practice.

> Use this function only to convert a handle previously stored as a string value back to a valid handle. If you convert an arbitrary string to handle using this function and then reference the new handle, a system error will occur. If you use the VALID-HANDLE function to validate a handle generated from an arbitrary string value, a system error will occur.

I have read this before, but it is not clear how accurate this warning is. It seems to me that so long as there is an existing handle with that resource ID, it doesn't matter how the input string was created.

One important question: is this resource ID only allocated if the handle was previous "stored as a string"?

It looks like you are correct. The following code sample indicates that the IDs are pre-allocated.

```
  DEFINE VARIABLE hAlloc1 AS HANDLE.
  DEFINE VARIABLE hAlloc1Str AS CHARACTER.
  DEFINE VARIABLE hAlloc2 AS HANDLE.
  DEFINE VARIABLE hGuessedAlloc2 AS HANDLE.

  CREATE CALL hAlloc1.
  hAlloc1Str = STRING(hAlloc1).
  CREATE CALL hAlloc2.
  hGuessedAlloc2 = HANDLE(STRING(INT64(hAlloc1Str) + 1)).
  if (hGuessedAlloc2 eq hAlloc2) then
    message "Schrodinger's cat dies!".
```

The program outputs "Schrodinger's cat dies!"

#21 - 10/02/2013 04:08 PM - Greg Shah

> (1) I am not sure I completely understand the above,

Sadly, I don't either. I wonder if a "handle value" is just a slot in some array of data structures. When they "convert it to a string", they might just return its index into this array.

Later on, if they re-use the slot for a completely different handle, then it might have the same resource ID but since it points to a different resource, it would have a different UNIQUE-ID.

> leaving previously set handles with the same value pointing to older objects

This part doesn't make sense to me. I don't understand how you might get two handles with the same resource ID that are BOTH valid (pointing to a resource you can use) at the same time. Clearly, I don't have the full understanding here.

> (2) I have a feeling that it contradicts with the following on the UNIQUE-ID attribute reference page:
>
> The value of the HANDLE attribute is guaranteed to be unique among the HANDLE attributes for all object instances in an ABL session.

I suspect this is a typo. They probably meant:

> The value of the UNIQUE-ID attribute is guaranteed to be unique among the HANDLE attributes for all object instances in an ABL session.

I think you will need to try additional tests to try to find the answers. It may require loops to create large numbers of handles (and possibly delete them too)...

I have added Constantin as a watcher in case he has some other insight. He is really our handle expert.

> The program outputs "Schrodinger's cat dies!"

Hmmm. I thought it was impossible to know if Schrodinger's cat is alive or not. :)

> It looks like you are correct. The following code sample indicates that the IDs are pre-allocated.

The Progress developers have a poor understanding of security. Actually, they have poor design capabilities it most areas of computer science. The implementations they create are usually pretty flawed. I find that assuming the worst about their code is usually the correct assumption.

#22 - 10/15/2013 01:01 PM - Hynek Cihlar

- File hc_upd20131012a.zip added

Review request.

The attached changes implement all the points except for the improvements in resource ID allocation, ID-handle mapping and cleanup. Unless there are any objections I will track these under the related issue #2183.

#23 - 10/16/2013 10:58 AM - Greg Shah

Code Review 1012a

1. The testcases should not be in the same update zip.

2. There should not be a p2j/ directory in the update zip.

3. The builtin_functions.rules needs a history entry.

#24 - 10/16/2013 11:28 AM - Hynek Cihlar

- File hc_upd20131016b.zip added

- Status changed from WIP to Review

- File hc_upd20131016a.zip added

Resolved the issues from review.
Attached hc_upd20131016a.zip with p2j changes and hc_upd20131016b.zip with testcases changes.

#25 - 10/16/2013 11:51 AM - Greg Shah

Sorry, I should have seen this other thing before: all entries in builtin_functions.rules should be in alphabetical order so they are easy to find.

Please make that change, upload the fixed update and go into conversion regression testing.

Also, you can check in your testcases now.

#26 - 10/17/2013 04:52 PM - Hynek Cihlar

- File hc_upd20131017a.zip added

Built-in functions ordered alphabetically, see the attached hc_upd20131017a.zip.

#27 - 10/17/2013 05:02 PM - Greg Shah

It looks good. Please conversion regression test this.

#28 - 10/17/2013 05:04 PM - Hynek Cihlar

Yes, I forgot to mention that the regression test is in progress.

#29 - 10/18/2013 12:46 PM - Hynek Cihlar

Regression tests passed.

#30 - 10/18/2013 12:56 PM - Greg Shah

Check it in and distribute it.

#31 - 10/18/2013 02:04 PM - Hynek Cihlar

- % Done changed from 0 to 100

#32 - 10/18/2013 05:39 PM - Hynek Cihlar

Committed to p2j_repo as revision 10401.

#33 - 10/20/2013 12:09 PM - Greg Shah

- Status changed from Review to Closed

#34 - 11/16/2016 12:07 PM - Greg Shah

- Target version changed from Milestone 11 to Cleanup and Stablization for Server Features

Files

hc_upd20131012a.zip	11.5 KB	10/15/2013	Hynek Cihlar
hc_upd20131016a.zip	9.82 KB	10/16/2013	Hynek Cihlar
hc_upd20131016b.zip	1.47 KB	10/16/2013	Hynek Cihlar
hc_upd20131017a.zip	9.82 KB	10/17/2013	Hynek Cihlar