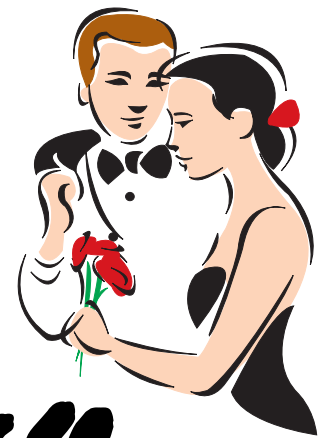


# RPG and Java, The Perfect Marriage



By George Farr

## Introduction

As I go around the country and the world talking to AS/400 RPG IV and COBOL programmers, I consistently get the same message. People are nervous and worried about the future of the AS/400 and in particular RPG. Here are some of the most common questions I get from you, our AS/400 customers: “Is RPG IV here to stay, or is IBM trying to kill it?” “V4R5 does not have any RPG IV enhancements, is that a signal from IBM that no more enhancements are planned for RPG?” “Is IBM turning the AS/400 into a Java machine to replace RPG?” “Is Java the only option for writing e-Business applications on the AS/400?” These are but a few of the questions AS/400 programmers keep asking. Let me address them here in this article.

Yes, Java is a great new language that works very well for e-Business applications and incorporates many great technologies. However, RPG is still the workhorse of the AS/400 that many businesses are happily and successfully working with. IBM is *not* trying to “kill RPG” or ceasing investment in RPG. Neither should you, unless you see a very clear business reason to jump completely to Java (if multiple platform support is required, for example). RPG is the robust, productive and efficient business language that brought the AS/400 to where it is today. Our strategy at IBM is to enable each language do what

it is best at, and enable you to mix and match the usage of each language in the context most appropriate for it. RPG is a great business language that works well and integrates well with the AS/400. Java is a good e-Business language for extending a business application to the Internet, a good object oriented language and a good multi-platform language. Making these two languages co-exist and easily integrate is our main goal.

The real power of e-business for existing AS/400 customers will only be fully realized with the successful marriage of these two great languages, which will require using Java to connect Web and GUI user interfaces to RPG IV based business logic.

Now back to the questions at hand, instead of answering these questions upfront and making your life easy, I would like to answer them by giving you a sneak preview of what’s coming in RPG IV in the next release. My intent is to wet your appetite on the new functionality that is planned for RPG IV as well as have you answer some of the above questions yourself.

## RPG IV prior to V4R5!

Before discussing what’s coming in the next release of RPG IV (i.e. in Version 5 release 1) Let’s look back at the enhancements and investment that IBM have done over the years since the inception of RPG IV. In V3R1 we had seen



the biggest enhancements ever done to RPG IV. The compiler had been totally restructured and rewritten to introduce the new language definition. Then came V3R2/V3R6, which introduced two major new enhancements. The first was "Procedures" and the second was signed and unsigned Integer data type. On came V3R7 where many more enhancements were incorporated into the language, these included:

- Floating point data type
- Conditional compilation and pointer arithmetic
- New built-in functions
- Long Names and Null field support

To continue the tradition of more enhancements to RPG IV, we introduced the following in V4R2:

- New indicator Data type
- Varying length character fields
- Compiler options on the H Specification

At this point as we were planning the V4R4 release (We skip shipped V4R3), Java was gaining more and more popularity and momentum in the industry. So, we turned focus a bit to making sure that Java and RPG interoperate well together.

Finally, we did not want to stop the tradition of more enhancements to RPG IV, and so we brought to you the following enhancements:

- New opcodes: EVALR, LEAVESR and Free form FOR loop
- OVERLAY(\*NEXT)
- More built in functions

And on we marched forward beyond V4R4...

## RPG IV V4R5 and beyond!

The latest release of OS/400 is V4R5, which came out in August 2000. This release of the operating system, while significant in other respects, is not an application development (AD) release. All AD software including all compilers and tools, were not updated. In IBM parlance, this was a "skip ship" for us. This allows us to concentrate on the next release, which you will see is very significant to AD. In fact, it will be the single most significant release for AD software since at least V3R1, and perhaps ever for the AS/400.



