Base Language - Feature #2050

ATTR_POLY/METH_POLY used as rvalue or in expressions

02/25/2013 09:56 AM - Constantin Asofiei

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

Description

Subtasks:

Feature # 2051: conversion support for ATTR_POLY/METH_POLY used as rvalue or in expression	**Closed**

Related issues:

Related to Base Language - Feature #1584: add conversion and runtime support ...	**Closed**	**12/17/2012**	**05/10/2013**
Related to Base Language - Feature #2287: implement missing type conversion f...	**Closed**		
Related to Base Language - Bug #2300: :: operator used in UI statements	**New**		

History

#1 - 02/25/2013 10:02 AM - Constantin Asofiei

Conversion for ATTR_POLY used in complex expressions fails, as the ExpressionConversionWorker can't determine the type. i.e:

```
def var h as handle.
message "something" + h:buffer-field("field"):buffer-value.
```

Need to test with all type of expressions (logical, int, dec, date, etc), not just char concatenation.

More, the runtime needs to be fixed as in 4GL, following is valid (entire discussion at #1992, comments 18 to 23):
CA:

```
> def var i as int.
> function func0 returns char.
>    return "1000".
> end.

> /* i = func0(). this is compile-time error */
> i = dynamic-function("func0"). /* this executes and sets i to 1000 */
> message i.
> 
```

> Idea is, if we have the ATTR/METH_POLY in an expression or as a rvalue, then 4GL does some automatic conversion of the "poly" rvalue to the lvalue's expected type. I know that for DYNAMIC-FUNCTION we disambiguate the return type based on the function's real return type, but this case that the lvalue can be assigned a not-compatible type.

GES:

> It seems like we will have to do the following:
>
> 1. Test the limits/capabilities of the 4GL polymorhpic return transformations, especially in regard to assignments. I suspect there are real limits in what silent conversions are allowed.
> 2. Determine our exposure and plan for a resolution. My hope is that the 4GL has a standard set of conversions and we can implement those in each wrapper's constructor that takes a BaseDataType (which you recently added for dynamic-function return values). Then we can wrapper any assignment of a *_POLY rvalue with the proper constructor.

#2 - 02/25/2013 10:16 AM - Greg Shah

The problem I was working (in #1655) is the same problem. I "fixed" it by eliminating the use of expressionType() in 2 places in where_clause.rules. But that "fix" won't work everywhere. It did get me further in code but then I hit this ATTR_POLY issue again, deeper in the directory.

