Base Language - Bug #2592

improve support for implicit type conversions

06/19/2015 03:36 PM - Ovidiu Maxiniuc

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Ovidiu Maxiniuc	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Cleanup and Stablization for Server Features		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Bug #2293: Fixing an unfixed extent output paramet...	**Closed**	**04/24/2014**
Related to Database - Bug #2295: In Progress output parameters ignore decimal...	**WIP**	**04/29/2014**

History

#1 - 06/19/2015 03:38 PM - Ovidiu Maxiniuc

I found the following external procedure caller.p:

```
PROCEDURE P1
...
RUN servant.p ( INPUT TODAY, OUTPUT TT1.intDate )
```

where:

```
DEFINE TEMP-TABLE TT1
...
    FIELD intDate                        AS INTEGER    FORMAT ">>>>>9"
...
```

The servant.p:

```
DEF INPUT  PARAMETER a-date    AS   DATE.
DEF OUTPUT PARAMETER out-int   AS   CHAR FORMAT "X(5)"....
out-int = SUBSTRING (STRING ...
```

As you can see, they are declaring an external procedure having the second (OUTPUT) parameter as CHAR, but when calling this procedure they are using a different type of field (an INTEGER).

I did not tested locally, but caller expects the return value from this construct. So apparently, the P4GL supports such automatic type conversion/casting. We surely don't.

#2 - 06/19/2015 03:39 PM - Ovidiu Maxiniuc

On 06/19/2015 10:04 PM, Constantin Asofiei wrote:

> Eric/Ovidiu,
>
> This is a new discovery. And it looks like is not just character to int, int can be converted to date, logical, decimal, and vice-versa.
>
> We should test all combinations of data types and see how they are accepted, for all modes (input,output,input-output)... once we have a matrix mapping the data-type relations, we can add constructors in each BDT sub-class (ControlFlowOps$InternalEntryCaller.valid:5218 checks if there is a constructor with the argument's type as parameter, if types do not match).
>
> Also, functions have this "property" too, if invoked via DYNAMIC-FUNCTION.

#3 - 06/19/2015 03:41 PM - Ovidiu Maxiniuc

I continued by digging into procedure call mechanism. In ControlFlowOps, there is InternalEntryCaller.validate() that attempts to replace the argument that does not fit with a proper one. As I see the code (block starting at 5257), we are only do this one way. I believe this should be done to support all three types of parameters: INPUT / OUTPUT and mixed.

- The current code only support the INPUT by searching to convert the current value to match the destination. D(S)
- It happened that I encountered here the other direction. So the matching should be the other way around, and no need to have a direct conversion. There is no a 'current value' for this kind of parameters, so the default (unknown) will suffice. S(D) and D().
- The INPUT-OUTPUT must verify that the value can be transported both way. There are two possibilities, we need to investigate how Progress does:
  - either accepts double conversion S(D) & D(S)
  - either does not accept conversion at all, so the types must match.

#4 - 06/19/2015 03:51 PM - Eric Faulhaber

- Project changed from Database to Base Language

#5 - 06/24/2015 10:06 AM - Ovidiu Maxiniuc

- Assignee set to Ovidiu Maxiniuc

I started the implementation for the optional conversions in message passing parameters. The forward checking is done. But I have some difficulties when setting back the values when the called procedure ends.

At this moment we have the FieldAssigner that is called from IntegerField constructor (for example) to wrap over our field using an intermediary variable. When the called procedure ends, the FieldAssigner should assign the variable used as OUTPUT parameter back to original field. However, the variable and the field have the same type, always. This is incorrect because the procedure we are calling could have a different type of argument and the conversion is necessary.

I think that FieldAssigner is not useful here. It will do the assign back when the procedure ends. The problem is, the wrapper it will try to assign was dropped when the ControlFlowOps.java checked the calling parameters and replaced with the a new value of the appropriate type (a character in this case). To fix the issue we need to reset the newly created wrapper back to IntegerField.

As a last resort alternative is to manually set the values in ControlFlowOps.invokeImpl() after the Method.invoke() returns and the args contain the

computed OUTPUT values. Anyway, I think this might not be possible as the original field is not available (only the wrapper from FieldAssigner).

#6 - 06/24/2015 10:47 AM - Ovidiu Maxiniuc

I have created branch 2592 from the main trunk (revno 10886). I committed the intermediary changes in ControlFlowOps to branch revision 10887.

#7 - 06/24/2015 11:55 AM - Ovidiu Maxiniuc

After further thinking:
what if we allow the internal variable from FieldAssigner to be updated. For example, if the IntegerField initially used an integer to wrap the field and the called procedure requires an OUTPUT character, we need to update the integer variable from FieldAssigner to a character. This happens in ControlFlowOps when the parameters are checked.

The problem is that in the ControlFlowOps, InternalEntryCaller.validate() we only have the wrapper, which has no reference to original field, FieldAssigner or IntegerField. The solution would be to have a context map in FieldAssigner and search it based on variable to be updated as key. If found, it should be rekeyed on the new variable (of the correct type) or even discarded because at most one of this operation can be realized.

Do you have any other maybe better ideas ?

#8 - 06/24/2015 12:30 PM - Ovidiu Maxiniuc

Looks like my idea from note 7 can be easier implemented thanks to the List<FieldAssigner> from FieldAssigner context.
Sorry for these incremental status updates.

#9 - 06/24/2015 12:40 PM - Greg Shah

> Sorry for these incremental status updates.

You have done exactly the right thing in posting these different ideas and thoughts. It is very valuable to those you will follow behind and work on this area in the future. We can see the options you considered and why you chose the final approach. Thank you for putting this all in the task.

#10 - 06/24/2015 01:32 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> Looks like my idea from note 7 can be easier implemented thanks to the List<FieldAssigner> from FieldAssigner context.

