User Interface - Bug #2795

Bug # 2677 (New): fix drawing and functional differences between P2J GUI and 4GL GUI

scoped variables/tables/etc and persistent procedures

10/27/2015 08:47 AM - Constantin Asofiei

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

History

#1 - 10/27/2015 08:49 AM - Constantin Asofiei

There is a new feature in GUI code which doesn't exist in P2J: if program A defines something NEW SHARED and is ran persistent, this data is visible to any other program B executed from within A's body (via let's say an internal procedure in A).

Currently, I think any shared data which is created within program A gets destroyed when A's external procedure is finished. We need some support to save this data and restore it when some internal procedure or function in A is ran.

#2 - 10/27/2015 11:01 AM - Constantin Asofiei

- % Done changed from 0 to 100

- Target version set to Milestone 12

- Status changed from New to WIP

The solution is in branch 2677a rev 10969.

The fix required to save the scoped data in SharedVariableManager and push/pop it each time a persistent procedure's internal entry is executed.

A testcases which shows this is uast/procedures/shared_data_a.p, uast/procedures/shared_data_b.p and uast/procedures/shared_data_run.p.

#3 - 10/28/2015 01:55 PM - Greg Shah

Code Review Task Branch 2677a Revision 10969

I'm good with the changes, but have one area in which I'm confused.

SVM.scopeFinished() uses mainwa.shared.getDictionaryAtScope(0, false); to add resources for persistent external procedures.

SVM.scopeDeleted() uses mainwa.shared.getDictionaryAtScope(gscope, true) where gscope = mainwa.shared.size() - 1.

It is not clear to me why it is safe to assume that these will always refer to the same scope.

#4 - 10/28/2015 02:03 PM - Constantin Asofiei

Greg Shah wrote:

> Code Review Task Branch 2677a Revision 10969
>
> I'm good with the changes, but have one area in which I'm confused.
>
> SVM.scopeFinished() uses mainwa.shared.getDictionaryAtScope(0, false); to add resources for persistent external procedures.

Non GLOBAL SHARED resources are added to the current scope - so this saves the current scope, for the persistent procedure being exited.

> SVM.scopeDeleted() uses mainwa.shared.getDictionaryAtScope(gscope, true) where gscope = mainwa.shared.size() - 1.

Ah, now I see it... what I wanted to do is remove any GLOBAL SHARED defined in the persistent procs, from the global scope. but I don't think this works properly, as these are not saved in the current scope, anyway.

I'll do some more testing

#5 - 11/10/2015 05:12 AM - Greg Shah

Did you put the final changes in for this one? Can this be closed?

#6 - 11/10/2015 07:37 AM - Constantin Asofiei

Greg Shah wrote:

> Did you put the final changes in for this one? Can this be closed?

No, haven't got the chance to finish it. Will do after #2809

#7 - 12/09/2015 11:38 AM - Constantin Asofiei

There is another issue here, for DEFINE NEW GLOBAL SHARED TEMP-TABLE: if there already exists a declaration for the same temp-table, then the statement must be treated as DEFINE SHARED TEMP-TABLE. Currently P2J converts this to:

```
Tt21_1.Buf tt2 = TemporaryBuffer.define(Tt21_1.Buf.class, "tt2", "tt2", false);

public void execute()
{
   externalProcedure(new Block()
   {
      public void body()
      {
         RecordBuffer.openScope(tt2);
         SharedVariableManager.addTempTable(ScopeLevel.GLOBAL, "tt2", tt2);
```

So, regardless if I check if the temp-table is already globally defined, tt2 field in the java code is already assigned (and a temp-table created, when is not needed).

This needs to be converted to something else... maybe something like for the example above, and don't call TemporaryBuffer.define unless this is the first definition of this temp-table as GLOBAL:

```
Tt21_1.Buf tt2 = SharedVariableManager.addTempTable(ScopeLevel.GLOBAL, "tt2", Tt21_1.Buf.class, "tt2", "tt2",
false);
```

#8 - 12/09/2015 12:30 PM - Eric Faulhaber

Constantin Asofiei wrote:

> There is another issue here, for DEFINE NEW GLOBAL SHARED TEMP-TABLE: if there already exists a declaration for the same temp-table, then the statement must be treated as DEFINE SHARED TEMP-TABLE. Currently P2J converts this to:
> [...]
> So, regardless if I check if the temp-table is already globally defined, tt2 field in the java code is already assigned (and a temp-table created, when is not needed).
>
> This needs to be converted to something else... maybe something like for the example above, and don't call TemporaryBuffer.define unless this is the first definition of this temp-table as GLOBAL:
> [...]

Your initial solution seems to be runtime-only, but it seems now you are discussing a conversion issue. Or have I misunderstood your statement, "This needs to be converted to something else..."? Do you mean that the purpose of TemporaryBuffer.define has to be changed/converted to something else, based on which temp-table resources exist on the stack at the time it is invoked? Or are you suggesting a change to static conversion? If the latter, I'm not sure how that would work.

#9 - 12/09/2015 12:53 PM - Constantin Asofiei

Eric Faulhaber wrote:

> Your initial solution seems to be runtime-only, but it seems now you are discussing a conversion issue.

Yes, this is a new conversion issue as a DEFINE NEW GLOBAL SHARED TEMP-TABLE sometimes needs to be treated as DEFINE SHARED TEMP-TABLE, if the table is already defined globally by someone else. See SVM.addVariable - it has this code:

```
      if (scope == ScopeLevel.GLOBAL)
      {
         T current = (T) wa.shared.getSymbolAtScope(name, -1);

         // ignore repeated global definitions
         if (current != null)
         {
            return current;
         }
      }
```

which checks the global registry for the var name - and return that if it exists. Otherwise, it defaults to registering the variable into the global scope. We need a similar approach for the temp-tables.