Here are the details:

```
.../supp/ocoreex0.p
EXPRESSION EXECUTION ERROR:
---------------------------
cls = expressionType(this)
      ^  { Unknown ATTR_POLY with oldtype of KW_BUF_VAL }
---------------------------
ERROR:
java.lang.RuntimeException: ERROR!  Active Rule:
-----------------------
      RULE REPORT
-----------------------
Rule Type :   WALK
Source AST:  [ IF ] BLOCK/PROCEDURE/BLOCK/INNER_BLOCK/BLOCK/STATEMENT/KW_IF/KW_ELSE/INNER_BLOCK/BLOCK/STATEMEN
T/KW_ASSIGN/ASSIGN/EXPRESSION/LPARENS/FUNC_POLY/ @2216:51 {97117800506153}
Copy AST  :  [ IF ] BLOCK/PROCEDURE/BLOCK/INNER_BLOCK/BLOCK/STATEMENT/KW_IF/KW_ELSE/INNER_BLOCK/BLOCK/STATEMEN
T/KW_ASSIGN/ASSIGN/EXPRESSION/LPARENS/FUNC_POLY/ @2216:51 {97117800506153}
Condition :  cls = expressionType(this)
Loop      :  false
--- END RULE REPORT ---

        at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:732)
        at com.goldencode.p2j.convert.ConversionDriver.processTrees(ConversionDriver.java:948)
        at com.goldencode.p2j.convert.ConversionDriver.back(ConversionDriver.java:852)
        at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1729)
Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:7 [FUNC_POLY id=971178005061
53]
        at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:226)
        at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:160)
        at com.goldencode.p2j.pattern.PatternEngine.apply(PatternEngine.java:1119)
        at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1017)
        at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:704)
        ... 3 more
Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:7
        at com.goldencode.expr.Expression.execute(Expression.java:430)
        at com.goldencode.p2j.pattern.Rule.apply(Rule.java:401)
        at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:640)
        at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:609)
        at com.goldencode.p2j.pattern.Rule.apply(Rule.java:440)
        at com.goldencode.p2j.pattern.Rule.executeActions(Rule.java:640)
        at com.goldencode.p2j.pattern.Rule.coreProcessing(Rule.java:609)
        at com.goldencode.p2j.pattern.Rule.apply(Rule.java:440)
        at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:531)
        at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:50)
        at com.goldencode.p2j.pattern.RuleContainer.apply(RuleContainer.java:531)
        at com.goldencode.p2j.pattern.RuleSet.apply(RuleSet.java:50)
        at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:213)
        ... 7 more
Caused by: java.lang.UnsupportedOperationException: Unknown ATTR_POLY with oldtype of KW_BUF_VAL
        at com.goldencode.p2j.convert.ExpressionConversionWorker.expressionType(ExpressionConversionWorker.jav
a:811)
        at com.goldencode.p2j.convert.ExpressionConversionWorker.expressionType(ExpressionConversionWorker.jav
a:317)
        at com.goldencode.p2j.convert.ExpressionConversionWorker.expressionType(ExpressionConversionWorker.jav
a:701)
        at com.goldencode.p2j.convert.ExpressionConversionWorker$ExpressionHelper.expressionType(ExpressionCon
versionWorker.java:1048)
        at com.goldencode.expr.CE10196.execute(Unknown Source)
        at com.goldencode.expr.Expression.execute(Expression.java:336)
        ... 19 more
```

#3 - 02/25/2013 10:20 AM - Constantin Asofiei

I wonder if ExpressionConversionWorker should ignore these _POLY cases ? And determine the expression type based on the other operands in the expression ? Problems would be only if there are only _POLY in the entire expression.

#4 - 02/25/2013 11:21 AM - Greg Shah

- Target version set to Milestone 7

The issue only occurs for those ATTR_POLY/FUNC_POLY/METH_POLY cases where we cannot detect the type based on the children/parameters.

Here are the possible *_POLY cases (there are no METH_POLY):

```
ABSOLUTE         - FUNC_POLY returns an INTEGER or INT64 or DECIMAL based directly on the parameter type

ACCUMULATE       - FUNC_POLY returns a type depending on the sub-type of the accumulator (KW_COUNT/KW_SUB_CNT 
are INTEGER, KW_AVERAGE is DECIMAL, KW_MAX/KW_SUB_MAX/KW_MIN/KW_SUB_MIN/KW_TOTAL/KW_SUB_TOT return the same ty
pe as the following expression)

ADD-INTERVAL     - FUNC_POLY returns a DATE, DATETIME or DATETIME-TZ  based directly on the parameter type

BUFFER-VALUE     - ATTR_POLY returns the same type as the field being referenced, but the field name is specif
ied in a runtime expression, plus there may be some morphing going on based on context

DEFAULT-VALUE    - ATTR_POLY not supported yet (and is not investigated either)

DYNAMIC-FUNCTION - FUNC_POLY returns the type based on the function that is called, which can change at runtim
e since the function name is an expression, this case needs context analysis (see #1920 and #1972)

DYNAMIC-INVOKE   - FUNC_POLY not supported yet (and is not investigated either)

GET-BYTES        - FUNC_POLY I believe this returns a RAW or MEMPTR but I have not tested for conditions yet

IF               - FUNC_POLY returns the same type as the 2nd operand (not the condition, but the value return
ed on true) EXCEPT that there is a natural widening that occurs if the 2nd operand is integer/int64 and the 3r
d operand is decimal (then this returns decimal) AND if the 2nd operand is the UNKNOWN literal then the type i
s defined by the 3rd operand

INPUT           - FUNC_POLY returns the same type as the widget referent EXCEPT that there are times when the 
4GL will morph this return value to a character, when the containing code requires it (we do some context anal
ysis already)

MAXIMUM         - FUNC_POLY returns a type based on the parameters, where parameter types can be intermixed th
ere can be widening results (e.g. mixed integer and decimal will return decimal)

MINIMUM         - FUNC_POLY same as MAXIMUM 

NORMALIZE       - FUNC_POLY not supported yet (and is not investigated either)

RETURN-VALUE    - ATTR_POLY not supported yet (and is not investigated either)

SUPER           - FUNC_POLY should return the same type as the containing function, not sure if there is any m
orphing

UNBOX           - FUNC_POLY not supported yet (and is not investigated either)
```

