Base Language - Feature #1607

implement full named events support

10/19/2012 10:06 AM - Greg Shah

Status:	Closed	**Start date:**	01/29/2013
Priority:	Normal	**Due date:**	07/01/2013
Assignee:		**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Runtime Support for Server Features		
billable:	No	**version:**	
vendor_id:	GCD		

Description

Subtasks:

Feature # 1976: conversion support for named events	**Closed**
Feature # 1977: implement the runtime support for named events	**Closed**

Related issues:

Related to Base Language - Feature #3253: add cross-session publish/subscribe	**WIP**	
Blocked by Base Language - Feature #1606: implement persistent procedures/sup...	**Closed**	**11/09/2012**

History

#1 - 10/19/2012 10:10 AM - Greg Shah

The following support is needed: PUBLISH, SUBSCRIBE, UNSUBSCRIBE, parameter passing processing, chaining, super-procedure lookup. Because this is dependent upon super-procedures, this task is dependent upon #1606.

Note that there is some early support for PUBLISH in rules/convert/language_statements.rules and there is a stubbed out start of the runtime support in com/goldencode/p2j/util/NamedEventManager.java.

#2 - 10/31/2012 01:47 PM - Greg Shah

- Target version set to Milestone 7

#3 - 01/15/2013 11:58 AM - Greg Shah

Named events also support the PUBLISHED-EVENTS attribute, which the documentation suggests is a string created at compile-time rather than at runtime. This is a lower priority than the 3 main language statements.

#4 - 01/15/2013 12:02 PM - Greg Shah

The 4GL documentation suggests that the following are implicit uses of THIS-PROCEDURE:

PUBLISH without a FROM clause
SUBSCRIBE without a PROCEDURE clause
UNSUBSCRIBE without a PROCEDURE clause

Constantin: based on the state of the procedure handle work, should we plan to implement a lookup of THIS-PROCEDURE inside the NamedEventsManager runtime code? Or should we expect to make the THIS-PROCEDURE explicit in the converted application code?

#5 - 01/15/2013 12:41 PM - Constantin Asofiei

I don't see any valid reason to emit an explicit ProcedureManager.thisProcedure() call in these cases. The runtime should see that there is no handle reference and default to the handle returned by ProcedureManager.thisProcedure() call.

#6 - 01/15/2013 12:51 PM - Greg Shah

The ProcedureManager will have enough information to know the immediate caller (in the converted code) and associate that with the correct procedure handle?

#7 - 01/15/2013 01:28 PM - Constantin Asofiei

Yes. As THIS-PROCEDURE needs to return the external procedure handle to which the current statement belongs, I've implemented this way:
1. via scope notifications, each time an external procedure, internal procedure or function is started, i push the external-procedure instance to which it belongs to a stack (and remove it when it ends)
2. ProcedureManager.thisProcedure returns the top of this stack
3. computing the object instance at #1 is done using the closest Block instance, and searching up the this$<#> fields until this$0 is found (although Java compiler emits this$ fields as package protected, the java.lang.reflect.Field.setAccessible(true) makes it possible to get the value of such non-public fields).

#8 - 01/29/2013 01:07 PM - Constantin Asofiei

Looking at how the PUBLISH and SUBSCRIBE/USUBSCRIBE work, I think these will need to be kept at the external program instance, together with the other ProcedureManager.PersistentProc data. This, and combined with the fact that it integrates with the procedure handle internals, I think it would be best to add the following APIs to ProcedureManager:

- subscribeEvent
- publishEvent

The PROCEDURE subscriber-handle clause for SUBSCRIBE/UNSUBSCRIBE might make you think that this should emit as an API call for the subscriber-handle instance, but I think it would be best to emit this as a parameter for our static APIs.

About PUBLISHED-EVENTS - the documentation states that this attribute is collected at compile-time, so I think this can be solved at conversion time, which can save it as an attribute in name_map.xml, for the procedure.

#9 - 01/29/2013 01:18 PM - Greg Shah

```
add the following APIs to ProcedureManager:

    subscribeEvent
    publishEvent
```

1. Would these be used directly from the converted code or is this a statement about runtime internals?
2. What about unsubscribe?