> Or have I misunderstood your statement, "This needs to be converted to something else..."? Do you mean that the purpose of TemporaryBuffer.define has to be changed/converted to something else, based on which temp-table resources exist on the stack at the time it is invoked? Or are you suggesting a change to static conversion? If the latter, I'm not sure how that would work.

The converted code should be something like this:

```
   Tt21_1.Buf tt2 = SharedVariableManager.addTempTable(ScopeLevel.GLOBAL, "tt2", Tt21_1.Buf.class, "tt2", "tt2
");

   public void execute()
   {
      externalProcedure(new Block()
      {
         public void body()
         {
            RecordBuffer.openScope(tt2);
            // this one will not be emitte: SharedVariableManager.addTempTable(ScopeLevel.GLOBAL, "tt2", tt2);
```

The SVM.addTempTable will check the global scope (if there is another temp-table with the same name) and return that, if it exists. Otherwise, use TemporaryBuffer.define to create the temp-table and register it in the global scope.

I need to do some tests if the table definition does not match (different field count/types, indexes, etc), for the second DEFINE NEW GLOBAL SHARED TEMP-TABLE.

#10 - 12/09/2015 01:30 PM - Eric Faulhaber

Constantin Asofiei wrote:

> The SVM.addTempTable will check the global scope (if there is another temp-table with the same name) and return that, if it exists. Otherwise, use TemporaryBuffer.define to create the temp-table and register it in the global scope.

OK, I see. So, the determination of whether to create/define the new, global temp-table would be made at runtime. My concern was that we wouldn't have enough information to do this at conversion time.

> I need to do some tests if the table definition does not match (different field count/types, indexes, etc), for the second DEFINE NEW GLOBAL SHARED TEMP-TABLE.

Note that the logic which does this during conversion is a bit complicated and is being considered for change, due to concerns Stanislav raised w.r.t. browse. See #2564 (around note 216) and #2595. I'm adding Ovidiu as a watcher, since he architected the new DMO class/interface hierarchy for temp-tables.

Whatever the outcome of that rework, static conversion should already have determined whether the Progress compiler would consider any shared temp-tables defined in separate procedures as the same entity or not. What additional concerns do you have for this at runtime?

#11 - 12/10/2015 02:29 AM - Constantin Asofiei

Guys, is the blockDepth parameter in this TemporaryBuffer.define API ever emitted?

```
public static <T extends Temporary> T define(Class<T> dmoBufIface,
                                              String variable,
                                              String legacyName,
                                              boolean global,
                                              boolean undoable,
                                              int blockDepth)
```

#12 - 12/10/2015 03:04 AM - Constantin Asofiei

Eric Faulhaber wrote:

> What additional concerns do you have for this at runtime?

In GLOBAL case: for temp-tables, if the table's schema does not match with the already shared version, then this errors occurs:

```
In procedure shared_data_d.p, shared temp-table tt2 has a conflict in field, index or undo status. (2075)
```

For variables, this error is shown:

```
shared_data_d.p Conflict in extent, datatype or undo status for global ch. (390)
```

We should implement these checks at runtime.

#13 - 12/10/2015 11:09 AM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> Guys, is the blockDepth parameter in this TemporaryBuffer.define API ever emitted?
> [...]

Apparently, no. I was not able to find any reference in the conversion and neither at runtime (except for the internal calls, having -1 as default value for that parameter).

#14 - 12/14/2015 11:09 AM - Constantin Asofiei

Ovidiu, please describe what this change in variable_definition.rules tried to solve:

```
** 051 OM  20141203          Added double-initialization check for NEW SHARED GLOBAL EXTENT-s.
```

#15 - 12/14/2015 11:45 AM - Ovidiu Maxiniuc

IIRC, there was an issue when using initialized shared variables, something like this:
Procedure p1:

```
DEFINE NEW SHARED GLOBAL VARIABLE a AS INT EXTENT 2 INIT 1.
a[1] = 2.
MESSAGE a[1].
RUN p2.
MESSAGE a[1].
```

Procedure p2:

```
DEFINE SHARED GLOBAL VARIABLE a AS INT EXTENT INIT 1.
MESSAGE a[1].
```

This printed:
2
1
1
instead of
2
2
2
because the extent variable a was automatically initialized a second time in p2.
The new code collects the variables that were created and not yet INITed and, after first (and only) initialization they are dropped from the collection so, when the second procedure is called the re-initialization is simply skipped.

#16 - 12/14/2015 12:08 PM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> IIRC, there was an issue when using initialized shared variables, something like this:
> ...
> because the extent variable a was automatically initialized a second time in p2.
> The new code collects the variables that were created and not yet INITed and, after first (and only) initialization they are dropped from the collection so, when the second procedure is called the re-initialization is simply skipped.

OK, so the code is related to this: if a NEW GLOBAL SHARED var (extent or not) is already defined (NEW GLOBAL SHARED or NEW SHARED), then it must use that, and bypass the initialization code. This applies to temp-tables/buffers too.
There are two problems here:

1. the conditional code must emit for implicit initialization of extent vars, too.
2. the conversion code is broken, the init_global_shared_extents snippet from java_templates.tpl is not emitted correctly (the new BDT[] array is emitted at the SVM.addVariable, not in the inner block.