We need enhancements to expressionType() for the following: BUFFER-VALUE, DYNAMIC-FUNCTION, GET-BYTES, SUPER

I am going to investigate the behavior of these cases and look at the use cases in the current project to ensure we handle them all. Then I will attempt to implement context analysis of the parents/siblings to detect the target type in all of those cases. I am putting that logic into expressionType() and will try to use the same logic for all of these cases if possible.

#5 - 02/25/2013 07:48 PM - Greg Shah

- File ges_upd20130225b.zip added

- File buffer_value_context_analysis.p added

I have implemented major changes to ExpressionConversionWorker.expressionType() to do the following:

1. Add support for all *_POLY types which are supported in P2J today. That support may not be 100% but it is possibly (probably?) enough.

2. Take the fledgling context analysis that was previously done for INPUT (FUNC_POLY), made it a separate method and massively improved it to support a much wider range of features. I believe it supports the features needed by BUFFER-VALUE in this project, BUT I can't put my eyes on all 12000+ use cases. But the ones I found (the same idioms are used many times over) are supported.

3. Refactored to be cleaner and more reusable.

4. Fixed some bugs that I found while reading the code.

This resolves the failure in supp/ and I hope it should resolve the problem in raam/ (it is being tested now). In addition, I am about to run conversion regression testing on this. It should have no impact on Majic.

#6 - 02/25/2013 07:50 PM - Greg Shah

Yes, it does seem to resolve the problem in raam/. Now we get a different error (see #1985).

#7 - 02/25/2013 08:52 PM - Greg Shah

I believe this is passing conversion testing.

The following diffs are present:

```
1580c1580
<           return concat("INVALID SO #.  PLEASE ENTER SO # >= ", valueOf(fInputsFrame.getSCoNum()));
---
>           return concat("INVALID SO #.  PLEASE ENTER SO # >= ", valueOf(fInputsFrame.getScreenValue(fInputsFr
ame.widgetSCoNum())));
1608c1608
<           return concat("INVALID EMPLOYEE #.  PLEASE ENTER EMPLOYEE # >= ", valueOf(fInputsFrame.getSEmpNum()
));
---
>           return concat("INVALID EMPLOYEE #.  PLEASE ENTER EMPLOYEE # >= ", valueOf(fInputsFrame.getScreenVal
ue(fInputsFrame.widgetSEmpNum())));
1636c1636
<           return concat("INVALID DATE.  PLEASE ENTER DATE >= ", valueOf(fInputsFrame.getSDate()));
---
>           return concat("INVALID DATE.  PLEASE ENTER DATE >= ", valueOf(fInputsFrame.getScreenValue(fInputsFr
ame.widgetSDate())));
```

Somehow, the improved expression typing changes the INPUT FUNC_POLY processing. As far as I know, these are equivalent statements, so I think it is OK.

Constantin: what do you think?

I have checked this code into bzr as revision 10206. If this is a regression, then I will fix it tomorrow and check in another update.

#8 - 02/26/2013 03:42 AM - Constantin Asofiei

- Target version deleted (Milestone 7)

Somehow, the improved expression typing changes the INPUT FUNC_POLY processing. As far as I know, these are equivalent statements, so I think it is OK.
Constantin: what do you think?

getScreenValue and getter work the same as far as I can tell, after looking at the code.

