Base Language - Feature #1620

implement EXTENT parameter support

10/21/2012 09:27 AM - Greg Shah

Status:	Closed	**Start date:**	02/04/2013
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	8.00 hours
Target version:	Runtime Support for Server Features		
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to Base Language - Bug #2133: fix precision for decimal, dynamic-exte...	**Closed**	**01/21/2014**	**01/27/2014**
Related to Base Language - Feature #1621: implement dynamic/unspecified EXTEN...	**Closed**	**02/04/2013**	

History

#1 - 10/31/2012 01:44 PM - Greg Shah

- Target version set to Milestone 7

#2 - 02/04/2013 08:27 AM - Greg Shah

- Start date changed from 10/21/2012 to 02/04/2013

- Assignee set to Constantin Asofiei

Starting in Progress 10.x extent parameters are allowed in procedures and functions. Also, function return values can be defined as an array.

This means that extent variables (fields too?) can be passed to/returned from these blocks.

For details, see DEFINE PARAMETER, the new EXTENT statement (not the function), the FUNCTION statement and the section on parameter specifications.

This task is related to #1621.

#3 - 02/04/2013 01:40 PM - Constantin Asofiei

> Starting in Progress 10.x extent parameters are allowed in procedures and functions.

The solution is to use Java arrays for these kind of parameters; when using INPUT, OUTPUT and INPUT-OUTPUT mode, the same processing as for non-extent parameters can be used. In case the extent parameter is defined fixed-size, we can enforce this in Block.init (by emitting some assertion code which enforces the extent size to be the expected value). More, 4GL doesn't allow array elements to be passed to OUTPUT or INPUT-OUTPUT parameters (array elements are considered values).

> Also, function return values can be defined as an array.

Do you think this is a good time to refactor BlockManager.*function APIs so that generics will be used ? Eventually if we will implement the function's "RETURN CLASS" clause, I think we will have no choice but to use generics. The only problem here is that we will need to return the array "by value" (i.e. a copy) and not the original array passed to the RETURN clause.

    This means that extent variables (fields too?) can be passed to/returned from these blocks.

Yes, as long as the extent size is the same.

Another issue we will need to fix is how the GET-SIGNATURE reports such parameters - we will need to save in name_map.xml each parameter's extent size (if fixed). This will also be usefull when validating the parameter, as we will need to know the parameter's exent size (actually, we can emit the extent size directly in the generated code, as only fixed-size extent parameters will need special attention).

#4 - 02/04/2013 01:45 PM - Constantin Asofiei

- Status changed from New to WIP

#5 - 02/05/2013 04:15 PM - Greg Shah

```
Do you think this is a good time to refactor BlockManager.*function APIs so that generics will be used ? Eventually if we will implement the function's "RETURN CLASS" clause, I think we will have no choice but to use generics.
```

It may be time for this. However, I worry about how much development time this will take, since we have a tight schedule and we are already late with milestone 4.

```
 The only problem here is that we will need to return the array "by value" (i.e. a copy) and not the original array passed to the RETURN clause.
```

We need to consider the implications. I wonder if the delegated assignment is needed here too (see #1621). Note that one problem for any modification of the original array (size and elements) is that if it was undoable, we may have some maintenance of the undo snapshots that would need to be done.

#6 - 02/06/2013 05:28 AM - Constantin Asofiei

> It may be time for this. However, I worry about how much development time this will take, since we have a tight schedule and we are already late with milestone 4.

Actually, this is quite easy: control_flow.rules needs a couple of changes to emit the BlockManager.function (or BlockManager.extentFunction in case of functions which return extent) API calls and to emit the return class (which is already computed). The other BlockManager changes are simple too. What is not that simple is to implement the function's RETURN statement to handle extent variables (but we would have to do this anyway).

So, the new APIs will look like this:

- non-extent functions:

  ```
  <T extends BaseDataType> T function(Class<T>, Block)
  <T extends BaseDataType> T function(Class<T>, TransactionType, Block)
  ```

- fixed-size extent functions:

  ```
  <T extends BaseDataType> T[] extentFunction(Class<T>, int, Block)
  <T extends BaseDataType> T[] extentFunction(Class<T>, int, TransactionType, Block)
  ```

- dynamic-size extent functions:

  ```
  <T extends BaseDataType> T[] extentFunction(Class<T>, Block)
  <T extends BaseDataType> T[] extentFunction(Class<T>, TransactionType, Block)
  ```

- the new signature of functionBlock is:

  ```
   <T extends BaseDataType> T[] functionBlock(TransactionType level,
                                              Block           block,
                                              Class<T>        retType,
                                              int             extent)
  ```

  where extent will be -2 for non-extent functions, -1 for dynamic-extent functions, extent-size for other cases. In non-extent case, the returned value will be placed on the 0 element (which will be actually returned by the function APIs).

#7 - 02/06/2013 08:25 AM - Greg Shah

OK. Do it.

#8 - 02/06/2013 11:47 AM - Constantin Asofiei

1. is cleaner to let -1 for non-extent, 0 for dynamic-extent and >0 for fixed-size extent (these constants will be used in other places, plus for array/parameters instantiation, i.e. will be possible to create an array of size zero - i.e. a non-initialized dynamic-extent variable)
2. A problem with extent as parameters is that we can no longer convert an 1-extent variable to a no-extent variable. This applies for fields too. This is because 1-extent vars/fields can be passed as parameters, plus they can be on the right side of an dynamic-extent-var = 1-extent-var statement.
3. All extent fields will need simple getter (which returns an array) because in 4GL it is valid:

   ```
   def temp-table tt1 field f1 as int extent 5.
   def var i as int extent.
   function f0 returns int (input j as int extent 5).
   end.

   f0(tt1.f1). /* to pass an extent field as parameter */
   i = tt1.f1. /* to be on the right-side of an assignment */
   ```

