Base Language - Bug #2133

fix precision for decimal, dynamic-extent variables

04/25/2013 11:58 AM - Constantin Asofiei

Status:	Closed	**Start date:**	01/21/2014
Priority:	Normal	**Due date:**	01/27/2014
Assignee:	Hynek Cihlar	**% Done:**	100%
Category:		**Estimated time:**	15.00 hours
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to Base Language - Feature #1620: implement EXTENT parameter support	**Closed**	**02/04/2013**
Related to Base Language - Feature #1621: implement dynamic/unspecified EXTEN...	**Closed**	**02/04/2013**
Related to Base Language - Bug #2284: Fix parameter initialization	**Closed**	**04/13/2014**
Related to Base Language - Bug #2291: Could not resolve closest legacy method...	**Closed**	**04/24/2014**
Related to Base Language - Bug #2292: Fix error handling when unfixed extent ...	**Closed**	**04/24/2014**
Related to Base Language - Bug #2293: Fixing an unfixed extent output paramet...	**Closed**	**04/24/2014**
Related to Base Language - Bug #2294: In Progress the variable initializer ig...	**Closed**	**04/29/2014**
Related to Database - Bug #2295: In Progress output parameters ignore decimal...	**WIP**	**04/29/2014**

History

#1 - 04/25/2013 12:01 PM - Constantin Asofiei

The solution would be to save the precision "by reference" (i.e. for each decimal array reference, save its precision). When ArrayAssigner.resize (and I think ArrayAssigner.copyOf) is called, do precision management by i.e. removing the precision for the old array and saving it for the new one.

#2 - 05/03/2013 10:02 AM - Greg Shah

- Due date set to 08/05/2013

- Assignee set to Ovidiu Maxiniuc

- Start date set to 08/02/2013

#3 - 01/21/2014 11:34 AM - Greg Shah

- Assignee changed from Ovidiu Maxiniuc to Hynek Cihlar

- Due date changed from 08/05/2013 to 01/27/2014

- Start date changed from 08/02/2013 to 01/21/2014

#4 - 02/23/2014 03:12 PM - Hynek Cihlar

The decimal precision is effectively lost when (1) unfixed decimal array is defined and later fixed with the EXTENT statement or when (2) a fixed array is reset to unfixed state again with the EXTENT statement or (3) during array assignment (whether simple assignment or passing array as function parameters).

The problem is that the precision information is not preserved for the type (array) definition, but only for the array components.

The solution, as already outlined above, is to store the precision information for each array instance and copy the precision information when one of the cases mentioned above occurs, i.e. the array instance is copied.

The affected methods are decimal.setPrecision(decimal[], int), ArrayAssigner.resize and ArrayAssigner.copyOf. These have to properly handle storage management of the precision information. ArrayAssigner already stores dynamic (indeterminate) arrays in a scopable work area as raw references. An implementation approach would be to extend the storage schema - instead of storing the raw reference, a wrapper instance would be stored containing the array reference and the precision information.

#5 - 02/23/2014 03:13 PM - Hynek Cihlar

- Status changed from New to WIP

#6 - 02/24/2014 01:47 PM - Hynek Cihlar

When the dynamic array is resized (EXTENT statement is encountered), old array reference is "deregistered" and new reference is "registered" from/to ArrayAssigner.WorkArea. The method ArrayAssigner.WorkArea.deregister contains a logic to deregister the reference from all scopes, but the register counterpart only registers to the scope on the top of the scope stack.

Can it ever happen to have the array reference registered to multiple scopes on the scope stack?

#7 - 02/24/2014 02:18 PM - Greg Shah

> Can it ever happen to have the array reference registered to multiple scopes on the scope stack?

Yes. I believe this happens when it is passed as a parameter to a user-defined function, internal procedure or an external procedure.

#8 - 02/24/2014 02:26 PM - Hynek Cihlar

Greg Shah wrote:

> > Can it ever happen to have the array reference registered to multiple scopes on the scope stack?
>
> Yes. I believe this happens when it is passed as a parameter to a user-defined function, internal procedure or an external procedure.

I'll create a test case for this. Eventually the re-registration will have to be a bit smarter to register into all respective scopes.