I am having trouble following your idea. If it is simple enough to implement, please update the branch and I'll review. Otherwise, can you put together a bit of pseudocode, so I can understand the proposed change better?

#11 - 06/24/2015 03:01 PM - Ovidiu Maxiniuc

The branch has been updated to revision 10888. I already tested with ETF and it passed all +JULIAN_DATE tests that were listed in #1868.
I believe this is a viable candidate. Please review.

#12 - 06/24/2015 03:29 PM - Eric Faulhaber

Code review 10888:

This looks good to me, with 2 very minor points:

- Please rename FieldAssigner$Scope.list to something more representative of its new role, since it is no longer a List.
- You've given me credit for your work in the file header of ControlFlowOps: (ECF->OM).

Greg or Constantin: as our base language experts, could one of you please review this as well?

#13 - 06/24/2015 03:48 PM - Ovidiu Maxiniuc

- Status changed from New to WIP

New changes committed to revision 10889.

#14 - 06/24/2015 03:53 PM - Eric Faulhaber

10889 looks good, thanks, but I still want Greg or Constantin's stamp of approval, particularly on the ControlFlowOps changes.

#15 - 06/24/2015 05:04 PM - Greg Shah

- Status changed from WIP to New

Code Review Task Branch 2592 Revision 10889

The core functionality seems good.

1. In FieldAssigner.assign() you should use logging instead of e.printStackTrace().

2. Should we log instead of raising an exception in FieldAssigner.addAssigner()?

3. In ControlFlowOps, you removed a cast of null to handle on line 1055. Although this casts may be strictly unnecessary, the fact that we have so many variants of invokeImpl() makes the cast useful because any changes to the signature that cause breakage will be easily identified. I think it is best to leave it there.

Constantin: as the owner of ControlFlowOps, do you have any concerns with the update?

#16 - 06/25/2015 04:26 AM - Constantin Asofiei

Greg Shah wrote:

> Constantin: as the owner of ControlFlowOps, do you have any concerns with the update?

The logic looks OK.

Ovidiu, some questions:

- is the TODO at line 5280 still needed?
- are the existing c'tors in the BDT subclasses enough to cover all the cases of automatic/implicit parameter casting in 4GL?

#17 - 06/25/2015 10:10 AM - Ovidiu Maxiniuc

- Status changed from New to WIP

I updated the code according your reviews. The new revno is 10890.

Constantin: your 2nd question is very good. I need to create a testcase with full 3D matrix of conversions to be sure of that. We should be aware that P4GL might not handle all combinations (ex: date - logical (?)) and display errors in these cases.

There is something else I discovered while was testing with new update: with it, we do support various datatype conversions (I am referring now only to OUTPUT params) in the case that the arguments are fields. But we don't support normal variables!
If in the testacase from note 1 the second parameter is a normal variable, the converted code will just use its java counterpart when calling the procedure. When checking the datatypes, InternalEntryCaller.validate() will attempt to replace it with a correct type. This will fail because there is no mechanism like the FieldAssigner in this case (FieldAssigner$Scope:WARNING) FieldAssigner not registered, OUTPUT value may not be converted to correct type at return time. will be printed into console). As result, when the calling procedure ends, the OUTPUT variable remains unchanged!

#18 - 06/25/2015 11:31 AM - Greg Shah

These are very good points (the complete matrix and the support for variables). I think both of these issues need to be addressed before we check this code in. Otherwise, we will forever be chasing cases of this down which will cost much more time later than it would to just resolve it now.

#19 - 06/25/2015 11:50 AM - Ovidiu Maxiniuc

Support for variables.
I am thinking of the following changes in code:

- an abstract class Assigner with BaseDataType variable and Constructor<?> convCtor as fields and assign() abstract, with much of the implementation of current FieldAssigner
- FieldAssigner extends Assigner and implements assign() method as it is in revno 10890.
- VariableAssigner also extends Assigner with a similar assign() written for variables.
- the Scope.updateAssigner will automatically add variable to internal data structure if search fails (meaning that the call to procedure was done by hardcoding the variable so the Assigner is not aware of it).

The matrix should be effectively hand-written for following types of parameters:

- formal parameter types: int/int64/decimal, etc..
- actual values types: int/int64/decimal, etc..
- parameter passing types: INPUT / OUTPUT / INPUT-OUTPUT
- types of arguments: literals / variables / fields
  Looks fairy complicated, a 4-dimensional matrix.

#20 - 06/25/2015 12:25 PM - Greg Shah

Very good plan and design.

#21 - 06/26/2015 12:53 PM - Ovidiu Maxiniuc

The branch 2592 has been updated and rebased (main trunk revision 10887). It is not at revision 10892.
It passed my quick testcases, was not checked rigorously yet.

It contains a 1st round of OUTPUT variable parameter support. Both VariableAssigner and ControlFlowOps extends the OutputParameterAssigner abstract class. The algorithm is simple: ControlFlowOps attempts to update a variable. If an OutputParameterAssigner is found, the variable is updated, otherwise it is assumed that on that position there is a normal variable that doesn't have a FieldAssigner already set up so a VariableAssigner is created and configured at runtime.

There is one aspect I am not happy about the code: calling the OutputParameterAssigner.update(). Since the base class is abstract, the method must be accessed via one of the implementation, so I guess there is no cleaner solution.

#22 - 06/26/2015 04:03 PM - Ovidiu Maxiniuc

The testcases project was updated with uast/procedures/conversion-params.p. It contains the matrix for conversion parameters calls. I did not run it against P4GL.

#23 - 06/29/2015 02:48 PM - Ovidiu Maxiniuc

After running the testcase procedure against P4GL, I was able to detect the cases where the automatic/implicit conversion is required and where the errors should be thrown. The uast/procedures/conversion-params.p is revision 1286 will analyze the full matrix and:

- if the conversion c'tor fails to convert correctly between datatypes, will print the respective messages. Ideally should be none (as P4GL) does.
- if an extra constructor exists and somehow manages to do the conversion, an internal error number is displayed (00 - 80). The cases that contains the NO-ERROR clause, are expected to fail so they are guarded. If error is not present, a warning message is printed.