4. Please check if there are functions which return an extent, and the function is called using DYNAMIC-FUNCTION (and used in as a right-value in an assignment):

   ```
   function f0 returns int extent 2.
   end.

   def var i as int extent 2.
   i = dynamic-function("f0").
   ```

   If there are no such cases in current project, we can postpone this (else we will have to emit a ControlFlowOps.invokeDynamicExtentFunction in such cases and modify the ControlFlowOps runtime (which is not trivial).
5. must cast to Object any extent variable/field used in:
   - ControlFlowOps.invoke APIs - to not "explode" the elements of the array as individual elements of the varargs.
   - for TransactionManager.registerUndo/.register/.deregister, as these accept Undoables (and as I understand undo support for extent is done by saving each element from the array, not the array reference), we still need to change these APIs to work with varargs and handle arrays properly (as TM.register does)
6. any extent INPUT parameter will be initialized as in:

   ```
   integer[] extentParam = ArrayAssigner.copyOf(integer.class, _extentParam);
   ```

7. I think the right approach is to emit the extent parameter for any function/procedure as an OutputExtentParm (as you said at #1621).

At this time, I have conversion rules which:

1. properly convert 1-extent variables/fields
2. emit getter (in DMO iface and impl) for extent fields which returns an array, i.e. integer[] getF1()
3. fixed a problem in temp-table disambiguation, which didn't take into account the field's extent
4. emit dynamic-extent and initialize it as a zero-size array
5. refactored BlockManager.*Function APIs to use generics
6. added support for RETURN ext-var for functions returning extent
7. INPUT extent parameters use a copy determined using ArrayAssigner.copyOf
8. TM.deregister works with extent vars too

#9 - 02/06/2013 04:31 PM - Greg Shah

The good news is that there is no usage of user-defined functions returning an extent in any of the project code. The reason I know this is the bad news. When we updated the parser for 10.2b, we missed the addition of the EXTENT option to the function returns clause. So progress.g needs to be updated to support that syntax. If the project code (server or gui) had that construct, it wouldn't parse today. So we can defer that work, but when we do it we will need to update the func_return rule in progress.g.

#10 - 02/07/2013 09:00 AM - Constantin Asofiei

In convert/variable_definitions.rules, on line 550, it states that all output parameters for functions will initialize to unknown. But, in OE 10.2B I think is no longer the case - maybe they changed something when they added extent param support ? If in 4GL 9.0 the real behavior is to initialize function's output parameter to unknown, then we need to add some config parameter to know the project's Progress version.

```
function func0 returns int (output i1 as int, output c1 as char).
  message "<" i1 "><" c1 ">". /* shows "<0> < >" */
end.

def var i as int.
def var c as char.
i = 10.
c  = "aa".

func0(output i,output c).

message "<" i "><" c ">". /* shows "<0> < >" */
```

#11 - 02/07/2013 10:04 AM - Greg Shah

We are already needing a similar feature for int64 support. v9 compiles some things to integer that v10 compiles to int64 (unless the -noint64 command line parameter is used).

I don't know if we can confirm that output parameter inits to unknown in v9. But let's operate on the assumption that it is correct and base our behavior on the proposed version flag. We may need some helpers that interpret the flag.

For example, we probably should put the specific progress version number in there (e.g. "9.1C" for Majic) but we will also want to be able to ask "is this v9?" instead of reading the string and parsing it in lots of different rule sets.

#12 - 02/07/2013 11:19 AM - Constantin Asofiei

To be sure we don't brake anything in MAJIC, we will need to add this flag for this task (IMO, this is a conversion issue, and I will need to handle it now).

#13 - 02/07/2013 11:25 AM - Greg Shah

OK.

#14 - 02/07/2013 01:20 PM - Constantin Asofiei

> So progress.g needs to be updated to support that syntax. If the project code (server or gui) had that construct, it wouldn't parse today. So we can defer that work, but when we do it we will need to update the func_return rule in progress.g.

I've already done this yesterday, and it was easier to alter func_stmt rule so that it will save the extent at the KW_FUNCT node:

```
         fct:KW_FUNCT^ s:malformed_symbol!
         ftype = fr:func_return!
         (
            size = extent!
            {
               if (size >= 0) ##.putAnnotation("extent",  new Long(size));
            }
         )?
```

extent rule was modified too to return the actual extent.

#15 - 02/07/2013 01:58 PM - Greg Shah

OK. My only comment is that generally, I only drop nodes (using the !) that are "syntactic sugar". For example, DOT or END are things that are there to create the tree but are not needed to understand the syntax. I'm sure there are exceptions, but I prefer to leave the tree as much as possible as it was in the code.

There are 2 advantages to this:

1. It makes it easier to do reports when we leave behind the actual tree structure.

2. In the future, we will probably write an anti-parser for Progress. This would take a Progress AST and emit 4GL source code. This would be much easier if the original code was there. The point of doing this would be to use TRPL to do refactoring for 4GL projects.

#16 - 02/07/2013 02:04 PM - Constantin Asofiei

I was dropping it because I think it was emitting the extent literal (if present) in the generated code. I will modify to hide this node in post-parse-fixups phase or annotations phase, before it gets a chance to be emitted.

#17 - 02/09/2013 12:34 PM - Constantin Asofiei

- File ca_upd20130209b.zip added

- File ca_upd20130209a.zip added

Added almost final update which adds support for:

- support for p2j.cfg.xml's new source-code-version (see comments in common-progress.rules). When converting, I have an parser warning for the getSourceCodeVersion function in common-progress.rules, but I have no idea why. I've isolate it to this code, which if removed, the warning is no longer shown:

```
         <rule>codeVersion = getConfigParameter('source-code-version')</rule>
         <!-- if not set, default to 9.1C -->
         <rule>codeVersion == null
            <action>codeVersion = "9.1C"</action>
         </rule>
```

- extent parameters
- EXTENT statement
- function returning EXTENT (only simple function calls work at this time). Note that when compiling progress.g you will have warnings as now the extent rule returns a value.
- fix for field assigned to a DYNAMIC-FUNCTION call
- OUTPUT/INPUT-OUTPUT extent parameters when variables or fields are passed. In case of fields, a special wrapper is needed - see OutputExtentField.
- DYNAMIC-FUNCTION case is not implemented yet

I still have to do a final review (I have a feeling I have some dead code in java_templates.tpl), but this is good if you want to take a look at how the code gets converted.