#17 - 12/14/2015 12:18 PM - Constantin Asofiei

In regard to notes 15 and 16: I'm moving the initializer code for DEFINE NEW [GLOBAL] SHARED VARIABLE ... EXTENT ... INIT as a parameter for SharedVariableManager.addVariable as initializing the GLOBAL SHARED var is a runtime decision, not conversion. Also, this will solve both issues in note 16, and will reduce the code footprint, as the assignMulti calls will no longer be emitted for NEW SHARED vars.

#18 - 12/14/2015 01:59 PM - Constantin Asofiei

I've created branch 2795 in which rev 10960 contains:

- the GLOBAL SHARED resources are not removed when the procedure gets deleted
- reworked shared extent var conversion
- reworked shared temp-table conversion
- fixed matching of shared vars, when a DEF SHARED or DEF NEW GLOBAL SHARED matches a variable in the current registry (it must match by type, etc).
- fixed resize of extent shared vars (the shared dictionary must update the reference).

What is left:

- NO-UNDO matching for shared variables (this should be emitted as yet another parameter for the SharedVariableManager.addVariable).
- if the shared var type doesn't match, raise an exception which will be caught by ControlFlowOps and transformed into an ERROR and raised to the caller in which the RUN statement is executed.
- matching for BUFFER, TEMP-TABLE, FRAME, MENU resources (as these are the currently SHARED resources supported by P2J).

Ovidiu: can you post some details about how the TEMP-TABLEs should be matched? Is there some existing code which checks this? This needs to be done at runtime.

#19 - 12/14/2015 02:15 PM - Ovidiu Maxiniuc

> Ovidiu: can you post some details about how the TEMP-TABLEs should be matched? Is there some existing code which checks this? This needs to be done at runtime.

At this moment, there is virtually no runtime code for matching TEMP-TABLEs. All work is done at conversion time, the DMO classes and buffer variables are generated to make sure the code works (at least from the POV of compiler).
Previous work was done as part of #2595 task. The results of my research for matching tables are gathered in notes 5, 6, 8, 12.

#20 - 12/16/2015 09:18 AM - Greg Shah

> I'm moving the initializer code for DEFINE NEW [GLOBAL] SHARED VARIABLE ... EXTENT ... INIT as a parameter for SharedVariableManager.addVariable as initializing the GLOBAL SHARED var is a runtime decision, not conversion.

Did you use lambdas for this?

#21 - 12/17/2015 09:49 AM - Constantin Asofiei

Greg Shah wrote:

> > I'm moving the initializer code for DEFINE NEW [GLOBAL] SHARED VARIABLE ... EXTENT ... INIT as a parameter for SharedVariableManager.addVariable as initializing the GLOBAL SHARED var is a runtime decision, not conversion.
>
> Did you use lambdas for this?

Not yet, but I'll switch both the extent initialization and the global shared temp-table definition to lambda exprs once I've found all the behaviour.

#22 - 12/18/2015 03:54 PM - Constantin Asofiei

I've rebased 2795a from trunk rev 10960. As of rev 10966, what is missing is support for FRAME and MENU resources.

I'm planning to cleanup the code, document it and run conversion for GUI+server and full MAJIC regression testing.

If these pass, I would like to run a set of full customer tests - there are lots of changes in conversion/runtime. Ovidiu/Eric, can you help?

Finally, about shared temp-tables and extent fields: there is a new issue which is (I think) outside of the scope of this task. Consider a master (new shared) temp-table with an extent field f1 of 2. A slave (shared) temp-table is defined with field f1 extent of 3. When going back to the master temp-table, accessing index 3 directly is not possible, but is possible via the :: dereference operator. The issues here:

1. P2J doesn't support h::f1(3) construct (the Dereferenceable interface doesn't have APIs with the array index).
2. in the slave temp-table, the TemporaryBuffer.useShared will return the master Temporary instance - and this will limit the extent to 2... the slave temp-table is losing all its custom extent information (and maybe others, related to indexes, labels, etc).

#23 - 12/18/2015 05:01 PM - Eric Faulhaber

Constantin Asofiei wrote:

> If these pass, I would like to run a set of full customer tests - there are lots of changes in conversion/runtime. Ovidiu/Eric, can you help?

Yes, I'll set up an environment to convert with 2795a/10966. Let me know if you make further changes based on the outcome of your tests, such that I need to convert again.

#24 - 12/18/2015 05:05 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Finally, about shared temp-tables and extent fields: there is a new issue which is (I think) outside of the scope of this task. Consider a master (new shared) temp-table with an extent field f1 of 2. A slave (shared) temp-table is defined with field f1 extent of 3. When going back to the master temp-table, accessing index 3 directly is not possible, but is possible via the :: dereference operator. The issues here:
>
> 1. P2J doesn't support h::f1(3) construct (the Dereferenceable interface doesn't have APIs with the array index).
> 2. in the slave temp-table, the TemporaryBuffer.useShared will return the master Temporary instance - and this will limit the extent to 2... the slave temp-table is losing all its custom extent information (and maybe others, related to indexes, labels, etc).

Please create a separate issue for this. If you have test cases which show the expected outcome in Progress, please include them.

#25 - 12/18/2015 07:02 PM - Constantin Asofiei

2795a rev 10967 is a good candidate for review (there are some javadocs missing and some TODO's in appserver invocation, which I haven't checked yet, but besides those it should be OK).

I've started (on devsrv01) GUI/server conversion + MAJIC full testing (conv + runtime).

Eric Faulhaber wrote:

> Yes, I'll set up an environment to convert with 2795a/10966. Let me know if you make further changes based on the outcome of your tests, such that I need to convert again.

There are no other conversion changes in 10967.

#26 - 12/19/2015 05:06 AM - Constantin Asofiei

Unfortunately GUI and server conversion both failed with a NPE. I'm looking into it.