In regards to your other comments, I agree.

#10 - 01/29/2013 01:29 PM - Constantin Asofiei

1. Would these be used directly from the converted code or is this a statement about runtime internals?

I don't think we need to add a separate class for these events, so yes, they will be calld directly from converted code.

2. What about unsubscribe?

For unsubscribe, we can have a ProcedureManager.unsubscribeEvent API.

#11 - 01/30/2013 07:50 AM - Constantin Asofiei

After looking at what was already implemented in the conversion rules and stubed in the runtime, my conclusions are these:

1. use NamedEventManager as a placeholder for the APIs associated for PUBLISH, SUBSCRIBE and UNSUBSCRIBE. APIs defined in this class will be emitted by the conversion code.
2. each API in NamedEventManager will finally delegate the work to only a few methods in ProcedureManager (not exposed to the public). This way, we keep ProcedureManager clean (there is no need to add all the 24 API versions required for these statements there).
3. there are existing conversion rules which emit the THIS-PROCEDURE for PUBLISH without a FROM clause. I think the best approach is to make the subscriber handle required in our APIs. (this will help us a lot in not letting the APIs in NamedEventManager "explode", as we will need to manage all parameter combinations).So, I've changed conversion rules to emit the THIS-PROCEDURE as the subscriber for the "SUBSCRIBE without a PROCEDURE clause" and "UNSUBSCRIBE without a PROCEDURE clause" cases too.

At this time, the conversion rules are complete, except that I need to fix a case when the IN keyword preceded by a variable reference is treated as if it was part of a "var:<attribute> IN frame" clause, and not for the "IN publisher" clause. The problem I think is at the parser, I will solve this and post the sources for review.

#12 - 01/30/2013 09:06 AM - Constantin Asofiei

- File ca_upd20130130a.zip added

- File ca_upd20130130b.zip added

Sources are built on top of their bazaar versions as of today.
Testcases will be committed to bzr shortly.
I'll do a conversion regression testing today.

#13 - 01/30/2013 11:30 AM - Constantin Asofiei

- File ca_upd20130130c.zip added

Added merged sources.

#14 - 01/30/2013 12:19 PM - Constantin Asofiei

A note: for the PUBLISH statement, the mode for each parameter should be collected and emitted the same as is emitted for the RUN statement.

#15 - 01/30/2013 12:49 PM - Greg Shah

I'm fine with the changes, they look good.

Go ahead with conversion testing. We will bypass runtime testing for this one. It should be safe enough to include with testing for other changes.

#16 - 01/30/2013 12:52 PM - Constantin Asofiei

- File ca_upd20130130g.zip added

Added rules to emit the parameter modes, for the PUBLISH statement. I will leave the conversion to run on my machine, I'll have the results in the morning.

#17 - 01/30/2013 01:00 PM - Greg Shah

Good stuff. If it passes conversion testing, we will figure out the check in. I will find out if there are any conflicts with what is being regression tested in staging right now.

#18 - 01/30/2013 02:39 PM - Greg Shah

There are no conflicts with the stuff in staging. When this passes conversion testing, you can check it in and distribute it. We will have to coordinate applying it to staging to avoid their testing.

#19 - 01/31/2013 08:17 AM - Constantin Asofiei

Passed conversion regression testing, applied to bzr revision 10158. Not applied to staging.

#20 - 09/25/2013 08:19 AM - Constantin Asofiei

The publish/subscribe/unsubscribe rules from the documentation are simple, but practice proves to be more complex.

The following code behaves really weird (and applies only to UNSUBSCRIBE TO ALL ... IN cases, regardless of the publisher handle:

```
procedure proc0.
   message "foo".
end.

procedure proc1.
   message "bar".
end.

message "unsubscribe?" update l as log.

subscribe to "ev1" anywhere run-procedure "proc0".

if l then unsubscribe to all in this-procedure.

subscribe to "ev1" anywhere run-procedure "proc1".

publish "ev1" from this-procedure.
```

If UNSUBSCRIBE is not used, then the second SUBSCRIBE is a no-op (as there is already a subscription for the ev1 event and ANYWHERE clause); when PUBLISH is called in this case, only proc0 is executed. But, if the UNSUBSCRIBE is used, then the first SUBSCRIBE is somehow "hidden" as the second SUBSCRIBE manages to register itself; when PUBLISH is called in this case, both proc0 and proc1 are executed.