Note that runtime is not fully-tested yet, just conversion.

If we have a code like:

```
def var i as int extent 5.
update i validate(i[1] = i[2], "error") with frame f2.
```

our validation code will not compile. More, in P2J we have no validator set per the extent field, the validation expression is still set for the element on index 1. Code above converts to something like:

```
   integer[] i = new integer[5];
...
            f2Frame.widgetiArray0().setValidatable(new Validation10());
            f2Frame.openScope();
            FrameElement[] elementList2 = new FrameElement[]
            {
               new Element(subscript(i, 1), f2Frame.widgetiArray0()),
               new Element(subscript(i, 2), f2Frame.widgetiArray1()),
               new Element(subscript(i, 3), f2Frame.widgetiArray2()),
               new Element(subscript(i, 4), f2Frame.widgetiArray3()),
               new Element(subscript(i, 5), f2Frame.widgetiArray4())
            };

            f2Frame.update(elementList2);
...
   private class Validation10
   extends Validator
   {
      public logical validateExpression(BaseDataType _newValue_)
      {
         final integer _i = (integer) _newValue_;
         return isEqual(_i, subscript(_i, 2));
      }

      public character validateMessage(BaseDataType _newValue_)
      {
         return new character("error");
      }
   }
```

#18 - 02/11/2013 11:40 AM - Constantin Asofiei

- File ca_upd20130211c.zip added

Attached update which:

- is merged with bzr revision 10166
- emit ArrayAssigner.lengthOf in case of dynamic-extent variables (as this reports unknown if dynamic-extent is not initialized)
- the inner class extending OutputExtentParameter has setters/getters only for the variable set to the parameter by the caller. Each time the parameter changes its reference (due to a resize) and on Block.init is added a OutputExtentParameter.setParameter(<reference>) to update the parameter reference.
- in case of fixed extent output/input-output parameters, if the passed variable is dynamic-extent and not initialized, it needs to be initialized to the parameter's extent. Thus, the Block's instance field (mapping the actual parameter) needs to be initialized using <type>[] param = extParam.get(<extent>).
- implemented the new ArrayAssigner methods

What is left to do is more runtime testing, including fields passed as parameters (which are wrapped using the new OutputExtentField class).

#19 - 02/11/2013 11:46 AM - Greg Shah

```
 I have an parser warning for the getSourceCodeVersion function in common-progress.rules
```

Please provide details of the error.

```
Note that when compiling progress.g you will have warnings as now the extent rule returns a value.
```

Please provide details of the warnings. We will want to eliminate them so that real problems can be easily see in the build.

```
our validation code will not compile. More, in P2J we have no validator set per the extent field, the validati
on expression is still set for the element on index 1.
```

The individual element references inside a validation expression is a known problem. From our conversion reference:

```
The result works the same way as all other validation expression processing. The reason is simple: the validat
eExpression() method always receives a proposed value that is a scalar instance (the proposed value is not a J
ava array), so this syntax works naturally in the converted code.

If the 4GL were to contain an array subscript, it would not work properly. See the Limitations section for det
ails.
```

I will do a code review later today.

And this:

```
A special case occurs with variable or field array references (extents). In Progress, the specific extent elem
ent can be referenced without subscript when it is in a validation expression. This works in the converted cod
e (in fact it is the only form that can work right now). If the code has an explicit subscript, then it will c
onvert to code that will access the original instance of the variable/field instead of accessing the passedin
scalar instance to test against. That will mean that such validation expressions would never work properly.
```

If you can fix this with small effort, go ahead.

The validation registration processing problem was not known, but I guess I am not surprised. Again, if you can fix it quickly, then go ahead.

#20 - 02/11/2013 12:48 PM - Constantin Asofiei

> Please provide details of the error.

The warning is in red and it says:

```
------------------------------------------------------------------------------
Core Code Conversion
------------------------------------------------------------------------------

Optional rule set [customer_specific_conversion] not found.
line 1:1: unexpected token: 2347
```

> Please provide details of the warnings. We will want to eliminate them so that real problems can be easily see in the build.

```
    [antlr] .../progress.g:16836:12: warning:Rule 'extent' returns a value
    [antlr] .../progress.g:16904:12: warning:Rule 'extent' returns a value
    [antlr] .../progress.g:11584:12: warning:Rule 'extent' returns a value
    [antlr] .../progress.g:20124:12: warning:Rule 'extent' returns a value
    [antlr] .../progress.g:21891:34: warning:Rule 'extent' returns a value
```

I will fix these in progress.g.

I will look at the validation issues tomorrow and check how easy is to fix.

Note that I have some MAJIC-related files which I will need to change, as BlockManager.*function APIs have changed.

#21 - 02/11/2013 08:31 PM - Greg Shah

I have reviewed the proposed ruleset changes. They look good. I'll look at the runtime changes tomorrow.

Did you resolve the getSourceCodeVersion TRPL error? From the text, the problem is not obvious. I will have to consider it some more.

#22 - 02/12/2013 03:01 AM - Constantin Asofiei

> Did you resolve the getSourceCodeVersion TRPL error?

No, because the next step would have been to start checking the TRPL expression parser, and I didn't want to spend time on that now.