#9 - 02/27/2014 11:13 AM - Hynek Cihlar

Setting decimal precision in the DEFINE VARIABLE statement of an EXTENT with the missing number of array elements produces uncompilable Java code.

Example:

```
def var d1 as decimal extent decimals 1 init [1, 2, 3, 4, 5].
def var d2 as decimal extent 5 decimals 2 init [5,4,3,2,1].
def var d3 as decimal extent decimals 3.
```

```
def var d4 as decimal extent 1 decimals 4.
def var d5 like d1.
def var d6 like d2.
```

yields the following Java code (only relevant part shown):

```
         public void init()
         {
            ArrayAssigner.registerDynamicArray(d1);
            d1.setPrecision(1); // ERROR
            TransactionManager.register((Object) d1, (Object) d2, (Object) d3, (Object) d4, (Object) d5, (Obje
ct) d6);
            decimal.setPrecision(d2, 2);
            assignMulti(d3, new decimal(0));
            ArrayAssigner.registerDynamicArray(d3);
            d3.setPrecision(3); // ERROR
            assignMulti(d4, new decimal(0));
            decimal.setPrecision(d4, 4);
            ArrayAssigner.registerDynamicArray(d5);
            d5.setPrecision(1); // ERROR
            decimal.setPrecision(d6, 2);
         }
```

I am not sure how def var d3 as decimal extent 0 decimals 3. would behave, will test.

#10 - 02/27/2014 12:00 PM - Constantin Asofiei

Hynek Cihlar wrote:

> Setting decimal precision in the DEFINE VARIABLE statement of an EXTENT with the missing number of array elements produces uncompilable Java code.

When the number of array elements is missing, then we have a dynamic extent. And looks like there is a conversion problem here.

> I am not sure how def var d3 as decimal extent 0 decimals 3. would behave, will test.

extent 0 marks a scalar variable.

#11 - 02/27/2014 02:55 PM - Hynek Cihlar

Constantin Asofiei wrote:

> When the number of array elements is missing, then we have a dynamic extent. And looks like there is a conversion problem here.

Created issue #2250.

> extent 0 marks a scalar variable.

You are right, surprising behavior.

#12 - 03/09/2014 04:47 PM - Hynek Cihlar

There's a bug in ArrayAssigner.assignMulti(BaseDataType[], BaseDataType[]). When assigning a determinate array to a non-determinate array, ArrayAssigner.resize is called allocating new array. But since the assignMulti method doesn't return the newly allocated instance, the assignment has in this case no effect.

#13 - 03/10/2014 04:03 AM - Constantin Asofiei

Hynek Cihlar wrote:

> There's a bug in ArrayAssigner.assignMulti(BaseDataType[], BaseDataType[]). When assigning a determinate array to a non-determinate array, ArrayAssigner.resize is called allocating new array. But since the assignMulti method doesn't return the newly allocated instance, the assignment has in this case no effect.

OK, good catch. Go ahead and fix it, you will need a conversion change too, so that it emits like this:

```
target = ArrayAssigner.assignMulti(target, source)
```

If possible, emit this only when the source is not scalar, so you don't have to change all the assignMulti APIs to return the target.

#14 - 03/16/2014 04:54 PM - Hynek Cihlar

- File hc_upd20140316a.zip added

I am attaching the initial implementation for fixing the assignMulti method. The changes include the modification of assignMulti signature as well as the modification of the conversion for extent-to-extent assignment.

The conversion logic includes a check whether an expression is of an extent type. I put that logic into ExpressionConversionWorker.ExpressionHelper.isExpressionOfExtentType method. Is this the right place for such logic? Is it the right approach to find the extent for an expression or is there a simpler solution? Because of these unknowns I have so far implemented only two cases - a variable reference and function call expressions.

The changes don't include the final Javadocs and file headers yet.

#15 - 03/18/2014 04:09 AM - Constantin Asofiei

Hynek, about 0316a.zip. The implementation of isExpressionOfExtentType should check for extent annotations, not only for a KW_EXTENT child; as you already found, extent 0 defines a scalar var.