#9 - 02/26/2013 10:53 AM - Greg Shah

From #1985:

ra/ folder fails with:

```
.../ra/echrgad3.p
WARNING: no type calculated for BUFFER-VALUE in [127813931767083] BUFFER-VALUE [ATTR_POLY] @1441:99

EXPRESSION EXECUTION ERROR:
---------------------------
op1 = ecw.expressionType(copy)
          ^  { PLUS cannot have op1 (null). }
---------------------------
ERROR:
java.lang.RuntimeException: ERROR!  Active Rule:
-----------------------
      RULE REPORT
-----------------------
Rule Type :   WALK
Source AST:  [ + ] BLOCK/PROCEDURE/BLOCK/STATEMENT/KW_IF/KW_THEN/INNER_BLOCK/BLOCK/STATEMENT/KW_ASSIGN/ASSIGN/
EXPRESSION/PLUS/PLUS/ @1443:28 {127813931767037}
Copy AST  :  [ + ] BLOCK/PROCEDURE/BLOCK/STATEMENT/KW_IF/KW_THEN/INNER_BLOCK/BLOCK/STATEMENT/KW_ASSIGN/ASSIGN/
EXPRESSION/PLUS/PLUS/ @1443:28 {127813931767037}
Condition :  op1 = ecw.expressionType(copy)
Loop      :  false
--- END RULE REPORT ---

        at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:732)
        at com.goldencode.p2j.convert.ConversionDriver.processTrees(ConversionDriver.java:948)
        at com.goldencode.p2j.convert.ConversionDriver.back(ConversionDriver.java:836)
        at com.goldencode.p2j.convert.ConversionDriver.main(ConversionDriver.java:1729)
Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:11 [PLUS id=127813931767037]
        at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:226)
        at com.goldencode.p2j.pattern.AstWalker.walk(AstWalker.java:160)
        at com.goldencode.p2j.pattern.PatternEngine.apply(PatternEngine.java:1119)
        at com.goldencode.p2j.pattern.PatternEngine.processAst(PatternEngine.java:1017)
        at com.goldencode.p2j.pattern.PatternEngine.run(PatternEngine.java:704)
        ... 3 more
Caused by: com.goldencode.expr.ExpressionException: Expression execution error @1:11
        at com.goldencode.expr.Expression.execute(Expression.java:430)
        at com.goldencode.p2j.pattern.Rule.apply(Rule.java:401)
```

code snippet:

```
      ASSIGN OcQUeryString = "WHERE rntpropchgs.pr-seq-no = " + STRING(prhst.pr-seq-no)
                           + " AND rntpropchgs.rnttrn-cde = " + QUOTER(IhInputTTBuf:BUFFER-FIELD("rnttrn-cde")
:BUFFER-VALUE)
```

```
                              + " AND rntpropchgs.str-dte = " + IhInputTTBuf:BUFFER-FIELD("str-dte"):BUFFER-VALUE
                              + " AND rntpropchgs.crt-dte = " + IhInputTTBuf:BUFFER-FIELD("crt-dte"):BUFFER-VALUE
                              + " AND rntpropchgs.crt-tim = " + IhInputTTBuf:BUFFER-FIELD("crt-tim"):BUFFER-VALUE
 NO-ERROR.
```

#10 - 02/26/2013 10:57 AM - Greg Shah

- File ges_upd20130226a.zip added

- File buffer_value_context_analysis.p added

The issue is caused by the over-simplistic recursion protection in our context analysis. In addition, even in the case where we really cannot recurse safely, it is possible that we could still detect some better outcomes in the PLUS operator using only the 2nd operand.

I have resolved both of these issues in the attached update (the testcase was also enhanced to show the new problem). The recursion processing is now tracked by node ID, so the same node ID can never be entered twice. This refinement may not be the end of what can be done, but it is a good step forward.

This is in conversion regression testing now.

#11 - 02/26/2013 12:13 PM - Greg Shah

Passed conversion testing and is checked in as bzr revision 10211. Runtime recursion testing is deferred.

#12 - 02/27/2013 06:13 PM - Greg Shah

A new context analysis problem has been found in #1985 (see note 63). The 2nd operand of the PLUS is a FIELD_DEC type. The ExpressionConversionWorker code for PLUS (in both expressionType() and contextAnalysis()) needs to be reviewed as the possible source of the issue.