To answer Constantin's 2nd question from note 16, yes, we have all conversion c'tors we need, we have some extra so there are 16 cases when we should fail but we don't. I have the list of these and I'll handle them firstly.

Other issues I could found and investigate:

1. the datetime(-tz) to/from character are not working very well. From this POV, more informations on #2590. However, the assignments take another route, so a worker method might be necessary to handle both cases (#2590 uses the dynamic form: datetime(BaseDataType value)), the current patch for this issue, on the other hand, does the best effort to find the best conversion constructor, and will do this too well. It will prefer the more specific datetime(character spec).
2. the decimal methods for converting from String / character are not working correctly with European number formats (-E startup parameter, as configured on lincon01). There are two cases that must be handles, exactly as when working with datetime:
   - a literal taken from P4GL source code will always use the American format (something like: decimal.fromLiteral("1.52"))
   - a string that is converted at runtime will use the currently configured number format. From this point of view, the testcases will fail and will display 152,0000 instead of 1,5200 when runtime-converting 1.52 (the actual code: new decimal("1.52") in American runtime to be the same value as new decimal("1,52") in European).
     Date, character and integer values (expecting 2 but P2J will print 152) are a consequence of this fact. This should be probably deferred to another task.
3. There is a strange behavior in the case of character to logical conversion when the parameter is passed in OUTPUT mode. For example, if the OUTPUT parameter is assigned any of the following strings: 'yes', 'no', 'true' or 'false' regardless of case sensitivity, the call fill fail! However if the parameter is passed in INPUT-OUTPUT mode, P4GL returns successfully. P2J will ALWAYS return successfully. We should emulate this Progress BUG.

#24 - 06/29/2015 04:18 PM - Eric Faulhaber

As I was reviewing this code, I was wondering why VariableAssigner and OutputParameterAssigner were not in the com.goldencode.p2j.util package, since they do not appear to be specific to persistence. Then I recalled Hynek did some related work for extent fields not too long ago, but I wasn't directly involved. Please look at his InputOutputExtentParameter and AbstractExtentParameter classes. There seems to be overlap of purpose. Can we rationalize all this into a single hierarchy?

#25 - 06/29/2015 04:24 PM - Eric Faulhaber

The issue for Hynek's earlier work is #2293.

#26 - 07/08/2015 11:07 AM - Eric Faulhaber

Ovidiu wrote (in email):

> There are two main issues I would like to discuss and I would appreciate additional help:
>
> - at this moment we are not handling the implicit conversion at all in the case of the extent parameters. To be honest, I don't know if this is possible at this moment, using the informations available at runtime. How do we check if the arrays passed as parameters have the correct element types? We are able to see the passed values but the formal parameters are not available because of the java type erasure (the 'destination' types are dropped after the compiler is certain they match).
> - the second issue is related to array dimension. If the passed array does not have the expected size, Progress will drop the procedure execution. My problem is that we check this in initParameter() at the time a local variable is instantiated. I could not find a solution for this because this operation (Ex: integer[] p = extp.initParameter(2);) is performed in the block constructor so the BlockManager has not built yet the framework for error handling so any calls on ErrorManager will reflect on the caller procedure instead. I am not sure I made myself clear about this second issue (see case 5 of note 10 from #2293).

I spoke with Greg about this. It seems that these are two facets of the same problem. Please confirm that the function/procedure is never executed in these cases of mismatched element types and mismatched extent lengths. This would make sense, since it would be an unusual design to validate parameters once a function or procedure already is executing.

Assuming you confirm this is the case, the parameter validation will have to be pushed to ControlFlowOps, alongside all the other parameter validation that goes on there before a function or procedure is invoked. Extent length and type information for parameters already should be available via SourceNameMapper and its related runtime classes. Once validation is done, you can assume that any work done within the AbstractParameter hierarchy is using the correct types (and lengths, in the case of extent parameters).

#27 - 07/08/2015 11:16 AM - Eric Faulhaber

I read through ControlFlowOps a bit more, and I must be misunderstanding your second point above, because it appears the extent length validation already is happening (line 4551 in 2592/10903). Why does this need to be done again in InputOutputExtentParameter.initParameter?

#28 - 07/08/2015 01:07 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> I read through ControlFlowOps a bit more, and I must be misunderstanding your second point above, because it appears the extent length validation already is happening (line 4551 in 2592/10903). Why does this need to be done again in InputOutputExtentParameter.initParameter?

The validArguments() calls caller.valid() to detect if there are issues with the current arguments. The valid() does not check the size of the extent arguments. Probably it did some time ago (line 5137 in 2592/10903) when probably the EXTENTS were passes as java arrays (if this ever happen). If so, at this moment, that code is not useful any more because the EXTENTS are sent wrapped in an [Input-]OutputExtentParameter which is not an array, so the .isArray() will return false and the detection fails. Today, I am not aware of any other types of data passed to procedures as java array.
I guess that if the size is handles in ControlFlowOps, there is no need for the additional checking in InputOutputExtentParameter.initParameter.

#29 - 07/08/2015 02:18 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> I guess that if the size is handles in ControlFlowOps, there is no need for the additional checking in InputOutputExtentParameter.initParameter.

Yes, if in Progress the function is not executed as a result of an extent size mismatch error, then we have to handle it in ControlFlowOps, such that we don't invoke the converted version of the function.

I don't know the history of whether these were passed as Java arrays before (probably -- you may want to look at the VCS history for this section of code), but it seems that the code which is bypassed because of the isArray() check needs to be revisited.

Can anyone comment who was directly involved in writing (or subsequently changing) this ControlFlowOps parameter checking code?