Open the generated .ast file and check the existing annotations for a variable definition, extent function, extent field, etc: there should be an extent annotation for all these cases. And I don't think you need to explicitly check for each case (var def, func, table field, etc): if the refid points to an AST with the extent annotation, then is enough to assume that this is an extent.

#16 - 03/18/2014 11:02 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek, about 0316a.zip. The implementation of isExpressionOfExtentType should check for extent annotations, not only for a KW_EXTENT child; as you already found, extent 0 defines a scalar var.
>
> Open the generated .ast file and check the existing annotations for a variable definition, extent function, extent field, etc: there should be an extent annotation for all these cases. And I don't think you need to explicitly check for each case (var def, func, table field, etc): if the refid points to an AST with the extent annotation, then is enough to assume that this is an extent.

Yes I noted the extent annotation but wasn't sure I could rely on it in all the cases. I will change the implementation of isExpressionOfExtentType according to your suggestion. Unless you are against it, I will keep the isExpressionOfExtentType method even though the implementation simplified significantly.

#17 - 03/18/2014 01:59 PM - Constantin Asofiei

Hynek Cihlar wrote:

> Unless you are against it, I will keep the isExpressionOfExtentType method even though the implementation simplified significantly.

I don't think isExpressionOfExtentType is needed, just add the dereferencing code and the extent annotation check in the TRPL rules.

#18 - 03/25/2014 11:47 AM - Hynek Cihlar

- File hc_upd20140324a.zip added

Attached are changes based on the feedback from last review - isExpressionOfExtentType was removed. I submitted the code to regression testing. Please review.

#19 - 03/25/2014 02:11 PM - Constantin Asofiei

Hynek, the changes look good. How is the decimal precision working?

#20 - 03/26/2014 10:01 AM - Hynek Cihlar

I have finished a release-candidate implementation for the decimal precision. I am testing it and also looking for edge-cases so it is in-line with Progress behavior.

#21 - 03/26/2014 03:09 PM - Hynek Cihlar

The converted code doesn't give the correct error message when assigning a fixed array to an indeterminate and already fixed array. The correct message should be 'Indeterminate extent is already fixed to a dimension of x. (13738)' where x is the array length.

I fixed this in the ArrayAssigner as part of the decimal precision fix.

#22 - 03/26/2014 03:51 PM - Hynek Cihlar

In Progress, function/procedure fixed-extent parameter's components are initialized to unknown values when an unfixed indeterminate extent is passed in. P2J on the other hand initializes the components to zero values.

The following sample prints "2 ? ?" in Progress but "2 0 0" in P2J.

```
function foo returns int (input-output j as int extent 2).
  message extent(j) j[1] j[2].
end.
def var i as int extent.
foo(input-output i).
```

When an extent is fixed using the extent statement, the components are initialized to zero values - both in Progress and P2J.

#23 - 03/26/2014 04:08 PM - Hynek Cihlar

The following code sample causes the error message 'Calling procedure /home/hc/p34141_Untitled8.ped cannot input an indeterminate extent

parameter if the sub-procedure foo /home/hc/p34141_Untitled8.ped parameter
is a fixed size extent. (11421)' in Progress but not in P2J. The execution is not interrupted.

```
function foo returns int (input p as int extent 2).
end.
def var k as int extent.
foo(input k).
```

#24 - 03/26/2014 04:15 PM - Constantin Asofiei

Hynek Cihlar wrote:

> The following code sample causes the error message 'Calling procedure /home/hc/p34141_Untitled8.ped cannot input an indeterminate extent parameter if the sub-procedure foo /home/hc/p34141_Untitled8.ped parameter
> is a fixed size extent. (11421)' in Progress but not in P2J. The execution is not interrupted.
>
> [...]

Add a pause at the end of the program to be sure the message is really not displayed.

#25 - 03/26/2014 04:21 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > The following code sample causes the error message 'Calling procedure /home/hc/p34141_Untitled8.ped cannot input an indeterminate extent parameter if the sub-procedure foo /home/hc/p34141_Untitled8.ped parameter
> > is a fixed size extent. (11421)' in Progress but not in P2J. The execution is not interrupted.
> >
> > [...]
>
> Add a pause at the end of the program to be sure the message is really not displayed.