#23 - 02/12/2013 05:21 AM - Constantin Asofiei

About the validation clause attached to an extent variable/field and which uses the extent in the validation expression. I didn't check yet how complicated the changes will be, but what will need to change is this:

1. add a validation expression to each array element, in case of update i validate(...) where def var i as int extent 5. This is what I found during testing:

   ```
   def var i as int extent 3.
   update i validate(i[1] = 2, "err") with frame f1.
   ```

   When update is called, enter 2 for element on index 1 and another value for element on index 2. As no input clause was used in the validation expression, it checks the i[1] from the variable and not from the frame buffer. As i[1] is zero, validation will fail for element on index 2. Thus, I think 4GL adds the validation expression to each array element.
2. the interesting part is when variables are used as subscript in the validation expression, as in:

   ```
   def var i as int extent 3.
   def var j as int init 1.
   update i validate(i[j] = 2, "err") with frame f1.
   ```

   In this case, 4GL can't map i[j] to a specific widget, thus it uses the extent variable. In this example, no matter what value you set to element on index 1, it will not validate, as the pre-update i[j] is 0. If the validation expression is changed to input i[j] = 2, then widget value is forced and element on index 1 will pass validation when is set to 2.
3. the validation expression for each array element must be distinct, as it needs to disambiguate between i[<idx>] when <idx> is for the current element and i[<idx>] when <idx> is for another element (only when <idx> is a num literal). Thus, the following code:

   ```
   def var i as int extent 3.
   update i validate(i[1] = i[2], "err") with frame f1.
   ```

   needs to be converted as:

   ```
   ...
   f1Frame.widgetiArray0().setValidatable(new Validation1());
   f1Frame.widgetiArray1().setValidatable(new Validation2());
   f1Frame.widgetiArray2().setValidatable(new Validation3());
   ...
      private class Validation1
      extends Validator
      {
         public logical validateExpression(BaseDataType _newValue_)
         {
   ```

```
         final integer _i = (integer) _newValue_;
         /* as this validation expression is for element 1, all i[1] need to be substituted with the _newV
alue_ */
         return isEqual(_i, subscript(i, 2));
      }
      ...
   }
   private class Validation2
   extends Validator
   {
      public logical validateExpression(BaseDataType _newValue_)
      {
         /* as this validation expression is for element 2, all i[2] need to be substituted with the _newV
alue_ */
         final integer _i = (integer) _newValue_;
         return isEqual(subscript(i, 1), _i);
      }
      ...
   }
   private class Validation3
   extends Validator
   {
      public logical validateExpression(BaseDataType _newValue_)
      {
         /* as this validation expression is for element 3 and that is not used in the validation expressi
on, then elements 1 and 2 are refered directly */
         return isEqual(subscript(i, 1), subscript(i, 2));
      }
      ...
   }
```

4. when using INPUT clause in the validation expression, it will convert to frame widget getters, which work OK even with extent fields.

#24 - 02/12/2013 09:43 AM - Greg Shah

Code feedback:

1. The OutputExtentParameter getVariable(int) method should be located above the private methods in the file, instead of leaving it at the end, below the inner classes.

2. The ArrayAssigner.lengthOf() passes a null or an int to the new integer() constructor. I presume that javac will determine that this should use the integer(Number) constructor that will properly deal with null as unknown. BUT, this seems like too much hidden dependencies. Please make this

code less dependent by explicitly returning new integer() or new integer(int) depending on the situation.

3. In FieldReference, I think the getExtent() method name is misleading. It suggests that it will report the "extent" (size) of the field, but really it is just a artificial construct needed because Java cannot have polymorphic methods based only on return values. Perhaps Eric has an idea of a name he would prefer.

Otherwise, it all looks really good.

#25 - 02/12/2013 11:40 AM - Constantin Asofiei

List of rules describing behavior of mixed extent and dynamic-extent in parameters and assignments:

1. output mode:
A. parameter is dynamic-extent
invoked simple:
- passed variable is dynamic-extent, not initialized: OK
- passed variable is dynamic-extent, initialized: Err#1
- passed variable is fixed-extent: Err#1
invoked dynamically: the same
B. parameter is fixed-size
invoked simple:
- passed variable is dynamic-extent, not-initialized: OK
- passed variable is dynamic-extent, initialized to same extent: OK
- passed variable is dynamic-extent, initialized to different extent: Err#1
- passed variable is fixed-extent (same size): OK
- passed variable is fixed-extent (different size): Compiler error
invoked dynamically (differences):
- passed variable is fixed-extent (different size): Err#1

2. input mode:
A. parameter is dynamic-extent - everything works
B. parameter is fixed-size
invoked simple:
- passed variable is dynamic-extent, not-initialized: Err#2
- passed variable is dynamic-extent, initialized to same extent: Ok
- passed variable is dynamic-extent, initialized to different extent: Err#1
- passed variable is fixed-extent (same size): OK
- passed variable is fixed-extent (different size): Compiler ERror
invoked dynamically (differences):
- passed variable is fixed-extent (different size): Err#1

3. input-output mode:
A. parameter is dynamic-extent - everything works
B. parameter is fixed-size
invoked simple:
- passed variable is dynamic-extent, not-initialized: OK
- passed variable is dynamic-extent, initialized to same extent: Ok
- passed variable is dynamic-extent, initialized to different extent: Err#1
- passed variable is fixed-extent (same size): OK
- passed variable is fixed-extent (different size): Compiler ERror
invoked dynamically (differences):
- passed variable is fixed-extent (different size): Err#1

4. function return definition is dynamic-extent:
A. returns dynamic-extent var, not initialized:
- assigned to dynamic-extent var, not initialized: OK
- assigned to a dynamic-extent var, initialized: Err#3
- assigned to a fixed-extent var: Err#4
B. returns dynamic-extent var, initialized:
- assigned to dynamic-extent var, not initialized: OK
- assigned to a dynamic-extent var, initialized same size: OK
- assigned to a dynamic-extent var, initialized different size: Err#3
- assigned to a fixed-extent var, same size: OK
- assigned to a fixed-extent var, different size: Err#5
C. returns fixed-extent var:
- assigned to dynamic-extent var, not initialized: OK
- assigned to a dynamic-extent var, initialized same size: OK
- assigned to a dynamic-extent var, initialized different size: Err#3
- assigned to a fixed-extent var, same size: OK