#27 - 12/19/2015 05:47 AM - Constantin Asofiei

Constantin Asofiei wrote:

> Unfortunately GUI and server conversion both failed with a NPE. I'm looking into it.

I forgot to check support for shared non-temp-table buffers. I need to fix this before going into conversion testing.

#28 - 12/19/2015 07:08 AM - Constantin Asofiei

2795a rev 10968 fixes the conversion issue about non-tt shared buffers. I'm running conversion again (will let you know if it passes).

#29 - 12/21/2015 10:59 AM - Constantin Asofiei

As of 2795a rev 10970 MAJIC runtime testing exposed two issues:

1. the message displayed when shared var does not exist (Shared variable %s has not yet been created) is incorrect in P2J
2. there is a case where an extent var, instead of being defined 43 (as this is how is defined in the NEW SHARED), is set to 34 in the shared var. P2J fails correctly... this looks like a typo in the program where lookup fails (is defined as SHARED with extent 34). I think one of these changes might have been made after switching to P2J runtime, and it wasn't tested in the legacy system.

Also, there are changes in the MAJIC srcnew/java code, where shared variables are used - I'll create a separate task for these.

Eric: You can start the Server conversion, the issues above will not affect it.

#30 - 12/21/2015 12:57 PM - Constantin Asofiei

Ovidiu/Eric: something else to discuss. If an external program instantiation fails (for example, because a shared var can not be found), then we need to clean up after any other NEW SHARED resources which were defined BEFORE the point of failure. Currently, these are saved in the SharedVariableManager$WorkArea.pending map, which gets processed when the next scope is started.

I've added code to ControlFlowOps to clean this pending map if external program instantiation fails, but I'm not sure what I need to explicitly clean for NEW SHARED BUFFER and NEW SHARED TEMP-TABLE. I don't think is enough to just "lose" the reference from the pending map, we might need to simulate a "delete" on these. Any ideas what I should keep an eye on, to ensure any "pending" buffer/temp-table is cleaned up? In BufferManager, RecordBuffer, etc?

#31 - 12/21/2015 01:24 PM - Constantin Asofiei

In regard to note 30 - an idea would be to simulate an empty execute() block, which will go through the scope start/end processing for all Scopeable implementations... this would save us doing things explicitly and will allow the resources to be "cleaned" naturally. Although some special handling might be required for NEW GLOBAL SHARED, as these are not saved in the "pending" map.

#32 - 12/21/2015 02:52 PM - Ovidiu Maxiniuc

Constantin Asofiei wrote:

> In regard to note 30 - an idea would be to simulate an empty execute() block, which will go through the scope start/end processing for all Scopeable implementations... this would save us doing things explicitly and will allow the resources to be "cleaned" naturally. Although some special handling might be required for NEW GLOBAL SHARED, as these are not saved in the "pending" map.

Indeed, that would be an idea. I believe that forcing an 'execute' would cost a little CPU, but since this happens only on exception cases, the cost is negligible overall.
I guess imagined the use of a temporary variable like this:

```
public void execute() {
    Block block;
    try {
        block = Block() { [...] };
    } catch (...) {
        block = emptyBodyBlockInstance;
    }
    externalProcedure(block);
}
```

Just a thought, we could split the creation of the variable into declaration and initialization in init() method and catch eventual initialization exceptions there and then handle them. The generated code wouldn't be affected.

#33 - 12/21/2015 03:00 PM - Constantin Asofiei

Ovidiu Maxiniuc wrote:

> Constantin Asofiei wrote:
>
> > In regard to note 30 - an idea would be to simulate an empty execute() block, which will go through the scope start/end processing for all Scopeable implementations... this would save us doing things explicitly and will allow the resources to be "cleaned" naturally. Although some special handling might be required for NEW GLOBAL SHARED, as these are not saved in the "pending" map.
>
> Indeed, that would be an idea. I believe that forcing an 'execute' would cost a little CPU, but since this happens only on exception cases, the cost is negligible overall.

I guess imagined the use of a temporary variable like this:

I don't want to complicate the converted code. I'm thinking just calling something like this, directly from SharedVariableManager (or ControlFlowOps, not sure yet):

```
externalProcedure(new Block(){});
```

and it just hit me... there is no "external program" instance, as instantiation failed. But it might work just pushing a new Object() as the "currently executing external program", to have a THIS-PROCEDURE surrogate. And also there might need some other state from ControlFlowOps to be pushed.

Another reason to do this is there might be state from non-shared temp-tables, buffers, etc which need to be cleaned up, and these are not explicitly tracked.

#34 - 12/21/2015 05:55 PM - Greg Shah

Code Review Task Branch 2795a Revision 10971

The changes are quite good.

Eric: Please review this ASAP.

1. Agent and SharedVariableManager need a history entries.

2. I prefer not to add child class references to BaseDataType in initializeDefaultExtent(). Can't we use a protected method that has a default implementating in BDT to provide the ctor and arg?

3. Several of the SharedVariableManager.addTempTable() variants are missing javadoc.

#35 - 12/21/2015 11:08 PM - Eric Faulhaber

Code review 2795a/10971:

In addition to Greg's comments above, please note the following:

Where you check for the TEMP_TABLE token type in TRPL rules, please check for WORK_TABLE as well, since this older syntax supports shared tables.

I am a little concerned about promoting all shared temp-tables to instance fields. Please make sure this does not invalidate the assumptions in buffer_name_conflicts.rules, particularly w.r.t. my changes in rev 10851.

Can a persist.* wildcard import statement be used in SharedVariableManager?

