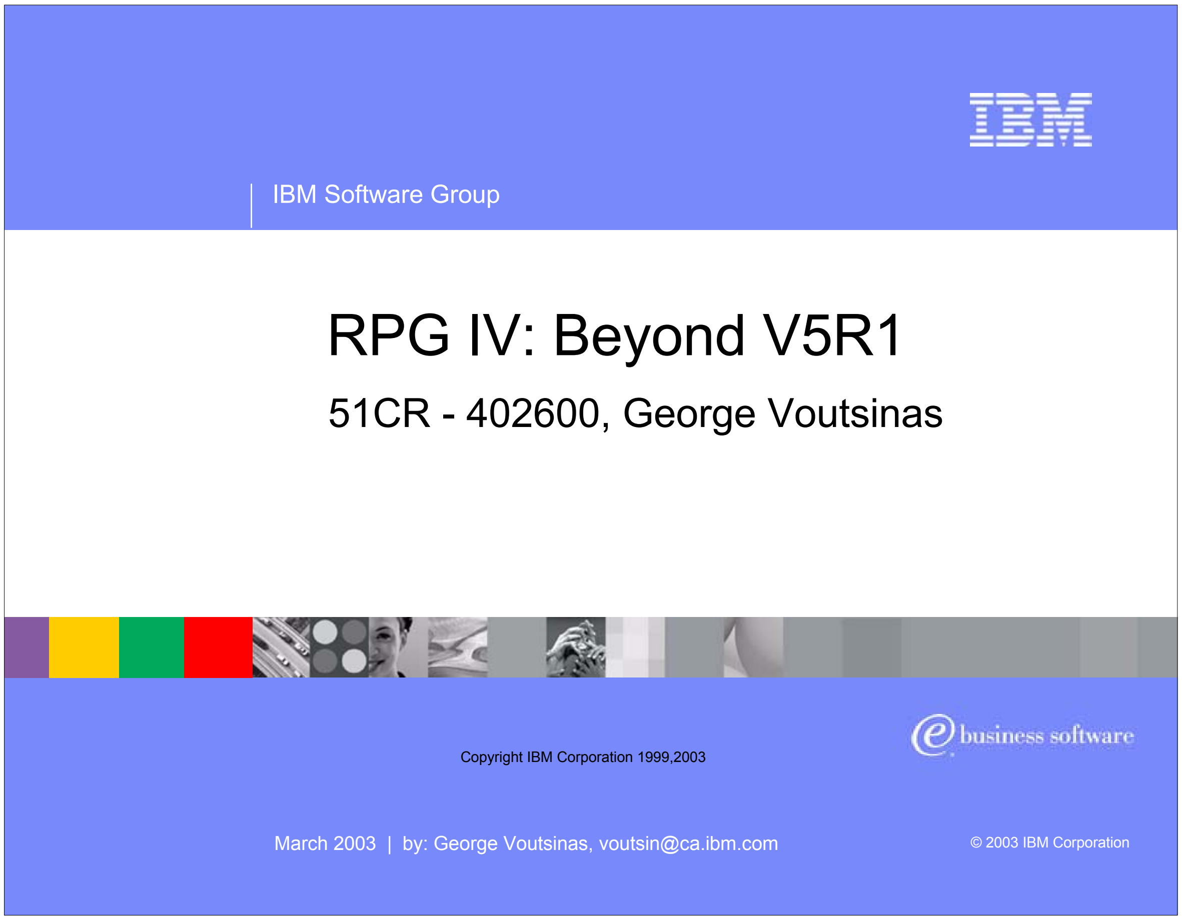


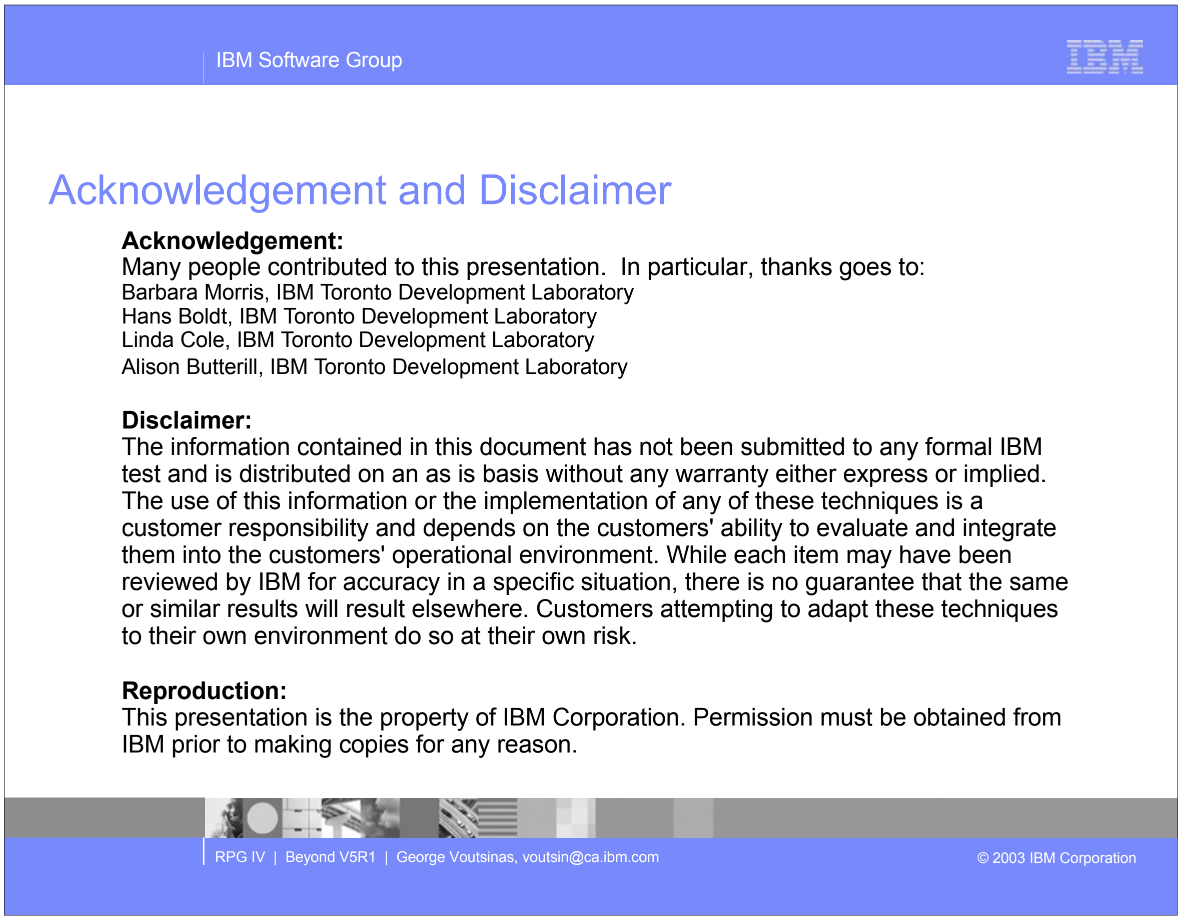
IBM Software Group

RPG IV: Beyond V5R1

51CR - 402600, George Voutsinas

March 2003 | by: George Voutsinas, voutsin@ca.ibm.com

In this presentation we will take you through all the enhancements that have been done in V5R2.
First we take a fast preview on what we have done in V5R1, i.e. just a few foils and then we dig into V5R2
If anything, this should tell you, our customers that RPG IV is still alive and ROCKs!
The reason we say this is the fact that customers always question IBM's commitment to the language. If you look back at every release of RPG, you can see how IBM is committed to the best business language on the iSeries. However, you should also understand that not only RPG IV is important but so is Java. IBM is committed to both. Let each language do what each language is good at. RPG IV is the best business language on the iSeries and Java is the best internet or eBusiness language. Combining both will give you the best eBusiness application possible.
Its important to note that IBM is only committed to enhancing RPG IV but NOT RPG II or RPG III. These languages are done and will not be enhanced at all. We encourage you to move forward.