The same with pause - the error message is not displayed in P2J.

#26 - 03/26/2014 04:40 PM - Constantin Asofiei

Hynek Cihlar wrote:

> Constantin Asofiei wrote:
>
> > Hynek Cihlar wrote:
> >
> > > The following code sample causes the error message 'Calling procedure /home/hc/p34141_Untitled8.ped cannot input an indeterminate extent parameter if the sub-procedure foo /home/hc/p34141_Untitled8.ped parameter
> > > is a fixed size extent. (11421)' in Progress but not in P2J. The execution is not interrupted.
> > >
> > > [...]
> >
> > Add a pause at the end of the program to be sure the message is really not displayed.
>
> The same with pause - the error message is not displayed in P2J.

Go ahead and fix this in ControlFlowOps$InternalEntryCaller.valid.

#27 - 03/26/2014 04:48 PM - Constantin Asofiei

Hynek Cihlar wrote:

> In Progress, function/procedure fixed-extent parameter's components are initialized to unknown values when an unfixed indeterminate extent is passed in. P2J on the other hand initializes the components to zero values.

This one I think requires a conversion change: integer[] j = extj.getVariable(2); needs to be converted to integer[] j = extj.getVariable(2, true); in case of input-output parameters, where the true flag means to set the extent array elements to unknown.

#28 - 03/28/2014 04:24 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Go ahead and fix this in ControlFlowOps$InternalEntryCaller.valid.

For the function case, ControlFlowOps is not called. Perhaps it should be fixed somewhere in BlockManager.

Also I mentioned that the execution is not interrupted. To be more specific, the function body is not executed and the execution resumes just after the function call (!).

#29 - 03/29/2014 08:32 PM - Hynek Cihlar

- File hc_upd20140329b.zip added

Constantin Asofiei wrote:

> This one I think requires a conversion change: integer[] j = extj.getVariable(2); needs to be converted to integer[] j = extj.getVariable(2, true); in case of input-output parameters, where the true flag means to set the extent array elements to unknown.

Based on your suggestion, this is what I did. I added method OutputExtentParameter.getVariable(int, boolean). The method calls into ArrayAssigner.resize that handles the array allocation and initialization. I extended ArrayAssigner.resize as well to allow for "unknown value" or "default value" initialization depending on the context. Last I changed the conversion to call OutputExtentParameter.getVariable(int, boolean) for input-output parameters. This fixes the case of input-output parameters. See the attached hc_upd20140329b.zip.

Then I found out that a similar issue exists with output parameters - in Progress the array components are initialized to default values, but in P2J to unknown values. The output variable is actually initialized twice - first to correct default from the method OutputExtentParameter.getVariable(int) and second to incorrect unknown by calling assignMulti from the function block's init method.

May I suggest the following solution? To revert the change in OutputExtentParameter and ArrayAssigner removing the responsibility of initializing the parameter from the getVariable method and to modify the conversion to emit the initialization code into the function block's init method. That is the init method would contain assignMulti(varName, decimal.instantiateUnknown()) for input-output params and assignMulti(varName, decimal.instantiateDefault()) for output params.

#30 - 03/31/2014 03:29 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Constantin Asofiei wrote:
> For the function case, ControlFlowOps is not called. Perhaps it should be fixed somewhere in BlockManager.
>
> Also I mentioned that the execution is not interrupted. To be more specific, the function body is not executed and the execution resumes just after the function call (!).

Simple legacy function calls convert to java method calls, bypassing ControlFlowOps. But as the signature matches (in terms that both function call and definition pass an extent param), the java code will compile safely. I think the conversion rules should check the fact that either the func definition or the caller uses a dynamic extent argument and convert this into a ControlFlowOps.invokeFunction API call, instead of a java method call. BlockManager has no knowledge about the arguments passed by the caller, so it can' be used.

#31 - 03/31/2014 03:31 AM - Constantin Asofiei

Hynek Cihlar wrote:

> May I suggest the following solution? To revert the change in OutputExtentParameter and ArrayAssigner removing the responsibility of initializing the parameter from the getVariable method and to modify the conversion to emit the initialization code into the function block's init method. That is the init method would contain assignMulti(varName, decimal.instantiateUnknown()) for input-output params and assignMulti(varName, decimal.instantiateDefault()) for output params.