This doesn't make any sense, because:

1. if UNSUBSCRIBE ... IN is used, then this shouldn't touch SUBSCRIBE ... ANYWHERE at all. Documentation states that a SUBSCRIBE ANYWHERE can be removed only by UNSUBSCRIBE without IN clause.
2. if UNSUBSCRIBE without IN clause is used, then the second subscribe succeeds and only proc1 is executed.

More, if we add an UNSUBSCRIBE without IN clause after a UNSUBSCRIBE ... IN clause:

```
procedure proc0.
   message "foo".
end.

procedure proc1.
   message "bar".
end.

message "unsubscribe?" update l as log.

subscribe to "ev1" anywhere run-procedure "proc0".

if l then unsubscribe to all in this-procedure.

subscribe to "ev1" anywhere run-procedure "proc1".

unsubscribe to all.

publish "ev1" from this-procedure.
```

when UNSUBSCRIBE ... IN is executed, the first subscribe survives the last UNSUBSCRIBE TO ALL statement - for some reason it doesn't remove it.

I think the conclusion is that the first UNSUBSCRIBE TO ALL IN handle call marks as "read-only" all found ANYWHERE subscriptions, instead of removing them.

#21 - 09/25/2013 09:08 AM - Greg Shah

It seems that every new feature, even straightforward ones, will always have some significant divergence from the Progress documentation.

I guess this just means that you will have to do additional testcases...

#22 - 09/25/2013 10:36 AM - Constantin Asofiei

- File ca_upd20130925a.zip added

A first version of the runtime for named events. I've also added support for the PUBLISHED-EVENTS attribute for procedure handles. Basically, the code is in these ProcedureManager methods:

1. publish - for PUBLISH statement
2. subscribe - for SUBSCRIBE and SUBSCRIBE ANYWHERE
3. unsubscribe - for UNSUBSCRIBE and UNSUBSCRIBE ALL

Beside more testing of the named events when multiple procedures are involved, there is this feature I need to check: PUBLISH statement uses an implicit NO-ERROR clause, but this clause is used only for the statement's parameters and the determined to-be-invoked procedure-related validation. When the procedure code is executed, the NO-ERROR clause doesn't look like is in effect any more, as the ERROR condition is not raised at the caller too, after the procedure has finished.

#23 - 09/25/2013 10:44 AM - Greg Shah

> When the procedure code is executed, the NO-ERROR clause doesn't look like is in effect any more, as the ERROR condition is not raised at the caller too, after the procedure has finished.

If I understand correctly, this sounds like the behavior you get when you RUN proc.p (whatever) NO-ERROR. We have special support (in TransactionManager/BlockManager) for how this works.

#24 - 09/25/2013 10:46 AM - Constantin Asofiei

Greg Shah wrote:

> > When the procedure code is executed, the NO-ERROR clause doesn't look like is in effect any more, as the ERROR condition is not raised at the caller too, after the procedure has finished.
>
> If I understand correctly, this sounds like the behavior you get when you RUN proc.p (whatever) NO-ERROR. We have special support (in TransactionManager/BlockManager) for how this works.

Forgot to mention, the problem is that error messages (the ones which set error-status:error to true in case of statements with NO-ERROR clause) are displayed at the terminal.

#25 - 09/25/2013 11:09 AM - Greg Shah

Code Review 0925a

1. Why are only persistent procedures checked for subscriptions? The ProcedureManager code uses PersistentProc and the ppMap. The Progress docs (at least for the SUBSCRIBE stmt) do not list a limitation that the registered procedure must be persistent. I can imagine a scenario where a procedure that is known to be on the call stack subscribes to an event that will be published by something deeper in the call tree. It seems that the subscribing procedure in that case would exist and be callable even if it was not persistent.