#36 - 12/21/2015 11:30 PM - Eric Faulhaber

I ran the cut-down search unit tests. Every API invoked produced the following error in the server and client logs (reported as SEVERE in the server.log):

```
Invalid or inappropriate handle value given to DELETE OBJECT or DELETE PROCEDURE statement. (5425)
```

However, all the tests passed, so perhaps this is an issue during cleanup which does not affect the XML output and/or temp-table results analyzed by the tests.

There is no stack trace or other useful information in the logs. I have not run any of the larger test sets yet.

#37 - 12/22/2015 07:58 AM - Constantin Asofiei

Eric Faulhaber wrote:

> Where you check for the TEMP_TABLE token type in TRPL rules, please check for WORK_TABLE as well, since this older syntax supports shared tables.

Thanks, only record_scoping.rules needed a check for WORK_TABLE. buffer_definition.rules already handles this with the etype var.

> I am a little concerned about promoting all shared temp-tables to instance fields. Please make sure this does not invalidate the assumptions in buffer_name_conflicts.rules, particularly w.r.t. my changes in rev 10851.

I think the trunk P2J already converts shared buffers/temp-tables to instance fields. More, temp-tables (shared or not) and shared buffers can be defined only in the external program, they can not be defined in procedures/functions. I've checked what happens in P2J if a buffer with the same name as a shared one is defined in an internal procedure, and in the converted code is disambiguated to a dedicate var scoped to the internal procedure. So I don't think there are issues related to name conflicts.

> Can a persist.* wildcard import statement be used in SharedVariableManager?

Yes, I've fixed it.

#38 - 12/22/2015 08:48 AM - Constantin Asofiei

Greg Shah wrote:

> 1. Agent and SharedVariableManager need a history entries.

Fixed in 10975

> 2. I prefer not to add child class references to BaseDataType in initializeDefaultExtent(). Can't we use a protected method that has a default implementating in BDT to provide the ctor and arg?

See 10973 - I've added a BDT.instantiateDefaultExtent() which gets the default value for extent vars. It is overridden as needed, otherwise it just returns the variables' default initialization value.

> 3. Several of the SharedVariableManager.addTempTable() variants are missing javadoc.

Fixed in 10975.

So, what is left as of rev 10975:

- resource cleanup (shared and non-shared) if external program instantiation fails
- the handle-related message Eric posted in note 36

#39 - 12/22/2015 10:14 AM - Constantin Asofiei

I've rebased 2795a from trunk rev 10961 (new rev 10976).

Rev 10977 fixes another NPE related to shared frame check (widget format). Runtime testing restarted.

#40 - 12/22/2015 11:42 AM - Constantin Asofiei

2795a rev 10978 adds logging to HandleOps.invalidDelete(handle). To enable it, add this node to directory.xml's logging/loggers section:

```
            <node class="string" name="com.goldencode.p2j.util.HandleOps">
```

```
            <node-attribute name="value" value="FINE"/>
          </node>
```

#41 - 12/22/2015 03:43 PM - Constantin Asofiei

2795a rev 10979 adds instantiateDefaultExtent() to recid.

#42 - 12/23/2015 11:39 AM - Eric Faulhaber

Constantin Asofiei wrote:

> 2795a rev 10978 adds logging to HandleOps.invalidDelete(handle). To enable it, add this node to directory.xml's logging/loggers section:
> [...]

I've enabled the logging and run the cut-down search tests with 2795a/10979. All tests pass. Server log is attached to #1868-77.

#43 - 12/25/2015 12:59 PM - Eric Faulhaber

I've run the full search tests. Functionally, the results are pretty good (other than the inappropriate handle issue reported in note 36 above, which seems to occur for every API call). However, there is a severe performance regression. The run normally takes 80-90 minutes, but with 2795a/10979, it took more than 10 hours. I don't know if the regression is in the 2795a branch, or in the 2332a branch, which was merged to trunk rev. 10961, because I didn't run the unit tests for that branch myself. I'll re-run the tests with trunk 10961 to further isolate the issue.

I did a bit of profiling while the tests were running, and we seem to be spending an inordinate amount of time in the Text.deepAssign method, invoked when new character objects are constructed by DMOs. Since the code for that method was not changed in this branch, it seems that something changed to cause the DMO getter methods to be invoked far more often than before. I haven't analyzed beyond that yet.

We had no new test failures, however there were 6 new test timeout errors. Results and logs are posted in #1868-78.

#44 - 12/26/2015 03:15 PM - Eric Faulhaber

Eric Faulhaber wrote:

> I'll re-run the tests with trunk 10961 to further isolate the issue.

This run was normal (~75 minutes), so the performance regression is in the changes specific to the 2795a branch. Not sure if it's related to the inappropriate handle error, but I suspect not, since it seems to be related to DMO use.

#45 - 12/29/2015 03:54 AM - Constantin Asofiei

I think the reason is related to this issue: all buffers created for a temp-table must be deleted if the temp-table gets deleted.

Also, there is another issue: if a NEW GLOBAL SHARED is resolved to an existing temp-table, then it must create another buffer for it, and not re-use the default-buffer for the temp-table.

#46 - 12/30/2015 07:08 AM - Constantin Asofiei

2795a rev 10981 fixes the temp-table buffer delete (see testcases/uast/procedures/shared_tt_run.p). There is a conversion change, so reconversion is required for any further testing. The main rule is: if a temp-table gets deleted, all its associated buffers must be deleted, too.

There is another issue related to the buffer order reported by the SESSION:FIRST-BUFFER list - currently, P2J adds buffers to this list in their "creation order". But I don't think this is correct: for the temp-tables, the order looks like is given by the temp-table creation order and only after that, by the buffer creation order. I don't plan to fix this in the scope of this task.

Next I'm checking how buffers for permanent tables are deleted.