*George Farr*

The following is a subset of the enhancements that you may expect to see in that release.

- New built in functions such as: %CHECKR, %XLATE, %LOOKUP, %SCANR, and many more
- MONITOR Operation group
- Free Form Calculation Specification
- Runtime control of file open
- ELSEIF Operation code
- Qualified names in data structures
- Increased Java related support

The rest of this article will discuss some of these enhancements and give you some examples to illustrate the function. I would like your feedback on these and other enhancements you would like to see happen to RPG IV. If you would like to contribute, please contact me. I am discussing this out in the open to make sure that IBM is doing exactly what you need to run your

business with. Obviously, all what is about to be discussed here is still in a design and implementation phase, so some aspects may end up being changed, removed, deferred or totally replaced before the release is finalized.

Also, before I describe these enhancements, please remember the questions I discussed earlier in this article. I am positive that they will be all answered in your own mind by the time you have read this article. Enjoy!

C* col.18	28	33	43
C*			
C* Factor 1	op-code	Factor 2	Result field
C* Occurrence Value	OCCUR	Data Structure	Occurance Value
C*			
C 6	OCCUR	DataStruct1	

*Figure 1*

First of all, since Java is a threaded language and RPG is not, we made sure that RPG IV is thread safe and therefore Java can call RPG without any potential problems. Next, we looked at all the data types that Java has and we made sure that RPG IV is compatible with that. The only missing data type was 1 and 8 byte integers and UCS-2 (Unicode). So we added that support in V4R4.

You may have heard about some of the other major changes and enhancements coming in the areas of WebSphere tooling and compiler and tool packaging ([www.ibm.com/software/ad/as400](http://www.ibm.com/software/ad/as400)). However you may not have heard what the plans are for RPG IV in the V5R1 release coming in first half 2000. The good news is, RPG IV will enjoy its biggest release ever!

## New built in functions

Since we introduced built-in functions, (or BIFs for short) in the first release of RPG IV, they became very popular with RPG programmers for their simplicity and ease of use. Over the last many releases we have introduced BIFs in many areas. One of the main advantages of BIFs is

→ the ability to use them in free form expressions. In V5R1 expect to see many more new BIFs that provide functionality within free form expressions that was previously available only in fixed form calculation specifications.

For example, take the OCCUR operation code. Currently the fixed format of the opcode is as shown in **Figure 1**. Factor 1 contains the numeric value to be used to set the occurrence of the data structure specified in factor 2. The result field is optional and if it is specified, it will be assigned the value of the multi-occurring data structure specified in factor 2. Our example above is showing how we set the occurrence of the multi occurring data structure, “DataStruct1” to 6. The new format for a BIF to represent the OCCUR opcode will be as shown in the example in **Figure 2**.

As you see in the example, we start by specifying the %OCCUR BIF on the left hand side of the EVAL operation code to cause it to change the current occurrence of “DataStruct1” to 10. The second EVAL operation above causes the %OCCUR BIF to return the current occurrence number of the specified data structure. In our case above the value returned will be 10.

```

* . 1 .. + .. 2 .. + .. 3 .. + .. 4 .. + .. 5 .. + .. 6
*
C*
C          EVAL      %OCCUR(DataStruct1) = 10
C          EVAL      OCC = %OCCUR(DataStruct1)
C* OCC = 10

```

**Figure 2**

This is just one example of an existing operation code that can be used in its fixed format style and also as a BIF in a middle of an expression. I won’t cover all of the BIFs here, but hopefully you can appreciate their ease-of-use value. Our intention is to enable ever more free-form capability into the language, while still retaining the fixed-form heritage to give you time to evolve your coding style. Ultimately, this means every fixed format opcode will support a BIF alternative.