2. In unsubscribe(), perhaps the comment // event is not validated should be changed to // Progress does not validate the event?

#26 - 09/25/2013 11:15 AM - Constantin Asofiei

> 1. Why are only persistent procedures checked for subscriptions? The ProcedureManager code uses PersistentProc and the ppMap.

In ProcedureManager, all procedures are saved in the ppMap and deleted, via WorkArea's Scopeable implementation. If a procedure instance needs to live after it finished, it will get registered with the WorkArea.persistentProcedures map (which actually holds all the persistent procedures created via RUN PERSISTENT). If the procedure is in this map, it will not be removed from ppMap when WorkArea.scopeFinished is called.

> 2. In unsubscribe(), perhaps the comment // event is not validated should be changed to // Progress does not validate the event?

Yes, your phrase is better.

#27 - 09/25/2013 11:48 AM - Greg Shah

> In ProcedureManager, all procedures are saved in the ppMap and deleted, via WorkArea's Scopeable implementation. If a procedure instance needs to live after it finished, it will get registered with the WorkArea.persistentProcedures map (which actually holds all the persistent procedures created via RUN PERSISTENT). If the procedure is in this map, it will not be removed from ppMap when WorkArea.scopeFinished is called.

Please put some javadoc into the class to explain the following:

1. On the ppMap member to explain that this map is all procedures, not just persistent procedures.
2. On the locateProcedure() method to explain that although this returns a PersistentProcedure instance, the method will find any procedure that is active on the stack or is active because it is persistent.

I think part of the problem is the name of the PersistentProc inner class, which I thought (because of the name) to be only representing persistent procedures. Perhaps it should be renamed? Something like ProcWrapper?

#28 - 09/26/2013 04:47 AM - Constantin Asofiei

- File ca_upd20130926b.zip added

- File named_events_tests_20130926c.zip added

Attached update is going through regression testing now.

Beside other fixes, I've renamed PersistentProc to ProcedureData (as this is more of a data container and not a wrapper) and ppMap member to pMap.

#29 - 09/26/2013 08:31 AM - Greg Shah

Code Review 0926b

I'm fine with the code. Before checking in, please make these comment changes:

1. This comment:

```
// walk through all the persistent procedures which are subscribed to this event
```

should be:

```
// walk through all the procedures which are subscribed to this event
```

2. Several javadoc link entries like this:

```
{@link WorkArea#ppMap persistent procedure}
```

should be:

```
{@link WorkArea#ppMap procedure}
```

#30 - 09/26/2013 09:47 AM - Constantin Asofiei

- File ca_upd20130926d.zip added

Attached version contains the comment changes at note 29. Runtime testing is still in progress - conversion testing has passed.

#31 - 09/26/2013 01:31 PM - Constantin Asofiei

The runtime testing seems OK, but I want to do another run, to clean some failures.

#32 - 09/27/2013 04:09 AM - Constantin Asofiei

ca_upd20130926d.zip was committed to revision 10388.

#33 - 09/27/2013 08:34 AM - Greg Shah

- Status changed from New to Closed

#34 - 11/16/2016 11:43 AM - Greg Shah

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

#35 - 02/24/2017 10:56 AM - Greg Shah

- Related to Feature #3253: add cross-session publish/subscribe added

Files

ca_upd20130130a.zip	19.4 KB	01/30/2013	Constantin Asofiei
ca_upd20130130b.zip	3.03 KB	01/30/2013	Constantin Asofiei
ca_upd20130130c.zip	19.6 KB	01/30/2013	Constantin Asofiei
ca_upd20130130g.zip	22.2 KB	01/30/2013	Constantin Asofiei
ca_upd20130925a.zip	62.2 KB	09/25/2013	Constantin Asofiei
ca_upd20130926b.zip	63 KB	09/26/2013	Constantin Asofiei
named_events_tests_20130926c.zip	13.5 KB	09/26/2013	Constantin Asofiei
ca_upd20130926d.zip	63 KB	09/26/2013	Constantin Asofiei