#47 - 12/30/2015 08:32 AM - Constantin Asofiei

2795a rev 10981 looks like is working OK with the permanent buffers, too - see testcases/uast/procedures/shared_pt_run.p. But there is the same problem - the buffer order in the SESSION:FIRST-BUFFER list is incorrect. Again, P2J reports them following the creation order, while 4GL uses some other order... not sure why, as all the buffers are for the same table.

I'm starting MAJIC testing with rev 10981 next.

Eric: please run the testing again (don't forget to reconvert). Note that I haven't fixed explicitly the performance issue yet... the only conversion/runtime change which might affect it is that I'm setting the TemporaryBuffer.global flag for NEW SHARED GLOBAL temp-tables, but I couldn't duplicate. I couldn't find a reason why this is happening.

#48 - 12/31/2015 03:03 PM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric: please run the testing again (don't forget to reconvert). Note that I haven't fixed explicitly the performance issue yet... the only conversion/runtime change which might affect it is that I'm setting the TemporaryBuffer.global flag for NEW SHARED GLOBAL temp-tables, but I couldn't duplicate. I couldn't find a reason why this is happening.

The changes between rev 10979 and 10981 looked OK to me, and I've run this through the full search test again.

The good news is that the performance issue seems to be resolved; I'm guessing this was due the expense of processing the inappropriate handle error.

The bad news is that we had regressions: 29 tests failed, when normally only 6 should. I have not analyzed the results; however, they are posted at #1868-79. The 2 timeout errors are not related to this issue.

#49 - 12/31/2015 03:10 PM - Eric Faulhaber

Eric Faulhaber wrote:

> The bad news is that we had regressions: 29 tests failed, when normally only 6 should.

I am not sure I was working with a clean database for this run, so I am restoring the database and re-running this test set.

#50 - 01/01/2016 03:51 PM - Eric Faulhaber

The full search test re-run showed no regressions (only the expected 6 failed tests), so it seems my previous run was due to using an altered database instance.

The running time of both of the last two runs was at the high end of what I normally see for this test set, so there might a slight performance degradation, but nothing drastic.

I'll run the remaining unit test sets and report any regressions.

#51 - 01/03/2016 02:24 PM - Eric Faulhaber

The common search and the system maintenance CRUD test sets both completed with no new regressions. The developer test set hung (an intermittent problem not related to this issue), so I am running it again.

#52 - 01/04/2016 12:48 PM - Eric Faulhaber

There seems to be one test failing that wasn't before, in the developer test set:

```
com.something.calendarMaintenance.RenjhgjhgjgjgMaintenanceTest.cSearchCalendarYearCurrent
```

I've posted the full results to #1868-80.

I also tried running this test in single mode. The result was the same: #1868-81.

I'm going to try running this with the current P2J trunk version to see if I can eliminate task branch 2795a as the cause of the regression.

LE: GES modified the package and class name above to remove customer-specific references.

#53 - 01/04/2016 01:10 PM - Eric Faulhaber

Eric Faulhaber wrote:

> I'm going to try running this with the current P2J trunk version to see if I can eliminate task branch 2795a as the cause of the regression.

Confirmed that we have the same error in the trunk (rev. 10961), so 2795a/10981 is not the cause. If this branch has passed regular regression testing, please merge to trunk.

#54 - 01/05/2016 07:58 AM - Constantin Asofiei

2795a rev 10982 adds support for FRAME:BORDER- attributes - this was required as the fixes in this branch exposed show-stopper issues in the GUI tests.

#55 - 01/06/2016 06:17 AM - Constantin Asofiei

Greg, if 2795a rev 10981/10982 look OK to you, 2795a can be released.

#56 - 01/06/2016 11:30 AM - Greg Shah

Code Review Task Branch 2795a Revision 10982

The changes look good.

#57 - 01/06/2016 11:35 AM - Greg Shah

Some questions:

1. The issue noted about NEW GLOBAL SHARED temp tables not reusing existing buffers seems fixed in SVM.addTempTable(), correct?

2. You mentioned an issue with the TemporaryBuffer.global flag. Is this still an issue?

3. If we merge this to trunk, I guess that the 454 testcases will have the following regressions until 2875a is merged:

- screen flickering on every click
- a visible scroll-bar issue

Am I understanding correctly?

#58 - 01/06/2016 11:54 AM - Constantin Asofiei

Greg Shah wrote:

> Some questions:
>
> 1. The issue noted about NEW GLOBAL SHARED temp tables not reusing existing buffers seems fixed in SVM.addTempTable(), correct?

Correct. If temp-table is already defined (global or not), a subsequent NEW GLOBAL SHARED will create a new slave buffer for that temp-table; the master buffer is created only on the very first definition of the temp-table, and not re-used.

> 2. You mentioned an issue with the TemporaryBuffer.global flag. Is this still an issue?

No, is no longer an issue: GLOBAL SHARED temp-tables survive until end of context (and this flag is used for dynamic temp-tables, too), plus the javadoc in TemporaryBuffer.makeBuffer states that global - true if the buffer survives until end of context life; false if it expires upon leaving the top level procedure scope..

3. If we merge this to trunk, I guess that the 454 testcases will have the following regressions until 2875a is merged:

   - screen flickering on every click
   - a visible scroll-bar issue

   Am I understanding correctly?

With trunk + 2795a we will not be able to get past the User Defined Alerts window, as the window will freeze after the alert box is dismissed (as documented in #2272-406). After 2875a is merged, the two issues you mention will appear, and also the alert-box related to dialog-frame width remains (related to User Defined Alerts window), but will no longer freeze the window. So the missing UI issue (from your list) is the unexpected alert-box related to dialog-frame width.