#30 - 07/09/2015 10:15 AM - Ovidiu Maxiniuc

It looks to me that line 4551 in 2592/10903 will only check for types errors: both argument and formal parameter to be arrays or not.
In the case they are both some kind of arrays (java or AbstractExtentParameter) is not handed at this level. I am adding the code now.

Another thing: I believe the ArgValidationErrors enum is broken (or at least its usage). Here is why:

- being an enum, we have only 10 constant (?) instances, no more, no less,
- the class also has two fields: expectedExtent and paramExtent this is already a 1st break to constant-ness of the enum, but if the enum is used locally and not long lived this should not be an issue,
- the enum values are cached in long-living structure, inside a CacheValue. This is bad.

Suppose we have two procedures that are processed. The first one is processed and return value is cached with one item from ArgValidationErrors it is eventually configured with the appropriate values for the fields according to respective call. Then the second procedure is called. When the cache value is created, it uses the same ArgValidationErrors 'const item' but with different values for the fields. Since the rv is really the same object as in first call, the fields will be overwritten.

I am going to fix this issue by using a normal class with a type field.

#31 - 07/09/2015 11:37 AM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> ...
> Since the rv is really the same object as in first call, the fields will be overwritten.

Yes, good catch. I missed this when I added the caching recently.

> I am going to fix this issue by using a normal class with a type field.

I'm not sure I understand from this description what your new implementation will look like, but that's OK, I'll wait for the review. I'll just note that it is important to preserve as much as we can of the caching, since this has helped with performance. Obviously, you are right that the specific, expected and actual extent values cannot be cached, so this portion of the validation check would have to be outside the caching.

I added the caching to avoid the cost of resolving (including the failure to resolve) the target method for every unique combination of calling class, method name, signature, and "function-ness" (the function boolean argument to InternalEntryCaller.valid). Apparently, I was too aggressive in caching results that relied upon the arguments themselves, which could vary from call to call. Please review the caching implementation with that in mind, as there may be other aspects of the caching that need to be fixed.

#32 - 07/10/2015 09:35 AM - Ovidiu Maxiniuc

The 2592 branch was rebased with 10895 main trunk and now it is at revno 10906.

#33 - 07/10/2015 12:32 PM - Ovidiu Maxiniuc

Hynek,

I found in one of your testcase files the following test:

```
def temp-table tt1
   [...]
   field f16 as decimal decimals 3 extent 2
   [...]
```

```
function foo12 returns int extent (input-output p as decimal extent).
  if extent(p) <> 2 then message "ERR (foo12) p <> 2".
  p[1] = 1.5555555555.
  p[2] = 1.5555555555.
  if p[1] <> 1.5555555555 then message "ERR (foo12) p[1] <> 1.5555555555".
  if p[2] <> 1.5555555555 then message "ERR (foo12) p[2] <> 1.5555555555".
end.
```

```
f16[1] = 1.5555555555.
if f16[1] <> 1.556 then message "ERR f16[1] <> 1.556".
```

```
    if f16[2] <> 0 then message "ERR f16[2] <> 0".
    foo12(input-output f16).
    if f16[1] <> 1.5555555555 then message "ERR f16[1] <> 1.5555555555".
    if f16[2] <> 1.5555555555 then message "ERR f16[2] <> 1.5555555555".
    f16[1] = 1.5555555555.
    if f16[1] <> 1.556 then message "ERR f16[1] <> 1.556".
    if f16[2] <> 1.5555555555 then message "ERR f16[2] <> 1.5555555555".
```

It works perfectly on P4GL, no message is printed. However, P2J (with my changes) is not perfect, it will print the following 3 messages:

```
    ERR f16[1] <> 1.5555555555
    ERR f16[2] <> 1.5555555555
    ERR f16[2] <> 1.5555555555
```

I analyzed the cause for these and it seems very strange behavior.

- In Progress, when the decimal extent field is assigned back from foo12, it looses temporarily the designated precision (3) and the default (10) is used instead. But as soon the f16[1] field is assigned back to a value, it will regain the declared precision.The f16[2] is not reassigned so it will keep the precision of 10.
- In P2J, the values are processed with standard precision (10) inside function and when assigning back, the precision of the original field remain unchanged (3), so the above errors will be printed.

I don't know how you got such testcase but it seems to me like a nasty behavior to mimic.

I could not spot any handling in the code nor special comment on #2293 thread. I believe your tests also the same error messages. What was the solution to this? Was this quirk deferred to another task or delayed?

#34 - 07/10/2015 12:33 PM - Hynek Cihlar

Ovidiu,

it's been a while since I was playing with extents.

The case you refer to shows that 4GL doesn't propagate precision to a function argument, instead it uses the precision declared on the argument which is default in this case. Also the precision is not adjusted when the input-output (or output) argument value is assigned back to the field/variable at the function call.

The implementation I introduced was handling all the cases in the file you mention. Or did you find otherwise?

I am afraid we will have to mimic this however it seems ilogical. If I can be of any help to implement this, please let me know.

Thanks,
Hynek

#35 - 07/10/2015 12:40 PM - Eric Faulhaber

Ovidiu, please try this testcase without your changes applied and track back to where it broke (which may be part of your changes, or may be older).

#36 - 07/10/2015 12:42 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Ovidiu, please try this testcase without your changes applied and track back to where it broke (which may be part of your changes, or may be older).

Yes, I will do that to be sure if this is new regression.

#37 - 07/10/2015 01:25 PM - Ovidiu Maxiniuc

It seems like this case was somehow overlooked, with revno 10895 of main trunk it displays:

```
ERR f16[1] <> 1.5555555555
ERR f16[2] <> 1.5555555555
ERR f16[2] <> 1.5555555555
Procedure complete. Press space bar to continue.
```

Hynek Cihlar wrote:

> The case you refer to shows that 4GL doesn't propagate precision to a function argument, instead it uses the precision declared on the argument which is default in this case. Also the precision is not adjusted when the input-output (or output) argument value is assigned back to the field/variable at the function call.
> The implementation I introduced was handling all the cases in the file you mention. Or did you find otherwise?