- assigned to a fixed-extent var, different size: Err#5

5. function return definition is fixed-extent:
A. returns dynamic-extent var, not initialized:
- assigned to dynamic-extent var, not initialized: OK
- assigned to a dynamic-extent var, initialized: Err#3
- assigned to a fixed-extent var, same size as func def return: Err#4
- assigned to a fixed-extent var, different size than func def return: Compiler Err#5
dynamic-function case:
- assigned to a fixed-extent var, different size than func def return: Err#4
B. returns dynamic-extent var, initialized:
- assigned to dynamic-extent var, not initialized: OK
- assigned to a dynamic-extent var, initialized same size as returned var: OK
- assigned to a dynamic-extent var, initialized different size than returned var: Err#3
- assigned to a fixed-extent var, same size as func def return, but different than the returned var extent: Err#5
- assigned to a fixed-extent var, same size as func def return, same as the returned var extent:OK
- assigned to a fixed-extent var, different size than func def return, same as the returned var extent:Compiler Error #5
- assigned to a fixed-extent var, different size than func def return, different than the returned var extent:Compiler Error #5
dynamic-function case:
- assigned to a fixed-extent var, different size than func def return, same as the returned var extent:OK
- assigned to a fixed-extent var, different size than func def return, different than the returned var extent:Error #5
C. returns fixed-extent var:
- assigned to dynamic-extent var, not initialized: OK
- assigned to a dynamic-extent var, initialized same size: OK
- assigned to a dynamic-extent var, initialized different size: Err#3
- assigned to a fixed-extent var, same size: OK
- assigned to a fixed-extent var, different size: Compiler Error Err#5
dynamic-function case:
- assigned to a fixed-extent var, different size: Err#5

6. extent-to-extent var assignment
A. left-side is dynamic-extent, not initialized
- right-side is dynamic-extent, not-initialized: OK
- right-side is dynamic-extent, initialized: OK
- right-side is fixed-extent: OK
B. left-side is dynamic-extent, initialized
- right-side is dynamic-extent, not-initialized: Err#3
- right-side is dynamic-extent, initialized, same size: OK
- right-side is dynamic-extent, initialized, different size: Err#3
- right-side is fixed-extent, same size: OK
- right-side is fixed-extent, different size: Err#3
C. left-side is fixed-extent
- right-side is dynamic-extent, not-initialized: Err#4
- right-side is dynamic-extent, initialized, same size: OK
- right-side is dynamic-extent, initialized, different size: Err#5
- right-side is fixed-extent, same size: OK
- right-side is fixed-extent, different size: Compiler Error Err#5

7. subscript usage
A. subscript is constant, in bounds.
- var is dynamic-extent, not initialized: Err#6
- var is dynamic-extent, initialized: OK
- var is fixed-extent: OK
B. subscript is constant, outside bounds.
- var is dynamic-extent, not initialized: Err#6
- var is dynamic-extent, initialized: Err#7
- var is fixed-extent: Compiler error
C. subscript is var, in bounds
- var is dynamic-extent, not initialized: Err#6
- var is dynamic-extent, initialized: OK
- var is fixed-extent: OK
D. subscript is var, outside bounds
- var is dynamic-extent, not initialized: Err#6
- var is dynamic-extent, initialized: Err#7
- var is fixed-extent: Err#8
E. subscript is unknown var
- var is dynamic-extent, not initialized: OK
- var is dynamic-extent, initialized: OK
- var is fixed-extent: OK
F. subscript is var, set to 0
- var is dynamic-extent, not initialized: Err#8
- var is dynamic-extent, initialized: Err#8
- var is fixed-extent: Err#9

Err#1 (no error raised, just message unless NO-ERROR clause is present):

```
Extent parameter dimension of <var-extent> from procedure <source-proc:name> is mismatched with sub-procedure
<fname> <this-proc:name> parameter dimension of <par-extent> ...(328) (11428)
```

Err#2 (no error raised, just message unless NO-ERROR clause is present):

```
Calling procedure <source-proc:name> cannot input an indeterminate extent parameter if the sub-procedure <fnam
e>  <this-proc:name> parameter is a fixed size extent. (11421)
```

Err#3 (error is raised)

```
Indeterminate extent is already fixed to a dimension of <var-extent>. (13738)
```

Err#4 (error is raised)

```
Uninitialized array used as source of assignment. (14906)
```

Err#5 (error is raised)

```
Whole-array assignment target and source must have the same extent unless the target is indeterminate. (14905)
```

Err#6 (error is raised)

```
Invalid runtime use of indeterminate extent.  It must be set to a fixed dimension. (11387)
```

Err#7 (error is raised)

```
Array subscript <idx> is out of range.  The indeterminate extent is fixed to a dimension of <var-extent>. (113
88)
```

Err#8 (error is raised)

```
** Array subscript <idx> is out of range. (26)
```

Err#9 (error is raised)

```
** Array subscript 0 is out of range. (26)
** Unable to update  Field. (142)
```

#26 - 02/12/2013 12:52 PM - Greg Shah

Wow. Great work!

If we are far off from this implementation, then we will need to defer the runtime work until later. If close, then we can fix the runtime parts. I consider "close" as effort of 1 day or less.

In regard to the extent rule usage in progress.g, it seems that the following places can match the extent usage:

1. def_var_stmt
2. def_parm_stmt which calls regular_parm (where extent is used)
3. def_prop_stmt
4. constructor_stmt which calls run_stmt_parms with defineVariable true which calls parameter with defineVariable true where extent can used
5. method_stmt (same call tree as constructor stmt)
6. func_stmt (same call tree as constructor stmt)
7. getter_setter (same call tree as constructor stmt)

All of these places can define an extent var/parm and they all need to get an "extent" annotation if there is size >= 0. I assume that you are already doing this, but I just want to confirm it. The annotation needs to be created at the node that would be matched in the "var_def" function in common-progress.rules. I am putting together better reports for the various extent cases and I want to be able to rely upon this annotation.