#59 - 01/08/2016 04:20 AM - Constantin Asofiei

rebased branch 2795a from trunk rev 10962. new rev 10983.

#60 - 01/20/2016 11:39 AM - Greg Shah

At this point, I think all the issues this exposed in the customer application are resolved in the trunk (when 2875a was merged).

Can you please:

1. rebase to the latest trunk revision
2. confirm that the code no longer exposes regressions in the customer app
3. merge these changes into 2647a where they will be included in a full regression testing cycle

#61 - 01/20/2016 11:57 AM - Constantin Asofiei

1. rebase to the latest trunk revision

Rebased branch 2795a from trunk rev 10964. New rev 10985

A reminder that #2945 contains MAJIC source code changes and baseline changes required by this task. So, for runtime testing, the updates attached at #2945 need to be used.

#62 - 01/20/2016 12:14 PM - Constantin Asofiei

Hynek, previously I had this code in FrameGuiImpl:

```
         if (cfg.dialog)
         {
            FrameWindowGuiImpl dialogWindow = WindowManager.findWindow(wcfg.windowID);

            titleHeight = dialogWindow.getTitleBar().height();
         }
```

After rebase, there is no more FrameWindowGuiImpl class - please tell me which class replaces this. I need to find the dialog's title-bar height.

#63 - 01/20/2016 04:08 PM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek, previously I had this code in FrameGuiImpl:
> [...]
>
> After rebase, there is no more FrameWindowGuiImpl class - please tell me which class replaces this. I need to find the dialog's title-bar height.

The new class name is DialogBoxWindow.

#64 - 01/21/2016 04:13 AM - Constantin Asofiei

Hynek Cihlar wrote:

> The new class name is DialogBoxWindow.

Thanks, this solves the compilation problem (committed to rev 10986) but unfortunately with trunk + 2795a the GUI project fails after test selection.

The problem I'm seeing is this, after selecting the test: for a dialog frame, this code in FrameGuiImpl.initialize:

```
      if (wcfg.dialog && wcfg.windowID == -1)
      {
         // if no dialog window has been created, create it here
         DialogBoxWindow dialogWindow = new DialogBoxWindow(wcfg.title);

         // realize early, the window may be needed even before
```

```
         // it is shown for the first time
         dialogWindow.realize();

         // assign the window id so it is attached in super.initialize()
         wcfg.windowID = dialogWindow.getId().asInt();
      }

      super.initialize(id, wcfg);
```

sets the wcfg.windowID correctly, but after super.initialize(id,wcfg) is called, the wcfg.windowID gets set to 1 (the ID of the default window). Later in initialize, when this line is executed:

```
            DialogBoxWindow dialogWindow = WindowManager.findWindow(wcfg.windowID);
```

I get a ClassCastException as the wcfg.windowID is set to 1...

Can you please take a look? I've placed the GUI jars and sources (so that you don't need to reconvert with 2795a) into devsrv01:/tmp/2795a.

#65 - 01/21/2016 08:54 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > The new class name is DialogBoxWindow.
>
> Thanks, this solves the compilation problem (committed to rev 10986) but unfortunately with trunk + 2795a the GUI project fails after test selection.
>
> The problem I'm seeing is this, after selecting the test: for a dialog frame, this code in FrameGuiImpl.initialize:
> [...]
>
> sets the wcfg.windowID correctly, but after super.initialize(id,wcfg) is called, the wcfg.windowID gets set to 1 (the ID of the default window). Later in initialize, when this line is executed:
> [...]
> I get a ClassCastException as the wcfg.windowID is set to 1...

FrameConfig.windowID in case of a dialog box is used for (a) specifying the id of the dialog window the frame will be attached to during initialize (in Frame.attachToWindow()) and (b) after the frame has been attached to indicate the "logical" parent (so that FRAME:WINDOW attribute returns the correct value , default window handle is expected in this case). I agree this is a bit schizophrenic and certainly a subject for improvement.

I think the solution is to use Frame.topLevelWindow() to get the actual window owner instead of WindowManager.findWindow(cfg.windowId) since cfg.windowId after Frame.initialize() holds the logical window.

#66 - 01/21/2016 09:38 AM - Constantin Asofiei

Hynek Cihlar wrote:

> I think the solution is to use Frame.topLevelWindow() to get the actual window owner instead of WindowManager.findWindow(cfg.windowId) since cfg.windowId after Frame.initialize() holds the logical window.

OK, this fixes the issue and now the client gets to the User Defined Alerts window and from this to the main test window.

Greg: the following issues are still there for the User Defined Alerts window:

- a visible scroll-bar issue (this is a regression, introduced by trunk rev 10963 I think - the window mngmt improvements)
- unexpected alert-box related to dialog-frame width

#67 - 01/21/2016 09:52 AM - Greg Shah

Hynek: It was my understanding that both of those issues were already resolved in 2875a.

#68 - 01/22/2016 04:55 AM - Hynek Cihlar

Greg Shah wrote:

> Hynek: It was my understanding that both of those issues were already resolved in 2875a.

Ad the visible scroll-bar: I was able to reproduce it and the fix in 2875a worked ok. I will check in 2795a.

Ad the unexpected alert-box. Sorry I must have overlooked this one, there is no such fix for this in 2875a. I will fix this as soon as I finish the above.

#69 - 01/22/2016 04:43 PM - Hynek Cihlar

Hynek Cihlar wrote:

> Greg Shah wrote:
>
> > Hynek: It was my understanding that both of those issues were already resolved in 2875a.
>
> Ad the visible scroll-bar: I was able to reproduce it and the fix in 2875a worked ok. I will check in 2795a.