Well, you might have missed it. At this moment this testcase fails as you can see above. Looking into decimal.assign(NumberType, boolean) that the BufferRecord calls in 10695, there is no action to copy the precision. By contrary, the comment in decimal-to-decimal case states that the new values to be saved is truncated *as set by our precision*.

> I am afraid we will have to mimic this however it seems ilogical. If I can be of any help to implement this, please let me know.

The problem is that we need to temporarily overwrite the precision of the database field until the field is reassigned. This means we need to temporarily save the old/declared precision (somewhere) and restore it back when the field is assign. Much worse is that I am not able to tell which is one assignment and which is the other one.

#38 - 07/10/2015 01:49 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> The problem is that we need to temporarily overwrite the precision of the database field until the field is reassigned. This means we need to temporarily save the old/declared precision (somewhere) and restore it back when the field is assign.

Yes, the DMOs are designed to disallow setting precision through the setter method calls. The private field is mutable, but only the value.

> Much worse is that I am not able to tell which is one assignment and which is the other one.

What do you mean? In what code are you unable to make the distinction?

#39 - 07/10/2015 02:00 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> It seems like this case was somehow overlooked, with revno 10895 of main trunk it displays:
> [...]

It may already be obvious to you from reviewing the code whether this was missed initially or regressed later, in which case you can ignore this suggestion. If not, you should try it also with rev. 10577. If that is not working, then we know it was just an oversight and we can determine how/when we want to implement this "feature". But if it works, then we know it was regressed some time after that, and we should be able to pinpoint the regression and fix it.

#40 - 07/10/2015 02:03 PM - Hynek Cihlar

Sorry I didn't realize this earlier, the output parameter precision for fields is covered by #2295.

#41 - 07/10/2015 02:08 PM - Eric Faulhaber

Sorry I didn't make this connection sooner. OK, so we've never implemented a solution. Ovidiu, please move forward with the remaining aspects of this issue and get the remaining test cases (which are not specific to this quirk) working. We'll handle that issue separately.

#42 - 07/13/2015 12:16 AM - Eric Faulhaber

Ovidiu, I did a quick review of 2592/10907 (though you have not asked for it yet). It looks as though you're nearly finished, aside from the work now

deferred to #2295. Please document what is still left to do. Some other comments/questions follow...

In various methods in BlockManager, you added a calee variable. Please change this to callee.

I'm not sure I properly understand the idea behind the new method decimal.fromLiteral. Is the assumption that all decimal literals are in American number format based on the way the P2J conversion emits these in the converted code, or is this assumption based on the literals as found in Progress code?

The database-related changes look ok to me (as do the rest of the changes). However, the changes are extensive, and most of this update is about base language and control flow, in code I am less familiar with (especially the control flow ops). So, I'd feel more comfortable with a second review by one of our team members more expert in these areas (when you're ready with a release candidate).

#43 - 07/13/2015 07:15 AM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Ovidiu, I did a quick review of 2592/10907 (though you have not asked for it yet). It looks as though you're nearly finished, aside from the work now deferred to #2295. Please document what is still left to do. Some other comments/questions follow...

Thanks Eric. To be honest, I am not completely pleased with the class hierarchy I merged because it does not respect the OOP principles. Related to this hierarchy, please let me know if you agree to move the FieldAssigner, OutputExtentField and InputOutputExtentField from com.goldencode.p2j.persist.type to com.goldencode.p2j.util where I think it's a better place for them.
I am finishing today the implicit conversion for extent parameters and then advance to automated testing stage while waiting for the review.

> In various methods in BlockManager, you added a calee variable. Please change this to callee.

Thanks for spotting this I fixed both names of the local variables.

> I'm not sure I properly understand the idea behind the new method decimal.fromLiteral. Is the assumption that all decimal literals are in American number format based on the way the P2J conversion emits these in the converted code, or is this assumption based on the literals as found in Progress code?

Progress literals always use '.' as decimal separator (American number format). The P4GL decimal(char) function will use the active number format, which for the customer is ',' because they are using European format. P2J was using the the conversion constructors decimal(String/character) for both cases: when parsing a decimal literal from P4GL source files and when doing the conversions (both implicit and explicit). For second case it was OK, but decimal literals values were seriously wrong (one example is documented in javadoc, other tescase: run message 1.222 < 2 on a box configured with European number format: P4GL will display yes but P2J will display no because the left operand is emitted in java source as decimal("1.222") which will be parsed at runtime using European syntax and evaluated to 1222).

> The database-related changes look ok to me (as do the rest of the changes). However, the changes are extensive, and most of this update is about base language and control flow, in code I am less familiar with (especially the control flow ops). So, I'd feel more comfortable with a second review by one of our team members more expert in these areas (when you're ready with a release candidate).

Agreed, after final commit, I'll wait for Greg's or Constantin's review.

#44 - 07/13/2015 02:56 PM - Ovidiu Maxiniuc

I have found a new issue when calling a procedure with a single EXTENT parameter. Consider a procedure

```
PROCEDURE p1:
   DEFINE INPUT PARAMETER q AS CHARACTER EXTENT 2.
   MESSAGE "OK".
END.
```

Suppose we want to call this procedure with an extent field as argument: RUN p1(INPUT tt1.ext2c).
The following code will be generated:

```
ControlFlowOps.invokeWithMode("p1", "I", tt1.getExt2c());
```

Because the syntax from ControlFlowOps is

```
invokeWithMode(String    name, String    modes,  Object... args)
```

the table field array will be matched with args parameter array, so the will be understood as an array of two character -s. As result the p1 procedure will not be matched (unless there is a duplicate with same name and two parameters) and the call will fail.