To this end, listed below are some other BIFs that you may expect to see in the next release:

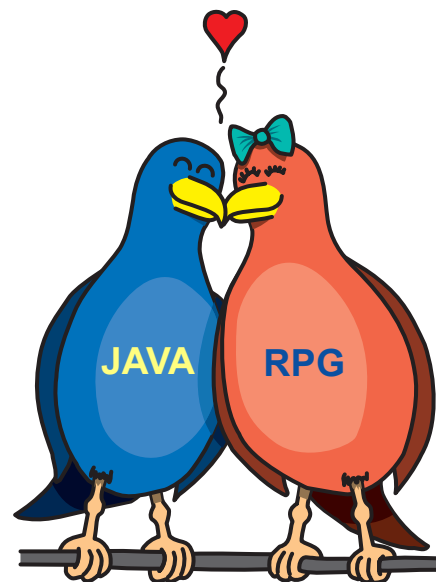
- %ALLOC
- %CHECK
- %CHECKR
- %DAYS
- %DIFF
- %HOURS
- %HOURS
- %LOOKUPXX
- %LOOKUPXX
- %MINUTES
- %MONTHS
- %MSECONDS
- %MSECONDS
- %REALLOC
- %REALLOC
- %SECONDS
- %SHTDN
- %SQRT
- %SQRT
- %TLOOKUPXX
- %XLATE
- %YEARS
- %YEARS

### MONITOR Operation Group

As you know, in RPG there are two ways that programs can handle exceptions:

- Using the famous error indicator or ‘E’ Extender that allows the handling of pre-defined set of exceptions.
- Using the ‘Program Error Subroutine’ (\*PSSR) or File error subroutine (INFSR).

These two methods serve their purpose well. However, when an error occurs and a branch is done to an error subroutine,



The MONITOR block has the code that is to be monitored for. In the example shown in **Figure 3**, you see the READ operation code being placed in the Monitor group. The On-Error blocks test for certain status codes or a range of status codes. If any one test evaluates to true (an exception does occur), the On-Error block is executed and the error is handled programmatically. If you have used Java, this may look familiar to you! This structure is similar to the **try** and **catch** clause in Java. Of course, it is also similar in functionality to the MONMSG statement in CL.

### Free Form Calculation Specification

RPG has historically been a fixed format language, with information entering in explicit column ranges. Over time, however, RPG IV has been evolving and introducing ever more free format syntax. For example, the C spec now supports eight opcodes that use an extended factor 2 in which free-form expressions can be coded using natural operators such as “=” for assignment and all the usual math and logical operators. As well, these expressions allow the use of built-in functions in place of field names, and even calls to user-defined functions (procedures). Further, with the V4R2 introduction of feedback built-in functions such as %ERROR and %EOF, you can even avoid using resulting

there is no way to return to the next instruction short of coding a label and a GOTO to it (assuming the programmer knows where the exception occurred). To enable more control of exception handling in RPG IV, the MONITOR operation code (Or group) is added. It consist of the following:

- A MONITOR block,
- One or more ON-ERROR blocks
- An ENDMON operation(Or END opcode)

```

C      MONITOR
C      READ          MASTER
C      IFNOT%EOF
C*     DO Something
C      ENDIF
C      On-Error      1211:1011
C*     Handle "closed file" error and
C*     "unknown record format" error
C      On-Error *FILE
C*     Handle other file errors
C      On-Error *PGM
C*     Handle errors processing fields
C      ENDMON

```

**Figure 3**

indicators, and hence resulting indicator columns. The introduction of the MONITOR statement in V5R1 continues this evolution.

In other specifications such as the Definition and File you also see partially free form specs, with the use of a free form keyword area in columns 44 to 80. Indeed, the H spec is totally free form allowing keywords anywhere in columns 8 though 80.

Over the past years, as programmers used the free format parts of RPG IV, they got to see the benefit of it and based on informal polling we constantly do, most have grown to like it. It is, arguably, easier to code, easier to read, and easier to teach to new programmers. By continuing to support the fixed format heritage of RPG, we allow to move to this new style at your own pace, and even on a programmer by programmer basis.

As a natural next step in that evolution, we are introducing in this release a total free form C Specification as you see it in the **Figure 4**. The way it will work is as follows: The free form calculation Specification (or CF as you see it in the example) is totally free format variation of the fixed form. An valid expression can be specified anywhere from column 8 to 80. Of course, the traditional fixed-form C-spec will continue to be supported.