This seems to be a frame layout issue. The scroll container shows the scroll bars because there is a visible skip frame item spanning larger than the visible size. The question is why is the skip there or why is it visible. This is still a WIP.

> Ad the unexpected alert-box. Sorry I must have overlooked this one, there is no such fix for this in 2875a. I will fix this as soon as I finish the above.

Frames in dialog-box have zero border, even with NO-BOX is not specified. This particular case was caused by Frame.canSetWidthChars() which validates the new size against a minimum size. The minimum size is increased to account for the border. So we ended up with incorrect minimum size and hence the alert-box. I have fixed this particular case, but there is more code like this that needs to be fixed. And btw the minimum size was also being calculated wrong for regular GUI frames, it just never appeared.

#70 - 01/23/2016 07:21 PM - Hynek Cihlar

Hynek Cihlar wrote:

> Hynek Cihlar wrote:
>
> > Greg Shah wrote:
> >
> > > Hynek: It was my understanding that both of those issues were already resolved in 2875a.
> >
> > Ad the visible scroll-bar: I was able to reproduce it and the fix in 2875a worked ok. I will check in 2795a.
>
> This seems to be a frame layout issue. The scroll container shows the scroll bars because there is a visible skip frame item spanning larger than the visible size. The question is why is the skip there or why is it visible. This is still a WIP.

Ok, this is not a layout issue but the missing runtime support for SCROLLABLE attribute. The frame contains some additional items outside of the visible area but is assigned SCROLLABLE to FALSE and hence no bars are shown.

This issue should be resolved once 2795a is merged with 2038b which adds the support for SCROLLABLE attribute.

#71 - 01/25/2016 09:11 AM - Greg Shah

> I have fixed this particular case, but there is more code like this that needs to be fixed.

You have fixed this in 2038b?

How much more code needs to be fixed? Do you propose to fix these places now?

#72 - 01/25/2016 10:27 AM - Hynek Cihlar

Greg Shah wrote:

> > I have fixed this particular case, but there is more code like this that needs to be fixed.
>
> You have fixed this in 2038b?

Yes I have fixed this particular case in 2038b.

> How much more code needs to be fixed? Do you propose to fix these places now?

I checked all the places where wrong box/border size is assumed for dialog-box. Some are obvious and I will fix these. Some are used in ChUI and possibly by GUI as well. The ChUI cases are risky and would require more testing, a wild guess plus 2 MD to implement and regression test.

#73 - 01/25/2016 10:29 AM - Hynek Cihlar

Note that the unexpected scroll bars issue in User Defined Alerts window is solved by #2038, task branch 2038b.

#74 - 01/25/2016 01:26 PM - Greg Shah

Constantin: Please rebase from the latest trunk and put this through regression testing.

If it passes testing, please merge the changes in 2795a into 2647a. Eric will do the final testing of that branch and in that branch is how it will merge to trunk.

#75 - 01/25/2016 01:42 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: Please rebase from the latest trunk and put this through regression testing.

Rebased from trunk rev 10965. new rev 10988.

Full testing started.

#76 - 01/26/2016 03:41 PM - Constantin Asofiei

Rebased 2795a from trunk rev ~~10965~~ 10966. new rev 10989

testing again to clean some false negatives.

#77 - 01/27/2016 07:43 AM - Constantin Asofiei

The only issue is with tc_item_master_007 - two records are ordered differently in the screen on step 80. Running in single mode doesn't help (as a certain DB state is required).

Anyway, I don't think this is related to 2795a changes.

Eric: please let me know if I can merge into 2647a.

#78 - 01/27/2016 09:00 AM - Eric Faulhaber

Constantin Asofiei wrote:

> Eric: please let me know if I can merge into 2647a.

Yes, please do.

#79 - 01/27/2016 09:17 AM - Constantin Asofiei

Eric, I have this conflict in recid.java:

```
   @Override
   public BaseDataType instantiateDefault()
```

```
   {
<<<<<<< TREE
      return new recid(0);
=======
      return new recid();
   }
....
>>>>>>> MERGE-SOURCE
   }
```

Are you sure instantiateDefault() needs to initialize the var with 0 instead of unknown? I ask this because converting a recid var initializes it to unknown (new recid()), and also in 4GL an uninitialized recid var is reported as unknown.

#80 - 01/27/2016 09:48 AM - Constantin Asofiei

Eric, I've merged 2795a into 2647a - see revision 10983.

With this revision, the following is needed:

1. for MAJIC testing - baseline and MAJIC updates from #2945
2. all projects require re-conversion

#81 - 01/27/2016 09:56 AM - Eric Faulhaber

Constantin Asofiei wrote:

> Are you sure instantiateDefault() needs to initialize the var with 0 instead of unknown? I ask this because converting a recid var initializes it to unknown (new recid()), and also in 4GL an uninitialized recid var is reported as unknown.

Sorry I wasn't able to answer in time. I was trying to find the code which led me to believe this, but I can't at the moment. I was going to suggest that you go ahead with your version. If I conclude it should be 0 after all, I'll change it back and I'll post the reason here.

#82 - 02/01/2016 08:12 AM - Greg Shah

Merged to trunk as part of revision 10967.

We can close this task, right?

#83 - 02/01/2016 08:13 AM - Constantin Asofiei

Greg Shah wrote:

We can close this task, right?

Correct

#84 - 02/01/2016 08:23 AM - Greg Shah

- Status changed from WIP to Closed

Please archive the task branch.

#85 - 11/16/2016 12:12 PM - Greg Shah

- Target version changed from Milestone 12 to GUI Support for a Complex ADM2 App