The solution is to let the compiler know that the last parameter of invokeWithMode() is a single object, and not an array. In consequence we should generated the following code:

```
ControlFlowOps.invokeWithMode("p1", "I", (Object) tt1.getExt2c());
```

The simple cast to Object seems to be enough to fix the issue. In the case where the procedure has more parameters, there are no issues.

The actual implementation requires checking the node type in convert/control_flow.rules:657 and if the node is a field, check for extent annotation locally, otherwise the refid for normal variables. This fix will be available in the next commit on the task branch.

#45 - 07/14/2015 12:12 PM - Eric Faulhaber

Code review 2592/10908

I only looked at the diffs from 10907 to 10908.

The conversion change in control_flow.rules looks fine.

The meld utility detected massive changes in ControlFlowOps betwee lines 5088 and 5465, but it appears this is largely due to a change in indent and that the actual changes are much smaller:

- a change in the initial test at line 5088 to detect the case which requires us to skip the processing and continue the loop immediately;
- some additional comments.
  Is there any other significant, functional change in this area I overlooked?

The rest of the changes look good, though I guess this is not a final pass? For example, I'm thinking of the TODO at OutputExtentParameter:152.

#46 - 07/14/2015 12:16 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> ... To be honest, I am not completely pleased with the class hierarchy I merged because it does not respect the OOP principles. Related to this hierarchy, please let me know if you agree to move the FieldAssigner, OutputExtentField and InputOutputExtentField from com.goldencode.p2j.persist.type to com.goldencode.p2j.util where I think it's a better place for them.

Is the cross-package inheritance your only concern, or is there some other OOP violation?

I'm reluctantly OK with moving these classes to util. I say, "reluctantly", because that package already is so bloated with every manner of feature and needs to be refactored. However, I'm OK with it because so much of the related functionality needed for the parameter support already is there.

#47 - 07/14/2015 01:56 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Code review 2592/10908
>
> I only looked at the diffs from 10907 to 10908.
>
> The conversion change in control_flow.rules looks fine.
>
> The meld utility detected massive changes in ControlFlowOps betwee lines 5088 and 5465, but it appears this is largely due to a change in indent and that the actual changes are much smaller:
>
> - a change in the initial test at line 5088 to detect the case which requires us to skip the processing and continue the loop immediately;
> - some additional comments.
>   Is there any other significant, functional change in this area I overlooked?

The new part is in else if (AEP.isAssignableFrom(expectedType) && AEP.isAssignableFrom(argType)) block, where the datatypes of the elements from AbstractExtentParameters. Please temporarily enable the 'All whitespace' text filter in meld preferences for a better view of the changes.

> The rest of the changes look good, though I guess this is not a final pass? For example, I'm thinking of the TODO at OutputExtentParameter:152.

That is an issue/question I could not give an exact answer. In P4GL there are testcases where the procedure stop when such condition is encountered and other are just going even though they are not in a NO-ERROR bracket. The only certain thing is that the error message is displayed in either cases. Must be another quirk here, maybe related to other reasons.

#48 - 07/14/2015 02:07 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> The new part is in else if (AEP.isAssignableFrom(expectedType) && AEP.isAssignableFrom(argType)) block, where the datatypes of the elements from AbstractExtentParameters. Please temporarily enable the 'All whitespace' text filter in meld preferences for a better view of the changes.

Thanks, that helped. I think the changes look OK, but it would be easy to miss a detail in this complex bit of code. It is good that you are testing thoroughly.

#49 - 07/14/2015 02:18 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Ovidiu Maxiniuc wrote:
>
> > ... To be honest, I am not completely pleased with the class hierarchy I merged because it does not respect the OOP principles. Related to this hierarchy, please let me know if you agree to move the FieldAssigner, OutputExtentField and InputOutputExtentField from com.goldencode.p2j.persist.type to com.goldencode.p2j.util where I think it's a better place for them.
>
> Is the cross-package inheritance your only concern, or is there some other OOP violation?

No, there are 2 things here:

- OOP issue: in AbstractParameter.java, the context local Scope is aware of the extending classes AbstractSimpleParameter and AbstractExtentParameter. A good OOP design should have handled them transparently. I tried to do that but the code tended to become too

complicated so I preferred the simpler approach.
- moving Field-related classes to p2j.util package.

  I'm reluctantly OK with moving these classes to util. I say, "reluctantly", because that package already is so bloated with every manner of feature and needs to be refactored. However, I'm OK with it because so much of the related functionality needed for the parameter support already is there.

There is another reason for my question: once those classed are moved to new package the conversion is also affected. The imports must reflect the new packages so the generated inner classes (that wraps extent field passed as parameters) be able to find their super classes.

#50 - 07/14/2015 02:44 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> There is another reason for my question: once those classed are moved to new package the conversion is also affected. The imports must reflect the new packages so the generated inner classes (that wraps extent field passed as parameters) be able to find their super classes.

That's a good point. If you see that as being more than an hour or two of work, I would prefer to defer that work to the Other Runtime Improvements milestone, given that the scope of this issue already has become much larger than anticipated.

#51 - 07/15/2015 05:31 AM - Ovidiu Maxiniuc

The revno 10909/2592 test ended with some failures:

- 3 tests in CTRL+C (known to fail in most runs)
- gso_269/48 in gso_tests (known to fail occasionally)
- tc_codes_employees_021/22 (known to fail occasionally)
  tc_dc_slot_026/8, tc_job_clock_004/35 (time related).

This revision does not contain the shift of the *Field classes to new package.

#52 - 07/15/2015 04:05 PM - Ovidiu Maxiniuc

I updated the branch. What is new in revno 10910:

- moved FieldAssigner, OutputExtentField and InputOutputExtentField to com.goldencode.p2j.util package - that was easier than expected, that package is mandatory because it holds classes that emulates the P4GL types;
- added second detection of conversion constructor in the case the exact-type conversion constructor does not exist;
- cleanup and updated javadocs in multiple files.