As you see from the example, it is easier to read and indent your code in the free format version. As you may very well expect, there are few rules, (like

everything else in life,) that you need to follow. For example, the first thing that must be coded after CF is the actual opcode. The compiler will expect this. Each opcode has a new free-format syntax that takes its operands on the right. Even though we tried to support all op-codes, in some cases we could not. For example, op-codes that require

resulting indicators are not allowed. However, some opcodes that we could not support already have existing built-in functions

(example: %SUBST), and for others, there are new built-in functions (example: %LOOKUP).

### Runtime Control of File Open

Today to control the exact file to be opened, RPG programmers have to issue an override command in a CL program or have to call QCMDXEC from their RPG program. Have you ever wanted to open a file at runtime dynamically without calling a CL program to do the override? In a world where RPG logic might be called directly from a Web or GUI client, this becomes more important. We offer help in V5R1. With the introduction of two new keywords, namely, EXTFILE and EXTMBR this task is made a lot easier for you. →

*Things to do When Mid-Range is Your Business Partner: #14*

# TAKE A BATH IN A HOCKEY POOL

It may be the only thing you have to keep track of.  
 At Mid-Range we're experts a keeping your iSeries 400 – AS/400 operating at peak performance.  
 From application software recommendations / hardware upgrades and performance tuning through logical partitioning, operational support /education and disaster recovery we have what you need.  
 So you have more time to concentrate on your business.  
 An advantage you can really bet on.

**MID-RANGE** 

*Working For Your Peace of Mind*

Call: 1-800-668-6470 [www.midrange.ca](http://www.midrange.ca)

```

CF read file
CF dow not %eof(file)
CF  if %error
CF    dsply 'Error occurred'
CF    leave
CF  else
CF    chain:n name database data
CF    EVAL time = hours *num_employees + overtime
CF*
CF*    *****DO SOMETHING*****
CF*
CF  endif
CF enddo

```

**Figure 4**

→ The parameters to these new keywords can be constants, but better yet, they can be variable names to indicate the file name and the member name respectively to be opened.

do ask great questions! Say, we have the following override in effect before calling our RPG program:

```
OVRDBF PRODUCTION FARR/INVENTORY
```

```

FMyFile  IF  F 10  DISK  EXTFILE('FARR/MASTER')
F        EXTMBR('*ALL')
Fproduct IF  F 10  DISK  EXTFILE(YourFile)
F        EXTMBR('*ALL')
C*
C          OPEN  MyFile
C* -----
C          EVAL  YourFile = 'FARR/PRODUCTION'
C          OPEN  YourFile

```

**Figure 5**

Lets see how this all work by looking at the example in **Figure 5**. Here the first F Specification declares the file “MyFile” and initializes it to “FARR/MASTER” as part of the EXTFILE parameter. The EXTMBR keyword indicates that all members will be opened when an OPEN on the file “MyFile” is executed. So, in the C Specification when we actually open the file “MyFile”, the actual file used will be all the members in file “FARR/MASTER”.

On the other hand, the second file we declare uses a field name as a parameter for the EXTFILE keyword. This as you see in the code will enable us to initialize the field to whatever file we want to open at run time and then explicitly open it. Great, huh! What if we have overrides in effect, you may ask! Well, first of all you

This will change what gets opened when we execute the statement “OPEN YourFile”. Since the field “YourFile” contains the value “FARR/PRODUCTION” and since PRODUCTION had an override in effect, the actual file that gets opened is “FARR/INVENTORY”.

### ELSEIF operation code

The opcode name gives it away, no need to explain it further! Well, in **Figure 6**, we show a little program fragment that uses the ELSEIF opcode. As you see in the example, the ELSEIF opcode is a combination of the IF and the ELSE opcodes. This opcode eliminates the need for a new nesting level and hence a new END-IF statement.

For small structures, the ELSEIF operation makes the code a lot easier to maintain and read. Of course, you can still use the traditional ELSE and IF statements and the SELECT/WHEN/OTHER combinations as usual. There is no performance benefit using one over the other, so it is your call.

### Qualified Names in Data Structures

Did you ever wonder why in RPG you are not allowed to declare same names of Data Structure subfields in multiple Data Structures? We are sure you always wondered why other high level languages allow it and RPG doesn't! Wait no longer! The next release of RPG IV will have “Qualified names” supported.