Go ahead, the approach looks OK.

#32 - 03/31/2014 02:56 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Constantin Asofiei wrote:
> > For the function case, ControlFlowOps is not called. Perhaps it should be fixed somewhere in BlockManager.
> >
> > Also I mentioned that the execution is not interrupted. To be more specific, the function body is not executed and the execution resumes just after the function call (!).
>
> Simple legacy function calls convert to java method calls, bypassing ControlFlowOps. But as the signature matches (in terms that both function call and definition pass an extent param), the java code will compile safely. I think the conversion rules should check the fact that either the func definition or the caller uses a dynamic extent argument and convert this into a ControlFlowOps.invokeFunction API call, instead of a java method call. BlockManager has no knowledge about the arguments passed by the caller, so it can' be used.

Ok, I will modify the conversion rules to invoke any function with one or more **input** extent parameters through ControlFlowOps and also implement

the "input parameter" validation in ControlFlowOps$InternalEntryCaller.valid.

#33 - 03/31/2014 07:14 PM - Hynek Cihlar

- File hc_upd20140329a.zip added

Attached is the final fix for the extent-to-extent assignment. The delta from the previous version (hc_upd20140324a.zip) is an identifier fix of the generated extent lvalue - target of the assignment.

The code passed regression testing. Please review.

#34 - 04/01/2014 07:09 AM - Constantin Asofiei

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > May I suggest the following solution? To revert the change in OutputExtentParameter and ArrayAssigner removing the responsibility of initializing the parameter from the getVariable method and to modify the conversion to emit the initialization code into the function block's init method. That is the init method would contain assignMulti(varName, decimal.instantiateUnknown()) for input-output params and assignMulti(varName, decimal.instantiateDefault()) for output params.
>
> Go ahead, the approach looks OK.

Sorry, I'm coming back on this one: when we can choose between emitting code and hiding it in the runtime, is best to hide it in the runtime. Thus, the assignMulti call which assigns either to default or unknown, should be hidden inside the OutputExtentParameter.getVariable. Before doing the changes, please post how the converted java code will look like, for INPUT, OUTPUT and INPUT-OUTPUT cases.

BTW, do your tests cover both function and procedure with extent parameters? We need to ensure the behaviour is the same in both cases, before doing the changes.

#35 - 04/02/2014 03:16 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Sorry, I'm coming back on this one: when we can choose between emitting code and hiding it in the runtime, is best to hide it in the runtime. Thus, the assignMulti call which assigns either to default or unknown, should be hidden inside the OutputExtentParameter.getVariable. Before doing the changes, please post how the converted java code will look like, for INPUT, OUTPUT and INPUT-OUTPUT cases.

OK, no problem. I'll change the implementation back to the initialization being part of the array allocation. See the attached DynamicExtentParamInit.java showing how the converted code will look like for functions and procedures with input, input-output and output parameters. Note the call extj.setParameter(j, true) for the input-output case which initializes the array components to unknown values. The extj.getVariable(2) call for the output case initializes the array components to zero values. Also note the missing explicit array assignments in the init functions of the function and procedure blocks.

> BTW, do your tests cover both function and procedure with extent parameters? We need to ensure the behaviour is the same in both cases, before doing the changes.

I include test cases with functions and internal procedures, no external procedures though.

#36 - 04/02/2014 03:17 PM - Hynek Cihlar

- File DynamicExtentParamInit.java added

File attachment for the #35 note above.

#37 - 04/02/2014 03:18 PM - Hynek Cihlar

Any update on the review request from the note #33?

#38 - 04/03/2014 02:02 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Note the call extj.setParameter(j, true) for the input-output case which initializes the array components to unknown values.

Shouldn't the initialization be at the getVariable call, not the setParameter? Because getVariable is the API which creates it.

> Any update on the review request from the note #33?