#13 - 03/01/2013 03:29 PM - Greg Shah

- File buffer_value_context_analysis.p added

- File buffer_value_dynamic_plus.p added

- File ges_upd20130301a.zip added

Attached is the fix (and testcase updates) for this case. Based on testing in the 4GL, one can indeed have the return type of the plus operator determined dynamically at runtime. In the case above, context analysis cannot be done. So the solution is to detect this case and emit a DynamicOps.plus(BaseDataType,BaseDataType) call that will accept any wrapped parameter, determine the right downstream plus() to call, call it and return the result. Right now it is just stubbed.

This is being conversion tested right now.

#14 - 03/01/2013 05:17 PM - Greg Shah

- File buffer_value_context_analysis.p added

- File ges_upd20130301b.zip added

This passed testing, but in further processing of the customer code, another (virtually identical) NPE was encountered. This time it looks like this:

```
h:BUFFER-FIELD("field"):BUFFER-VALUE = h:BUFFER-FIELD("field"):BUFFER-VALUE - h:BUFFER-FIELD("other-field"):BU
FFER-VALUE.
```

I made further updates to handle the MINUS operator dynamically. This is being conversion tested right now. The latest update (and testcase) is attached.

#15 - 03/01/2013 08:15 PM - Greg Shah

Passed conversion testing and checked in as bzr revision 10230.

#16 - 03/04/2013 03:32 PM - Greg Shah

- File ges_upd20130304a.zip added

#1985 note 96 shows a new problem: DB_REF_NON_STATIC needs to be supported. I have added it to the ECW. It uses context analysis to try to determine the type, then falls back to "BaseDataType" if nothing is found. Also as part of the attached update is a separate buffer promotion fix for BUFFER-COPY/BUFFER-COMPARE which fixes note 103 of #1985. Both fixes resolve the reported issues and they are going into conversion testing now.

#17 - 03/04/2013 05:04 PM - Greg Shah

- File ges_upd20130304b.zip added

New version of the update with another change to the parser that fixes a problem found in testing the server code. It isn't worth creating a new task for it. Restarting conversion testing now.

#18 - 03/04/2013 06:46 PM - Greg Shah

Passed conversion testing and checked in to bzr as revision 10239.

#19 - 03/09/2013 07:28 AM - Constantin Asofiei

The ExpressionConversion worker needs to treat the cases when a _POLY is used in a logical test, as in:

```
def var h as handle.

if h:buffer-value then message "a".

if not h:buffer-value then message "a".

if h:buffer-value or not h:buffer-value then message "a".

if h:buffer-value and not h:buffer-value then message "a".
```

#20 - 03/09/2013 11:13 AM - Greg Shah

The strange thing is that I already put in support for all of these cases. I guess it is not working properly. I will look into it.

#21 - 03/09/2013 03:34 PM - Greg Shah

I'm not sure the problem is in the ECW. Actually, there is never any attempt to call the ECW for these cases. If I understand your concern, it is because the results in these cases are not cast to a logical type?

#22 - 03/09/2013 03:52 PM - Constantin Asofiei

Hmm... on a second thought, this is similar to the DMO setter case... it should wrap the _POLY/DB_NON_REF in a logical instance, because this is what it expects.

#23 - 03/09/2013 03:56 PM - Greg Shah

Yes, I'm thinking about what is the right solution here. I wonder if I should make the previous solution (the extra boolean flag for expressionType() used in the DMO setter solution) more general purpose. The idea is that anything that will return a BaseDataType will be reported as BaseDataType instead of having the type inferred. I'm working on it now. Let me know if you have any concerns.