I also did further investigation about the TODOs left behind. Particularly the one in OutputExtentParameter. It was not related to the fact that the

extant filed is static or dynamic as I first thought but it depends on the type of the subroutine that has been called:

- if a function was called, then just display the error message and keep running
- if a procedure was called then that is a serious error and the condition will stop execution (unless launched in NO-ERROR mode, but the message is still printed)
  There are two solution I can see: add an extra parameter (like I did for routine name) or inform the OutputExtentParameter from the beginning whether it is used in a function or procedure and store the flag locally until validation time. I would incline for the first one.

#53 - 07/15/2015 11:11 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> - if a procedure was called then that is a serious error and the condition will stop execution (unless launched in NO-ERROR mode, but the message is still printed)
>   There are two solution I can see: add an extra parameter (like I did for routine name) or inform the OutputExtentParameter from the beginning whether it is used in a function or procedure and store the flag locally until validation time. I would incline for the first one.

This information is available as an InternalEntry$Type value (via SourceNameMapper). Can't you get it from there? It shouldn't be too performance-sensitive, since this is just for the error case, right?

#54 - 07/17/2015 05:06 AM - Paul E

I see this is marked as blocking #1868.

Can you give me a couple of examples of where we are relying on this implicit type coercion in the application code, and perhaps point out a test that is failing because of it?

#55 - 07/17/2015 05:36 AM - Ovidiu Maxiniuc

Paul Eames wrote:

> I see this is marked as blocking #1868.
>
> Can you give me a couple of examples of where we are relying on this implicit type coercion in the application code, and perhaps point out a test that is failing because of it?

The example from 1st note is from actual code taken from the application server code, isolated and anonymized. It is the test [Entity=|+JULIAN_DATE| TestType=|SEARCH| TestSubType=|SEARCHEABLE_ATTRIBUTE| xmlSuffix=|10|] from ETF that provided the testcase. However, the following tests in sequence: 11.. 23 also are fixed by applying the update from this task.
The exact code you can find in common/ajulnas0.p in PROCEDURE PRowCreate_Date that calls ra/ebacsev0.p.

#56 - 07/17/2015 06:05 AM - Constantin Asofiei

Ovidiu, about revision 10912:

- SourceNameMapper.getParameter:941 - you should check for param >= ie.getParameterListSize(), as param is 0-indexed, not 1-indexed. Fix it in InternalEntry.getParameter, too, please.
- ControlFlowOps:

```
                    // TODO: finish all cases of extent parameter validation
                    //       OM: which cases?
```

I think this TODO can be removed.

```
// TODO: check the OUTPUT mode case
```

What is the state of these TODOs? Do you plan to finish them?

#57 - 07/17/2015 01:57 PM - Ovidiu Maxiniuc

I was going further and investigating some other failed tests from #1868. I noticed that in P2J some query was failing because of an explicit conversion. Isolated it looks like this:

```
FOR FIRST TTFilterCriteria WHERE LOGICAL(TTFilterCriteria.FieldValue):
   ASSIGN VlincludeConsolidated = FALSE.
END.
```

The FieldValue was "VOID2". In Progress this is not a error condition, the VlincludeConsolidated remained true (as it was initialized). I continued my investigations and I saw that:

```
VlincludeConsolidated = logical(TTFilterCriteria.FieldValue).
```

it's also not an error, for same value of FieldValue. Clearly, the returned result of the explicit conversion was ? (unknown). Then I went back to initial query and adjusted like this:

```
FOR FIRST TTFilterCriteria WHERE LOGICAL(TTFilterCriteria.FieldValue) eq ?:
   ASSIGN VlincludeConsolidated = FALSE.
END.
```

In this case the query predicate is true so the variable is assigned. The message complaining about bad value (** Input value: VOID2 should be yes/no. (87)) was printed, but not as an error.
With these results, it was clear for me that we are not handling this correctly in conversion constructor logical(String, String), the

```
ErrorManager.recordOrThrowError(87, errmsg, true);
```

should be replaced with

```
    ErrorManager.displayError(87, errmsg, true);
```

Then I re-ran the implicit conversion matrix test I prepared for #2592. I got a little surprise: 4 of the testcases were failing (OUTPUT and INPUT-OUTPUT parameters of LOGICAL type).

Now, I think I understand how we should handle this: if the code uses an **explicit** conversion (using logical function), P4GL will eventually return ? (unknown), but will never fail. If it is the case of an **implicit** conversion - this is a error condition and the procedure is abend (well, unless in no-error mode).

I did not studied yet the other combinations of data types but I suppose the behavior is similar.

#58 - 07/17/2015 02:07 PM - Greg Shah

Aren't all implicit cases using the BaseDataType constructor?

#59 - 07/17/2015 02:22 PM - Ovidiu Maxiniuc

Greg Shah wrote:

> Aren't all implicit cases using the BaseDataType constructor?

At this point, no. The ControlFlowOps uses the reflection to find the proper conversion constructor. If none is found then the data types must be incompatible and the specific error condition is emitted before the actual call to called procedure or any constructor.

Using the BaseDataType could be a solution, but we need to store somewhere the compatibility matrix. I guess this would simplify the code in some degree, in place of using the slow reflection we can only check the compatibility and them let the BaseDataType constructor do the heavy working. The only drawback would be to choose the right method in the case of overloaded (with multiple signatures).

#60 - 07/17/2015 04:00 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Ovidiu, about revision 10912:
>
> - SourceNameMapper.getParameter:941 - you should check for param >= ie.getParameterListSize(), as param is 0-indexed, not 1-indexed. Fix it in InternalEntry.getParameter, too, please.

Nice catch. Thanks

- ControlFlowOps:
  [...]
  I think this TODO can be removed.

Ok.

> [...]
> What is the state of these TODOs? Do you plan to finish them?