I guess the update is still WIP, right? Because it doesn't compile and the disabled rule at convert/variable_definitions.rules:488 should be removed, if you no longer need it (now you have a false and condition, to disable it). Otherwise:

1. the javadoc for ArrayAssigner.resize initToUnknown is confusing/incomplete: Initialization flag, see above for more info., but there is nothing "above".
2. the compile error at ArrayAssigner.assignMulti:129
3. OutputExtentParameter is OK, as it adds the getVariable(int extent, boolean initToUnknown) API, and not the setParameter(T[] parameter, boolean initToUnknown) - so you can keep it.

#39 - 04/03/2014 01:09 PM - Hynek Cihlar

- File DynamicExtentParamInit.java added

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Note the call extj.setParameter(j, true) for the input-output case which initializes the array components to unknown values.
>
> Shouldn't the initialization be at the getVariable call, not the setParameter? Because getVariable is the API which creates it.

Yes, of course, sorry for that. The attached file is the correct example.

#40 - 04/03/2014 01:23 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Any update on the review request from the note #33?
>
> I guess the update is still WIP, right? Because it doesn't compile and the disabled rule at convert/variable_definitions.rules:488 should be removed, if you no longer need it (now you have a false and condition, to disable it).

I think you reviewed the wrong file. The attached file in note 33 (hc_upd20140329a.zip) fixes the issue with extent to extent assignment for the cases when the passed in array gets resized.

The changes you reviewed was a request for review of the initial implementation of the extent array component initialization when passed in as a function/procedure parameter. Yes it doesn't compile and it lacks proper Javadocs as mentioned in the related note. I will continue on this initial implementation to provide the final solution. I think you can ignore it at this point.

#41 - 04/04/2014 05:00 AM - Constantin Asofiei

Hynek Cihlar wrote:

> I think you reviewed the wrong file.

Yes, it was the wrong file. What confused me was that I expected 0329b.zip to be an incremental update of 0329a.zip, that's why I looked at 0329b.zip.

About 0329a.zip:

- assignments.rules:222 to 236: these lines should be one indent to the right
- otherwise the changes look good. Have you already ran runtime testing with 0329a.zip update? If yes and it passed, go ahead and release it once you fix the formatting issue.

#42 - 04/04/2014 11:37 AM - Hynek Cihlar

Hynek Cihlar wrote:

> Yes, of course, sorry for that. The attached file is the correct example.

Constantin, is the latest DynamicExtentParamInit.java acceptable now?

#43 - 04/04/2014 12:03 PM - Hynek Cihlar

- File hc_upd20140404a.zip added

Regression test of 0329a.zip passed. Fixed the indent in 0329a.zip (see the attached 0404a.zip) and committed to revision 10505.

#44 - 04/04/2014 12:12 PM - Constantin Asofiei

Hynek Cihlar wrote:

> Hynek Cihlar wrote:
>
> > Yes, of course, sorry for that. The attached file is the correct example.
>
> Constantin, is the latest DynamicExtentParamInit.java acceptable now?

Yes, it looks OK.

#45 - 04/13/2014 04:21 PM - Hynek Cihlar

- Subject changed from fix precision for decimal, dynamic-extent variables to Fix parameter initialization

Because the problem of parameter initialization goes beyond the extents, I created a separate issue for it. See #2284.

#46 - 04/13/2014 05:27 PM - Hynek Cihlar

- Subject changed from Fix parameter initialization to fix precision for decimal, dynamic-extent variables

#47 - 04/24/2014 11:00 AM - Greg Shah

Is there anything left to do on this task?

#48 - 04/24/2014 11:37 AM - Hynek Cihlar

Yes, there is. I have an implementation in place, but because of the few secondary issues I found (which some of them are already solved) it is not yet stable.

The latest problem I found is that Progress is inconsistent in regard to honoring decimal precision in variable initializers and output parameters and I am working on a fix.

#49 - 04/24/2014 05:09 PM - Hynek Cihlar

I created new issue for the error handling of input fixed extent parameters (as of the note 23), the issue number is #2292.

#50 - 04/26/2014 02:35 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.

#51 - 04/26/2014 12:02 PM - Constantin Asofiei

Hynek Cihlar wrote:

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

Is the variable initializer using some default decimal precision value, instead of the user-defined one?