#24 - 05/15/2013 11:26 AM - Ovidiu Maxiniuc

This is related to my note 116 from #1584.
After a talk with Constantin we have draw the conclusion that the fix for the issue will be automatically be done by correct implementation for _POLY cases. The effects were observed and studied using date/datetime/datetimetz but I'm positive all other datatypes behave the same.
As documentation say, the above mentioned datatypes are essentially uncomparable directly. You must first convert them to the same data type before. This is evident using static references of those values. However, when obtaining the values from dynamic-function or other source of _POLY values things apparently change.

```
FUNCTION getDate RETURN DATE: RETURN 12/31/2050. END.
FUNCTION getDateTime RETURN DATETIME: RETURN 2050-12-31T10:30:21.555. END.
FUNCTION getDateTimeTz RETURN DATETIME-TZ: RETURN 2050-12-31T10:30:21.555-11:00. END.

DISPLAY DYNAMIC-FUNCTION("getDate") EQ 12/31/2050 /* prints yes, evidently */
        DYNAMIC-FUNCTION("getDateTime") EQ 12/31/2050 /* prints yes */
        DYNAMIC-FUNCTION("getDateTimeTz") EQ 12/31/2050 /* prints yes */.

DISPLAY 2050-12-31T10:30:21.555 GT DYNAMIC-FUNCTION("getDate")  /* prints yes */
        2050-12-31T10:30:21.555 EQ DYNAMIC-FUNCTION("getDateTime") /* prints yes, evidently */
        2050-12-31T10:30:21.555 LT DYNAMIC-FUNCTION("getDateTimeTz") /* prints yes */.

/* DISPLAY 2050-12-31T10:30:21.555 GT 12/31/2050. -- won't compile */

DISPLAY DYNAMIC-FUNCTION("getDate") EQ  DYNAMIC-FUNCTION("getDate") /* prints yes, evidently */
        DYNAMIC-FUNCTION("getDateTime") LT  DYNAMIC-FUNCTION("getDate") /* fails dynamically */ NO-ERROR.
```

We can draw the conclusions:

- comparing statically a date and datetime literals is impossible without casting one of them. We can compare only compatible types.
- if one operator has a static value whose type is known at compile time and the other is a _POLY value than Progress silently inserts a casting function to the dynamic one so the runtime value will be converted to the known type (if possible/if the appropriate conversion is available). This is why 2nd and 3rd expression are first converted to DATE (dropping time information) prints "yes" and the 4th in which the 2nd operand will use 0 mtime after conversion to datetime.
- if both operands are obtained dynamically (_POLY values) then no conversion is performed at compile nor at runtime, if the values are static compatible they are processed as in 1st case, otherwise the "Incompatible data types in expression or assignment. (223)" error will break the execution.

#25 - 05/22/2014 04:58 PM - Greg Shah

As far as I know, only the issue in note 24 remains open for this task.

I believe that most of the problem in this case will be solved by implementing the _POLY constructors in #2287. However, the conversion may still need to be updated to properly detect and wrap the DYNAMIC-FUNCTION return value in the right constructor such that the _POLY code can be triggered. I believe we already mostly handle this (the ExpressionConversionWorker is heavily involved in this type/wrapping determination), but we may not handle all cases noted above.

#26 - 05/28/2014 12:51 PM - Greg Shah

- Target version set to Milestone 11

- Assignee set to Marius Gligor

#27 - 06/16/2014 03:02 AM - Marius Gligor

- Status changed from New to WIP

#28 - 06/25/2014 02:41 AM - Marius Gligor

- Status changed from WIP to Review

Passed MAJIC regression tests. Passed the customer's server conversion test.
Committed revision 10551.

#29 - 06/25/2014 08:56 AM - Greg Shah

- Status changed from Review to Closed

#30 - 11/16/2016 12:07 PM - Greg Shah

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

Files

ges_upd20130225b.zip	53.6 KB	02/26/2013	Greg Shah
buffer_value_context_analysis.p	367 Bytes	02/26/2013	Greg Shah
ges_upd20130226a.zip	24 KB	02/26/2013	Greg Shah
buffer_value_context_analysis.p	816 Bytes	02/26/2013	Greg Shah
ges_upd20130301a.zip	18.4 KB	03/01/2013	Greg Shah
buffer_value_context_analysis.p	938 Bytes	03/01/2013	Greg Shah
buffer_value_dynamic_plus.p	1.06 KB	03/01/2013	Greg Shah
ges_upd20130301b.zip	18.5 KB	03/01/2013	Greg Shah
buffer_value_context_analysis.p	1.04 KB	03/01/2013	Greg Shah
ges_upd20130304a.zip	279 KB	03/04/2013	Greg Shah
ges_upd20130304b.zip	279 KB	03/04/2013	Greg Shah