Base Language - Bug #2294

In Progress the variable initializer ignores the declared decimal precision

04/29/2014 10:57 AM - Hynek Cihlar

Status:	Closed	**Start date:**	04/29/2014
Priority:	Normal	**Due date:**	
Assignee:	Hynek Cihlar	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Base Language - Bug #2133: fix precision for decimal, dynamic-exte...	**Closed**	**01/21/2014**	**01/27/2014**

History

#1 - 04/29/2014 10:58 AM - Hynek Cihlar

In Progress the variable initializer doesn't care much about the declared decimal precision, see below.

```
def var d1 as decimal extent decimals 1 init [1.55].
message "d1[1] = 1.55" d1[1] = 1.55.
d1[1] = 1.55.
message "d1[1] = 1.6" d1[1] = 1.6.

def var d2 as decimal decimals 1 init 1.55.
message "d2 = 1.55" d2 = 1.55.
d2 = 1.55.
message "d2 = 1.6" d2 = 1.6.

def var d3 as decimal decimals 1 init 1.55.
message "d3 = 1.55" d3 = 1.55.
d3 = d3.
message "d3 = 1.6" d3 = 1.6.
```

outputs:

```
d1[1] = 1.55 yes
d1[1] = 1.6 yes
d2 = 1.55 yes
d2 = 1.6 yes
d3 = 1.55 yes
d3 = 1.6 yes
```

In the converted code, first the initial value is set and then the variable is set with the declared precision.

A possible solution could be to extend the logic of decimal to allow setting the precision for future value updates.

#2 - 05/01/2014 01:16 AM - Hynek Cihlar

- Status changed from New to WIP

#3 - 05/05/2014 01:22 PM - Hynek Cihlar

- File hc_upd20140505b.zip added

Please see the attached changes and provide feedback.

The solution seems to be functionally correct, however it introduces new public methods in the decimal class with yet another special decimal precision handling. The question is whether this special handling should not be part of the existing setPrecision methods.

The attached changes are on top of hc_upd20140430a.zip from #2133.

#4 - 05/06/2014 09:56 AM - Constantin Asofiei

I don't think is worth having both decimal.setPrecision and decimal.setLazyPrecision, because there is no case when setting the precision immediately, correct?

As your API changes are pretty major, please check the MAJIC's srcnew/java code for usage of decimal.setPrecision - and make sure the usage is correct with your findings (i.e. the hand-coded logic is OK with the new "precision is actually set on the next assignment" rule).

About the TODO in setTemporaryPrecision: this is used for shared variables, so you will need to check this, too.

#5 - 05/06/2014 12:37 PM - Hynek Cihlar

Constantin Asofiei wrote:

> I don't think is worth having both decimal.setPrecision and decimal.setLazyPrecision, because there is no case when setting the precision immediately, correct?
>
> As your API changes are pretty major, please check the MAJIC's srcnew/java code for usage of decimal.setPrecision - and make sure the usage is correct with your findings (i.e. the hand-coded logic is OK with the new "precision is actually set on the next assignment" rule).

In the converted code the only use case for setPrecision seems to be the variable definition. If I find the expected usage (precision-on-next-assignment) in the hand-crafted MAJIC code, can I assume there cannot be different use case in the future? I can imagine the behavior can be pretty surprising even when properly documented.

> About the TODO in setTemporaryPrecision: this is used for shared variables, so you will need to check this, too.

I will look at the shared variables and create some testcases to cover them.

#6 - 05/06/2014 12:46 PM - Hynek Cihlar

The only occurrence of setPrecision is in srcnew/java/aero/timco/majic/item/Toolissu.java. There it is used the same way as in the converted code, i.e. with the expected precision-on-next-assignment behavior.

#7 - 05/07/2014 02:23 AM - Constantin Asofiei

Hynek Cihlar wrote:

> In the converted code the only use case for setPrecision seems to be the variable definition. If I find the expected usage (precision-on-next-assignment) in the hand-crafted MAJIC code, can I assume there cannot be different use case in the future? I can imagine the behavior can be pretty surprising even when properly documented.

When writing Java code by hand (which needs to be 4GL-compatible), the developer needs to be careful to use the APIs as they are supposed to. So yes, we can only expect that the API is used in the proper way.

Something else you should cover: the variable definition with an INIT clause, i.e. def var d as dec decimal 2 init 1.22555. As I recall, we emit the initialization value directly at the c'tor and the precision is emitted via a decimal.setPrecision call.

> The only occurrence of setPrecision is in srcnew/java/aero/timco/majic/item/Toolissu.java. There it is used the same way as in the converted code, i.e. with the expected precision-on-next-assignment behavior.

OK, this actually is a converted 4GL program which is manually edited to add some logging. So if you don't add a new API and just change the setPrecision to act how 4GL does, then no changes are needed in srcnew/java code.

#8 - 05/07/2014 05:22 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > In the converted code the only use case for setPrecision seems to be the variable definition. If I find the expected usage (precision-on-next-assignment) in the hand-crafted MAJIC code, can I assume there cannot be different use case in the future? I can imagine the behavior can be pretty surprising even when properly documented.
>
> When writing Java code by hand (which needs to be 4GL-compatible), the developer needs to be careful to use the APIs as they are supposed to. So yes, we can only expect that the API is used in the proper way.

Ok.

> Something else you should cover: the variable definition with an INIT clause, i.e. def var d as dec decimal 2 init 1.22555. As I recall, we emit the initialization value directly at the c'tor and the precision is emitted via a decimal.setPrecision call.

Yes. This is actually the case described in the note 1.

> > The only occurrence of setPrecision is in srcnew/java/aero/timco/majic/item/Toolissu.java. There it is used the same way as in the converted code, i.e. with the expected precision-on-next-assignment behavior.

> OK, this actually is a converted 4GL program which is manually edited to add some logging. So if you don't add a new API and just change the setPrecision to act how 4GL does, then no changes are needed in srcnew/java code.

Ok.

#9 - 05/11/2014 04:41 PM - Hynek Cihlar

- File hc_upd20140511a.zip added

Please review the attached change set. It passes the companion uast/testcases which include cases for setTempPrecision, see the committed files var_init_precision.p, var_init_precision_proc1.p, var_init_precision_proc2.p. Also it contains a fix for correct setPrecision call during variable initialization, see variable_definitions.rules.

I have submitted the changes to regression testing.

#10 - 05/11/2014 05:02 PM - Hynek Cihlar

- Status changed from WIP to Review

#11 - 05/12/2014 11:14 AM - Constantin Asofiei

Hynek, the changes on 0511a.zip look good. How was the regression testing?

#12 - 05/15/2014 11:26 AM - Hynek Cihlar

The regression test (after several attempts) passed.

#13 - 05/15/2014 11:31 AM - Greg Shah

Good! Check in and distribute the changes.

#14 - 05/15/2014 12:55 PM - Hynek Cihlar

Committed to bzr revision 10532.

#15 - 05/15/2014 12:55 PM - Hynek Cihlar

- % Done changed from 0 to 100

#16 - 05/15/2014 04:22 PM - Greg Shah

- Status changed from Review to Closed

Files

hc_upd20140505b.zip	28.8 KB	05/05/2014	Hynek Cihlar
hc_upd20140511a.zip	28.6 KB	05/11/2014	Hynek Cihlar