#52 - 04/26/2014 12:13 PM - Hynek Cihlar

- File variable_definitions.rules.diff added

- File decimal.java.diff added

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > In Progress the variable initializer doesn't care much about the declared decimal precision, see below.
>
> Is the variable initializer using some default decimal precision value, instead of the user-defined one?

Yes, it is using the default of ten decimal places.

I have a working (prototype) fix, see the attached diff files. The relevant changes in decimal.java.diff are the setPrecision(int) and setLazyPrecision methods.

#53 - 04/29/2014 11:00 AM - Hynek Cihlar

I created a new issue #2294 for the initializer ignoring declared decimal precision - it is related not only to extents and it would otherwise block the fix for the subject of this issue.

#54 - 04/30/2014 12:15 PM - Hynek Cihlar

- File hc_upd20140430a.zip added

- Status changed from WIP to Review

The attached code fixes decimal precision for extent variables and extent-to-extent assignment when rvalue is a parameter. Note that this change doesn't resolve the cases described in the issues #2294 and #2295.

Please review.

#55 - 05/02/2014 03:21 AM - Constantin Asofiei

Hynek, the logic looks good, but there are some formatting issues:

- assignments.rules:
  - line 229 is too long
- decimal.java:
  - you have a tab on line 612
- ArrayAssigner:
  - WorkArea.dynamicArrayPrecisions - you can use the diamond to avoid re-using the generic definition, i.e. new HashMap<> (the line is too long)
  - WorkArea - for the new methods, re-check the javadoc formatting

Also, please commit your testcases to bzr (the testcases project).

#56 - 05/05/2014 12:54 PM - Hynek Cihlar

- File hc_upd20140505a.zip added

Attached are the fixed formatting issues. Also, the related testcases are committed in the file testcases/uast/extent_decimal_precision.p.

I am submitting the changes to regression testing.

#57 - 05/06/2014 09:41 AM - Constantin Asofiei

The changes look good.

As a side note, when building testcases, instead of using this:

```
/* expect all 'yes' */
message "d1[1] = 1.1" d1[1] = 1.1.
```

you might find it useful to write the test like this:

```
if d1[1] <> 1.1 then message "test 'd1[1] = 1.1' is incorrect!".
```

This will avoid checking the screen/side-by-side comparison for non-UI features. And when you run the test, if a message is shown on screen/file, you know exactly where to look.

#58 - 05/06/2014 12:25 PM - Hynek Cihlar

Constantin Asofiei wrote:

> As a side note, when building testcases, instead of using this:
> [...]
> you might find it useful to write the test like this:
> [...]
>
> This will avoid checking the screen/side-by-side comparison for non-UI features. And when you run the test, if a message is shown on screen/file, you know exactly where to look.

Indeed this is more useful. Thanks for the tip!

#59 - 05/08/2014 12:38 PM - Hynek Cihlar

0505a passed regression test and was committed to bzr revision 10524.

#60 - 05/09/2014 04:15 PM - Hynek Cihlar

- % Done changed from 0 to 100

#61 - 05/12/2014 11:52 AM - Greg Shah

- Status changed from Review to Closed

#62 - 11/16/2016 12:07 PM - Greg Shah

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

Files

hc_upd20140316a.zip	31.5 KB	03/16/2014	Hynek Cihlar
hc_upd20140324a.zip	10.5 KB	03/25/2014	Hynek Cihlar
hc_upd20140329b.zip	22.5 KB	03/30/2014	Hynek Cihlar
hc_upd20140329a.zip	22.7 KB	03/31/2014	Hynek Cihlar
DynamicExtentParamInit.java	5.41 KB	04/02/2014	Hynek Cihlar
DynamicExtentParamInit.java	5.4 KB	04/03/2014	Hynek Cihlar
hc_upd20140404a.zip	10.5 KB	04/04/2014	Hynek Cihlar
decimal.java.diff	4.2 KB	04/26/2014	Hynek Cihlar
variable_definitions.rules.diff	1.13 KB	04/26/2014	Hynek Cihlar
hc_upd20140430a.zip	26.8 KB	04/30/2014	Hynek Cihlar
hc_upd20140505a.zip	26.8 KB	05/05/2014	Hynek Cihlar