I added this reminder because my senses told me those cases should also be handled. I am not sure that is possible (ex: it's logical to have an ? as input parameter, but is it possible to have an typeless unknown on output?), I was focused on some particular cases required by the task at hand.

The branch 2592 was rebased with main trunk (revno 10896) and now the above changes were also committed. The latest revision is 10915.

#61 - 07/20/2015 10:08 AM - Greg Shah

> Using the BaseDataType could be a solution, but we need to store somewhere the compatibility matrix. I guess this would simplify the code in some degree, in place of using the slow reflection we can only check the compatibility and them let the BaseDataType constructor do the heavy working.

The BDT constructors were originally intended to be the only constructor used for implicit conversions. And the idea is that we would not use the BDT constructors for other purposes, so it was safe to put logic in there that was specific to implicit conversions.

I still think that is the best approach. The other advantages you note are also nice.

> The only drawback would be to choose the right method in the case of overloaded (with multiple signatures).

You mean that we will have to force a cast to BDT to "pick" this constructor? That is OK.

#62 - 07/21/2015 01:10 PM - Eric Faulhaber

Code review 2592/10915:

The items from the previous code review appear to have been addressed.

Ovidiu, what is the status of this issue now? Has this level of the code passed regression, such that it can be merged? I'd like to get these fixes in.

#63 - 07/21/2015 01:43 PM - Ovidiu Maxiniuc

Eric Faulhaber wrote:

> Code review 2592/10915:
>
> The items from the previous code review appear to have been addressed.
>
> Ovidiu, what is the status of this issue now? Has this level of the code passed regression, such that it can be merged? I'd like to get these fixes in.

The last regression testing was performed on revno 10912. It has a single flaw, in tc_item_master_031, (Mismatched data at line 6, column 72. Expected 'Y' and found 'R'). I run this manually and the test passed.
Ever since, there were no changes in the logic of the code, only the javadoc & comments were added (well, and the range check spotted by Constantin but that is evidently a piece of code that is not covered by majic testing, in fact that piece of code only takes effect in the case of intentionally malicious written code). I think a new regression test is not required.

The actual revno 10915 is a version behind the main trunk and should be rebased. I expect no conflicts with new GUI code.

I also would like to commit this code to main trunk. It is stable for the moment but much larger that I expected to be so it could conflict with future branches. I would like to continue the rest of the work on this task (re-routing all the conversion via BDT-parameter constructors instead of using reflection for finding the more appropriate one) for a next iteration of the branch. I already thought about the implementation and I estimate no more than 1-2 days of work.

#64 - 07/21/2015 02:26 PM - Eric Faulhaber

Please go ahead and rebase. If there are no conflicts as expected, go ahead and merge into the trunk.

#65 - 07/21/2015 03:26 PM - Ovidiu Maxiniuc

The 2582 branch advanced to revno 10916 after it was rebased with trunk revno 10897. There were no conflicts.
Then it was merged to main trunk as revno 10898. Notification email was sent to team.

#66 - 07/23/2015 02:41 PM - Ovidiu Maxiniuc

Eric,

I found out why the server log is filled with Two AbstractOutputParameterAssigner configured with same wrapper variable warnings. This happens because I used a Map having BaseDataType as keys. This is a very bad option! We should never use BDT for keying/fast lookup anything using standard java collections. The BDT has the equals() method overridden so two different java objects (an integer and a character fro example) having both unknown flag set, will be equals from the POV of P4GL, so a Map cannot hold them both :(.

The 2582 branch contains a usage of such map, so any attempt to pass two unknown OUTPUT parameters to a procedure could result in invalid output values.
I am trying to fix this regression on 2582a. The trick is to use a fast associative collection that does not rely on equals method for comparing the keys.

#67 - 07/23/2015 03:08 PM - Greg Shah

> I used a Map having BaseDataType as keys. This is a very bad option! We should never use BDT for keying/fast lookup anything using standard java collections.

Please update the BDT javadoc to clearly state this fact. It wasn't in my mind at all and we definitely need to make this clear to future users. It should be stated (in **bold**) in the class-level javadoc and in the equals() method.

#68 - 07/23/2015 03:10 PM - Eric Faulhaber

Ovidiu Maxiniuc wrote:

> The BDT has the equals() method overridden so two different java objects (an integer and a character fro example) having both unknown flag set, will be equals from the POV of P4GL...

Also, the BDT classes are mutable, so even in the case of not being unknown, it is not a reliable key (unless using reference-based equality).

> I am trying to fix this regression on 2582a.

I guess you mean 2592a?

> The trick is to use a fast associative collection that does not rely on equals method for comparing the keys.

As in IdentityHashMap? This will use reference equality.

#69 - 07/23/2015 03:21 PM - Ovidiu Maxiniuc

Greg,
I will add the requested notes in javadocs.

Eric,
2592a it is, I mistyped.
Thanks. I am replacing the HashMap with IdentityHashMap. It is exactly what is needed here because the intermediary value itself is the key, not its changeable value.

#70 - 07/23/2015 04:17 PM - Ovidiu Maxiniuc

I did a quick test and the warning is gone with the small change. I committed the update on 2592 as revno 10900.

#71 - 07/23/2015 06:04 PM - Eric Faulhaber

Code review 2592a/10900.

The changes look good. Please regression test and merge into trunk as soon as possible.

#72 - 07/24/2015 12:27 PM - Ovidiu Maxiniuc

The 2592a/10900 passed the regression testing. The harness results were saved as /opt/secure/clients/timco/majic_test_results/2592a_10900_16cd2a2_20150724_om.zip.

#73 - 07/28/2015 08:03 AM - Ovidiu Maxiniuc

The 2592a/10900 was merged to trunk as revision 10901 and the notification email was sent to team.

#74 - 08/05/2015 11:47 PM - Eric Faulhaber

- Status changed from WIP to Closed

- % Done changed from 0 to 100

#75 - 11/16/2016 12:06 PM - Greg Shah

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