#27 - 02/12/2013 02:57 PM - Constantin Asofiei

I'm implementing these rules from last to first, and not obvious cases I'll leave last. About the validation problems: once I found the correct place to fix this (in annotations/validation_post.rules), the fix was straightforward.

About progress.g's extent rule - I am on the way of moving all code from downPath("KW_EXTENT")-like usage to "extent" annotation.

#28 - 02/13/2013 09:18 AM - Constantin Asofiei

> 1. def_var_stmt

No special work needed, define_stmt calls sym.annotateVariableOptions which sets the extent annotation

> 2. def_parm_stmt which calls regular_parm (where extent is used)

I explicitly added the extent annotation.

> 3. def_prop_stmt

Same as def_var_stmt

> 4. constructor_stmt which calls run_stmt_parms with defineVariable true which calls parameter with defineVariable true where extent can used

5. method_stmt (same call tree as constructor stmt)
6. func_stmt (same call tree as constructor stmt)
7. getter_setter (same call tree as constructor stmt)

All these cases are handled by parameter rule, which calls sym.annotateVariableOptions, which sets the extent annotation.

Some progress details, fixed issues (related to errors):

1. subscript usage
2. extent-to-extent assignment
3. extent-to-var assignment
4. I've added rules to register each dynamic-extent array with ArrayAssigner, so that we can know which array is dynamic-extent and which one is not (even after the dynamic-extent has set its size).

I'll see what else I can finish in the next couple of hours and I'll place a final update today.

#29 - 02/13/2013 11:14 AM - Constantin Asofiei

- File ca_upd20130213b.zip added

- File ca_upd20130213a.zip added

Beside the issues on comment 28, the attached fix adds only some startup code for extent parameter validation. What is left is to:

1. finish parameter validation (use as reference extent_params_proc.p, which covers IMO all cases)
2. test function returning extent
3. implement dynamic-function for functions returning extent
4. test undo support for extent variables/fields, used as parameters and/or with dynamic extent.

I'll post in a separate task the changed MAJIC-related files shortly.

#30 - 02/13/2013 11:29 AM - Constantin Asofiei

- % Done changed from 0 to 70

#31 - 02/13/2013 03:03 PM - Constantin Asofiei

- Status changed from WIP to Review

#32 - 02/13/2013 07:02 PM - Greg Shah

Feedback:

1. The int members of enum ArgValidationErrors in ControlFlowOps need javadoc.

2. The progress.g needs comments about how the upstream code will call sym.annotateVariableOptions() and that code will see the EXTENT node

and will properly honor setting the extent annotation for def_var_stmt, def_prop_stmt... otherwise someone reading the code might think that the discarded size var means that the annotation will not be made.

3. I think there is an essential problem with the progress.g changes. It is valid in the 4GL to have EXTENT 0. This doesn't mean that the var is an indeterminate length array. This means that the var is scalar and not an array. But for us, the 0 means that it is an indeterminate length array. Please see the 4GL docs for define var where it describes this. Note that the current project DOES have several cases of EXTENT 0. This is important because the use of AS of LIKE can require the extent to be "removed" by explicitly using EXTENT 0.

#33 - 02/14/2013 12:12 AM - Constantin Asofiei

About issue 3 - I chose 0 to represent dynamic extent because this will produce a valid reference in the converted code, i.e. a 0-length array; the alternative would have been to initialize the array variable with null, and I didn't like that. I think for these 0-extent cases there is no way but to drop (or hide, if it works) the KW_EXTENT node, and leave an annotation behind that the KW_EXTENT node was dropped/hidden.

#34 - 02/14/2013 08:31 AM - Greg Shah

```
I think for these 0-extent cases there is no way but to drop (or hide, if it works) the KW_EXTENT node, and le
ave an annotation behind that the KW_EXTENT node was dropped/hidden.
```

Yes, I agree it must be dropped and the resulting var should be treated as scalar.

#35 - 02/14/2013 11:25 AM - Constantin Asofiei

- File ca_upd20130214b.zip added

Attached version does the following:

1. there is a feature which I think is valid only in 4GL v9:

   ```
   def var x as int extent 5.
   update x validate (x = 0, "msg") with frame f1.
   ```

   v10 doesn't allow the "widget" reference inside the validation expression using only the array's name, without subscript. Thus, I've added protection to allow this code to convert properly only in v9.
2. fixed a bug in validation_post.rules which caused MAJIC conversion error
3. dynamic-extent vars are mapped as -1 extent, but in converted code, it will create a 0-length array. With this, I fixed the issues of defining scalar variables using def var ... extent 0, and I managed to do this without making the parser remove the KW_EXTENT node (thus if anyone else looks through the generated ASTs, all the info is there).
4. fixed conversion of fields with their extent set to 0 (must be scalars)

There is one more problem about widget validation which I didn't test until now. When updating extent fields, the following code will not convert correctly (assuming person.schedule is defined of extent 5 and validation is set for the field, in the schema):

```
update person.schedule with frame f1.
```

For this code, validation expression is generated only for one field (not all fields). The problem is a little complicated to fix, as the expansion of the extent field is done AFTER the schema validation expressions are added to the frame (schema/fixups.xml), and the validation expression does not propagate to all the subscripted fields.

#36 - 02/14/2013 01:06 PM - Greg Shah

Code Review Feedback:

This version looks good. I am fine with this, however I have questions:

1. I assume you have resolved the TRPL expression parsing error? If not, then we must solve this before going into test.
2. Are any of the listed problems above going to cause regressions in Majic?