What does that mean? Well, a new keyword, QUALIFIED, will be added to indicate that Data Structure subfield names must be qualified by the “parent” data structure name when used anywhere in the program. The way you would reference a subfield in a qualified data structure is as follows:

```
ParentDS.ChildSubField
```

In addition, we are adding another keyword, LIKEDS, to allow you to define a Data Structure to be the same as another one in your program. Lets take a look at an example of how this would end up working out. The example in **Figure 7** shows three data structures, Customer, Supplier, SecondSupplier. All data structures have subfields with identical names, namely: Name and Address. As you see in the code we use the qualified data structure name format to initialize all data structures. Also note that “SecondSupplier” data structure is identical to “Supplier”. Couldn't be simpler! Hope you like it! You are welcome!

### Increased Java related support

As we close on V4R4, we can all be very proud at the “water cooler” when we talk to other programmers about RPG Data Types.



As an RPG programmer, you can lift your head up and proudly say that RPG has as many, if not more data types than Java does.

In addition, we wanted to continue our strategy of making RPG IV and Java interoperate well together. After all, the future is a mix of Java with RPG. This release is another one that will see a great Java related enhancement to RPG. That is to introduce an easy way, other than JNI, to call a Java class from an RPG program using RPG syntax. Why not have RPG IV call a Java class and invoke a Java method by simply doing a CALLP? Well, watch what you say because your wish might just come true in V5R1. This enhancement to RPG prototypes will enable programmers to indicate that procedures are actually Java methods! In turn, when a call is made using these prototypes, the compiler will internally generate call to the JNI. (We call it internal plumbing that RPG programmers should not worry about.)

The prototypes will also enable RPG to handle parameters correctly according to Java conventions. After all, RPG at this point has absolutely all data types that Java has. So life is easier now to mix both languages.

As always, the easiest way to explain this enhancement is by giving you small example.

```
CF IF      Day = 'Monday'
CF CALLP   MondayRoutine
CF ELSEIF  Day = 'Tuesday' or Day = 'wednesday'
CF CALLP   Midweek
CF ELSEIF  Day = 'Thursday' or Day = 'Friday'
CF CALLP   Endweek
CF ELSE
CF CALLP   WeekEnd
CF ENDIF
```

**Figure 6**

The **Figure 8** illustrates this...

As you see in the example we first declare the method to be used, in this case the “add” method in the “java.math.BigDecimal” package. The next line declares the constructor for that class. Think of the constructor as your \*INZSR subroutine that is called at the start of initialization. We declare three additional Objects of type “java.math.BigDecimal”. These all are of type “O”, your guessed it, “O” stands for the newly introduced type of Objects in RPG. The first 2 EVAL statements call the constructor methods to create the numeric values with the specified initial values. The last EVAL calls the add method in the BigDecimal class to do the actual addition. If you are not familiar with Java this may seem complicated.

However, believe it or not this is as simple as it gets to have RPG call Java. However, if this syntax looks a little daunting, look for a wizard inside the CODE/400 editor to generate it for you after you pick the class and the method you wish to call.

### Why All This?

In this rapidly changing world, I believe there is a strong continued role for RPG as a business logic language. However, to continue to realize that role, RPG has to evolve to accommodate new technologies such as Java. It also has to evolve to be ever more productive to code and maintain, and it has to evolve so it is appealing to the new generations of programmers while not abandoning the existing generations. →

```
*. 1 .. + .. 2 .. + .. 3 .. + .. 4 .. + .. 5 .. + .. 6
*
D Customer      DS           Qualified
D  Name          20A
D  Address       50A
D Supplier      DS           Qualified
D  Name          20A
D  Address       50A
D SecondSupplier DS         LIKEDS(Supplier)
D*-----
C*
C           Eval   Customer.Name = 'Barbara Morris'
C           Eval   Customer.Address = ' 2 Myway Drive'
C           Eval   Supplier.Name = 'Hans Boldt'
C           Eval   Supplier.Address='1 MywayTheBestway Blvd'
C           Eval   SecondSupplier.Name ='George Farr'
C           Eval   SecondSupplier.Address='3 ThisIsTheWay Ave'
```

**Figure 7**

→ In an e-Business role, it has to have less dependency on CL, and more flexibility to work in a world where snippets of code are called to do a fine-grained job, versus a large monolithic process. Last but not least, it has to continue to address your requirements and wish-list items which you continue to feed us each release. It is our hope that we on such a path with the enhancements RPG IV has enjoyed so far and which are coming in the next and subsequent releases.