#37 - 02/14/2013 01:25 PM - Constantin Asofiei

> 1. I assume you have resolved the TRPL expression parsing error? If not, then we must solve this before going into test.

The problem is gone in latest version, and after a quick check, looks like the root cause was with a return variable named "digit". If I changed this code:

```
      <function name="is_digit">
         <parameter name="str"   type="java.lang.String" />
         <return    name="digit" type="java.lang.Boolean" />

         <rule>
            digit = str != null and 
                    (str == "0" or str == "1" or str == "2" or str == "3" or str == "4" or 
                     str == "5" or str == "6" or str == "7" or str == "8" or str == "9")
         </rule>
      </function>
```

to this code:

```
      <function name="is_digit">
         <parameter name="str"   type="java.lang.String" />
         <return    name="digit1" type="java.lang.Boolean" />

         <rule>
            digit1 = str != null and 
                    (str == "0" or str == "1" or str == "2" or str == "3" or str == "4" or 
                     str == "5" or str == "6" or str == "7" or str == "8" or str == "9")
         </rule>
      </function>
```

There are no more parser errors. Btw, you might have noticed that is_digit was changed in latest update, and the reason was that the function was returning always false. I guess there are problems in expression parser if the literal name is the same as a token name ?

> 2. Are any of the listed problems above going to cause regressions in Majic?

I don't expected to, but I will have a good answer (conversion-wise) tomorrow morning. And the schema validation problem with extent fields I don't think it is used in MAJIC.

#38 - 02/15/2013 07:02 AM - Constantin Asofiei

- File ca_upd20130215c.zip added

After checking the converted sources, the following issues were fixed/found:

1. in some programs, the validation expression for an extent field (used in an update statement) now is set for all fields, not just first field. I don't think this should fail at runtime.
2. in a program, there is a case like def var x like tt1.f1 extent 1, where tt1.f1 is extent 10. This variable is used in frames, but was not treated as extent by my changes (array_expansion.rules had a problem which was fixed). Also, I can't say if 4GL v9 treated this as extent or not, as the var's label is explicitly set, and can't check using that.
3. fixed casting to Object in case the extent has a subscript
4. fixed wrapping the extent using OutputExtentParameter in case the extent has a subscript

Also, looks like there are some cases in MAJIC where the var AST in a validation expression/msg which is not used does not get dropped, and the code has dead-vars like final integer _var = (integer) _newValue_; in the validateExpression or validateMessage methods.

I'm reconverting again and I'll let you know the results.

#39 - 02/15/2013 11:12 AM - Greg Shah

The code looks good. Assuming your tests show that it needs no further changes, the plan is as follows:

1. As soon as the current changes (being tested in staging) are checked into bzr, merge up to that level.
2. Apply it to staging, reconvert and run the harness.

#40 - 02/15/2013 11:20 AM - Constantin Asofiei

What I'm currently reconverting doesn't fix the dead-vars emitted in validateExpression and validateMessage methods. Let me know if I should fix this before regression testing in staging (I've tried to duplicate it using a standalone testcase, but is not that obvious and I wasn't able to duplicate it).

#41 - 02/15/2013 11:26 AM - Greg Shah

Do these problems already exist in Majic today?

#42 - 02/15/2013 11:32 AM - Constantin Asofiei

OK, I've checked again and looks like I've been miss-interpreting the diff result, this update actually solves the issue of these dead-vars (which are in MAJIC today).

#43 - 02/15/2013 12:09 PM - Constantin Asofiei

Unfortunately the conversion on the linux box crashed when it was almost at the end, so I will not have a conclusion about conversion regression testing until tomorrow morning.

#44 - 02/15/2013 03:22 PM - Constantin Asofiei

- File ca_upd20130215e.zip added

Attached update merged with bzr revision 10173.

#45 - 02/15/2013 07:44 PM - Greg Shah

FYI, there are some javadoc errors. You can fix them later (in a different check in).

```
 [javadoc] /mnt/san/sata/gc/20121029/p2j/src/com/goldencode/p2j/util/OutputExtentParameter.java:43: warning - Tag @link: can't find get(int) in com.goldencode.p2j.util.OutputExtentParameter
  [javadoc] /mnt/san/sata/gc/20121029/p2j/src/com/goldencode/p2j/util/OutputExtentParameter.java:40: warning - Tag @link: can't find get(int) in com.goldencode.p2j.util.OutputExtentParameter
```

#46 - 02/16/2013 02:42 AM - Constantin Asofiei

- File ca_upd20130216a.zip added

This update:
- fixes decimals clause for 1-extent fields.
- fixes javadoc
- adds a TODO in decimal.setPrecision, for dynamic-extent vars - for these cases, the precision needs to be saved "by reference" and each time the dynamic-array's reference changes, the precision needs to be set again.

#47 - 02/18/2013 03:59 AM - Constantin Asofiei

- File ca_upd20130218a.zip added

This update fixes the following issues discovered on Saturday, during regression testing:

- ControlFlowOps NPE
- BlockManager NPE
- FieldReference subscript problem

#48 - 02/18/2013 08:29 AM - Constantin Asofiei

ca_upd20130218a.zip has been committed as 10175 (has passed regression testing).

#49 - 02/18/2013 09:21 AM - Constantin Asofiei

- File ca_upd20130218d.zip added

Added testcases, for reference (they are in bzr too).

#50 - 03/17/2013 10:01 AM - Constantin Asofiei

- File ca_upd20130317d.zip added

- File ca_upd20130317c.zip added

When fixing conversion of schema validation expressions for extent fields, it hit me that the P2J approach using SCHEMA_VALIDATIONS is completely wrong (if you think I'm mistaken, please explain why), as if a field with schema VALEXP appears in multiple frames, each occurance would register the same Validation# inner class as its validatable, with its widget. This is wrong as the VALEXP needs to be executed under the context of each frame (i.e. if the current widget is referenced, then the widget from the current frame must be used, and not one from another frame or just the field getter).