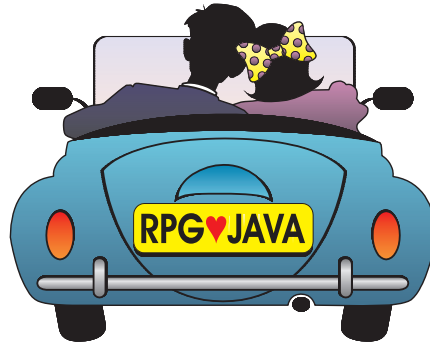
## Thank you is in order!

My Hat is off to my great RPG team, lead by Barbara Morris and Hans Boldt who contributed a great deal to making this release and many previous releases happen for RPG IV. I am sure Hans and Barbara will start giving you more of the details on these enhancements by writing additional articles if you all “Behave yourselves!” So, watch out for more!


## Final words...

If you like what you see so far, then tell your parents, friends, co-workers, cousins, in laws, and neighbors about it! If you don't, I suggest you remain silent and nothing will be held against you!

Kidding aside, I believe that RPG IV is, as always, the best business language that integrates very well with the iSeries and AS/400. It's proven year over year to be an excellent business language and a workhorse that had many successful businesses rely on it and succeed. IBM is not about to let that go.



In addition, with the Internet and e-Business we have a new form of business applications that are forming shape. Java has been proven to be the glue language for the Internet and e-Business. For those ready, willing and able to make the jump, it has also become a proficient language for business logic as well. With the introduction and maturation of Enterprise

Java Beans, this is even more true. IBM is betting heavily on Java. But rest assured, IBM also continues to bet heavily on RPG. You can be guaranteed that our Application Development strategy on the iSeries will be built toward the success and integration of both Java and RPG IV combined together. I believe that this upcoming release of RPG IV is the biggest release ever since the inception of RPG! I do hope that you agree with me after giving you this sneak preview of what you might expect to see coming to an AS/400 near you in V5R1 

**George Farr** works at the IBM Toronto laboratory, as the technical development manager for the RPG and VisualAge for RPG languages. George is a frequent speaker at COMMON and other conferences and user groups worldwide. He is also involved in education for RPG IV, Java, and VisualAge at IBM. George recently co-authored (with Phil Coulthard), the books **Java for RPG Programmers** and **Java for Cobol Programmers**. You can reach George at [farr@ca.ibm.com](mailto:farr@ca.ibm.com).

```
*. 1 .. + .. 2 .. + .. 3 .. + .. 4 .. + .. 5 .. + .. 6 .. + .. 7 .. + .. 8
*
D BigDecimal      C          'java.math.BigDecimal'
D add              PR          0  EXTPROC(*Java: BigDecimal: 'add')
D otherNum        0          CLASS(*Java: BigDecimal)
D                  CONST
D ConstructNum    PR          0  EXTPROC(*Java: BigDecimal: *CONSTRUCTOR)
D GetValue        PR          0  EXTPROC(*Java: BigDecimal: 'toString')
D                  CLASS(*Java: 'java.lang.String')
D getLongValue    PR          20I 0 EXTPROC(*Java: 'java.lang.Long': 'valueOf')
D                  STATIC
D string          0          CLASS(*Java: 'java.lang.String')
D                  CONST
D Num1            S          0  CLASS(*Java: BigDecimal)
D Num2            S          0  CLASS(*Java: BigDecimal)
D Total          S          0  CLASS(*Java: BigDecimal)
D longInteger     S          20I 0
D*-----
C                  EVAL      Num1 = ConstNum(225)
C                  EVAL      Num2 = ConstNum(175)
C                  EVAL      total = add(Num1: Num2)
C                  EVAL      longInteger = getLongValue (GetValue (total))
```

Figure 8