Attached update removes all usage of SCHEMA_VALIDATIONS node and refactors all code this way:

1. fixups/schema_validations will attach the VALEXP as a normal KW_VALIDATE node in the format phrase for each field (even if it appears

multiple times in the frame). This KW_VALIDATE node is marked with the schema_validation annotation, so downstream processing will be aware that this is the implicit validation
2. annotations/validation_prep will process the KW_VALIDATE node and:

- will remove the implicit/explicit validation if the frame has the NO-VALIDATE clause
- will remove the impliict validation if the the field has explicit validation and frame has no NO-VALIDATE clause.

#51 - 03/17/2013 10:03 AM - Constantin Asofiei

- File deleted (ca_upd20130317d.zip)

#52 - 03/17/2013 10:03 AM - Constantin Asofiei

- File ca_upd20130317d.zip added

Renamed test file.

#53 - 03/17/2013 11:16 AM - Constantin Asofiei

There are lots of changes in MAJIC converted code, here are some examples:

1. for extent fields, all elements get their distinct validation inner class
2. previously not-emitted validation inner classes are now emitted (this should clean with Ovidiu's related work, i.e. don't emit validation inner classes if they are not needed)
3. previously emitted validation code is now dropped
   and more

I'll review each one and see why this is happening, and I'll update here.

#54 - 03/17/2013 04:34 PM - Greg Shah

I'm OK with the changes. If all of the Majic modifications appear to be correct, we will go ahead with runtime testing of this update by itself.

#55 - 03/18/2013 10:42 AM - Constantin Asofiei

- File ca_upd20130318a.zip added

New version of the update in which:

1. the NO-VALIDATE clause with a frame must affect only the schema validation expressions, and not the explicit validation (old update was touching the explicit validation too).
2. field references in FORM will never trigger schema validation registration
3. fixed schema validation expressions in case of UPDATE <record> or INSERT <record>-like statements.

I'm putting this through conversion regression testing.

#56 - 03/18/2013 11:52 AM - Greg Shah

Code Review:

My only question is this: the schema_validation.rules includes support for the INSERT statement, but then the FORMAT_PHRASE node is removed (which would presumably contain the validation expression) from insert_statement_expansion.rules.

Otherwise, I don't have any problems with the code changes.

#57 - 03/18/2013 11:58 AM - Constantin Asofiei

> My only question is this: the schema_validation.rules includes support for the INSERT statement, but then the FORMAT_PHRASE node is removed (which would presumably contain the validation expression) from insert_statement_expansion.rules.

The insert_statement_expansion.rules duplicates the entire KW_INSERT node, to create the new KW_CREATE node (this includes the FORMAT_PHRASE). If not removed from the KW_CREATE, it would end up with a FORMAT_PHRASE, which is not correct. For the KW_UPDATE, again the entire KW_INSERT is duplicated, which includes the FORMAT_PHRASE, and will remain attached to this node.

#58 - 03/18/2013 12:30 PM - Greg Shah

This makes sense. I should have read the surrounding code more carefully and I would have seen this.

If it passes conversion testing AND if you deem the Majic changes to be correct, then go ahead and check in/distribute.

#59 - 03/19/2013 07:02 AM - Constantin Asofiei

> If it passes conversion testing AND if you deem the Majic changes to be correct, then go ahead and check in/distribute.

All changes are caused by the fact that the FORM doesn't trigger schema validation expression registration anymore. Separate runtime regression testing is still recommended. Commmitted to bzr revision 10298.

#60 - 03/19/2013 07:03 AM - Constantin Asofiei

- File ca_upd20130319a.zip added

Attached merged update, committed to bzr revision 10298.

#61 - 04/25/2013 11:07 AM - Greg Shah

What is to do on this task, if anything?

#62 - 04/25/2013 11:53 AM - Constantin Asofiei

The only TODO is this:

- adds a TODO in decimal.setPrecision, for dynamic-extent vars - for these cases, the precision needs to be saved "by reference" and each time the dynamic-array's reference changes, the precision needs to be set again.

#63 - 04/25/2013 11:56 AM - Greg Shah

Please create a separate task for this remaining TODO and we will close this. Make sure to put a time estimate in there and add me as a watcher. Set the target version for milestone 11.

#64 - 04/25/2013 12:15 PM - Greg Shah

- Status changed from Review to Closed

- % Done changed from 70 to 100

#65 - 11/16/2016 11:43 AM - Greg Shah

- Target version changed from Milestone 7 to Runtime Support for Server Features

Files

ca_upd20130209a.zip	546 KB	02/09/2013	Constantin Asofiei
ca_upd20130209b.zip	4.52 KB	02/09/2013	Constantin Asofiei
ca_upd20130211c.zip	547 KB	02/11/2013	Constantin Asofiei
ca_upd20130213a.zip	553 KB	02/13/2013	Constantin Asofiei
ca_upd20130213b.zip	8.74 KB	02/13/2013	Constantin Asofiei
ca_upd20130214b.zip	566 KB	02/14/2013	Constantin Asofiei
ca_upd20130215c.zip	572 KB	02/15/2013	Constantin Asofiei
ca_upd20130215e.zip	581 KB	02/15/2013	Constantin Asofiei
ca_upd20130216a.zip	586 KB	02/16/2013	Constantin Asofiei
ca_upd20130218a.zip	586 KB	02/18/2013	Constantin Asofiei
ca_upd20130218d.zip	10.7 KB	02/18/2013	Constantin Asofiei
ca_upd20130317c.zip	338 KB	03/17/2013	Constantin Asofiei
ca_upd20130317d.zip	383 Bytes	03/17/2013	Constantin Asofiei
ca_upd20130318a.zip	338 KB	03/18/2013	Constantin Asofiei
ca_upd20130319a.zip	338 KB	03/19/2013	Constantin Asofiei