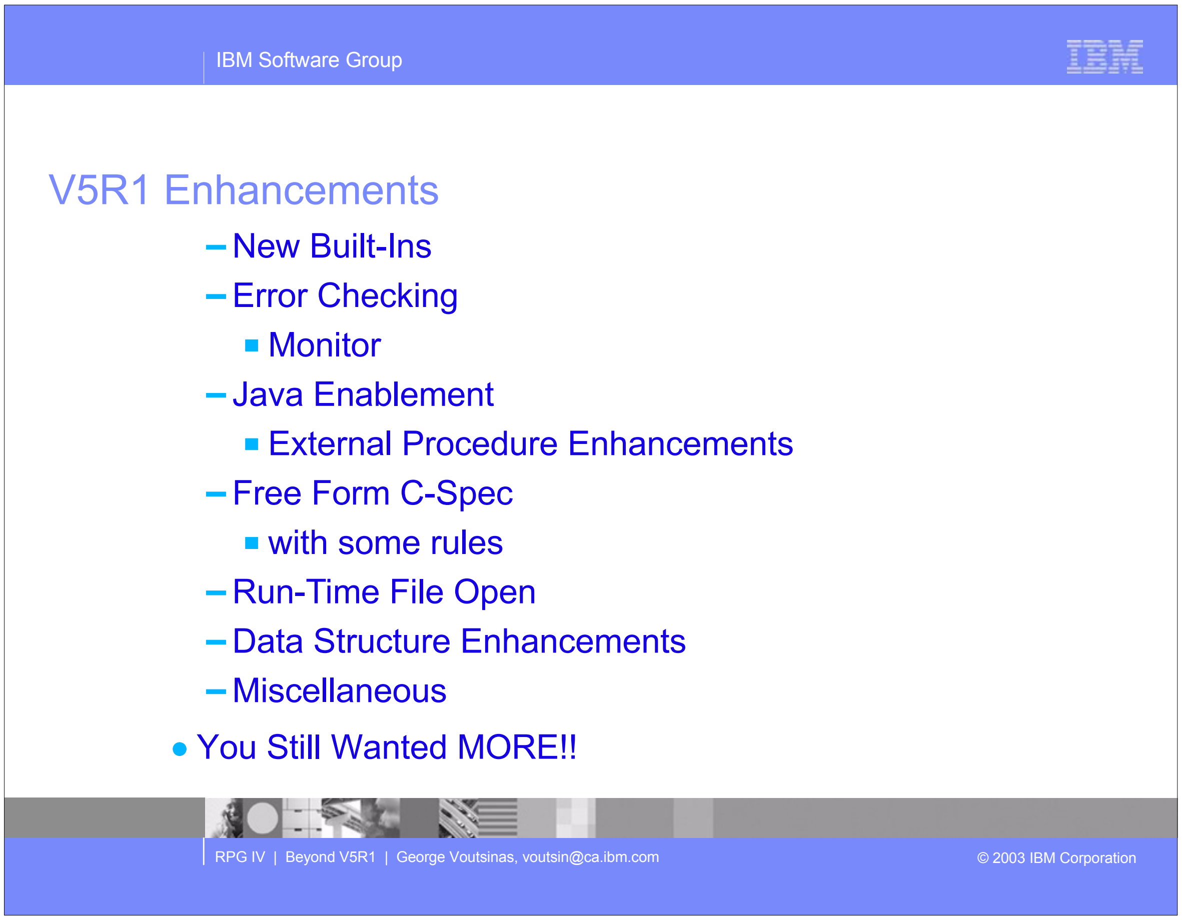


Acknowledgement and Disclaimer

Acknowledgement:
Many people contributed to this presentation. In particular, thanks goes to:
Barbara Morris, IBM Toronto Development Laboratory
Hans Boldt, IBM Toronto Development Laboratory
Linda Cole, IBM Toronto Development Laboratory
Alison Butterill, IBM Toronto Development Laboratory

...Top Requirements after V4R5 ..

- Free Form Calcs!
- Bitwise operations in expressions
- Monitor operation group
- BIF %OCCUR(mds) in expression
- Date/time/timestamp operations in expressions
- Keylists in D specs
- Multiple dimension arrays
- New built-in functions:
  - %CHECKR,%SCANR, %LOOKUP
- ELSEIF
- Runtime control of file to be opened
- More H-Spec keywords...
- Predefined compiler directives
- ...a lot more on the list!.....and the vote is...>>>



V5R1 Enhancements

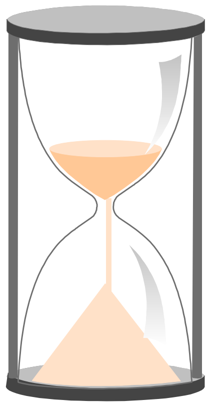
- New Built-Ins
- Error Checking
  - Monitor
- Java Enablement
  - External Procedure Enhancements
- Free Form C-Spec
  - with some rules
- Run-Time File Open
- Data Structure Enhancements
- Miscellaneous

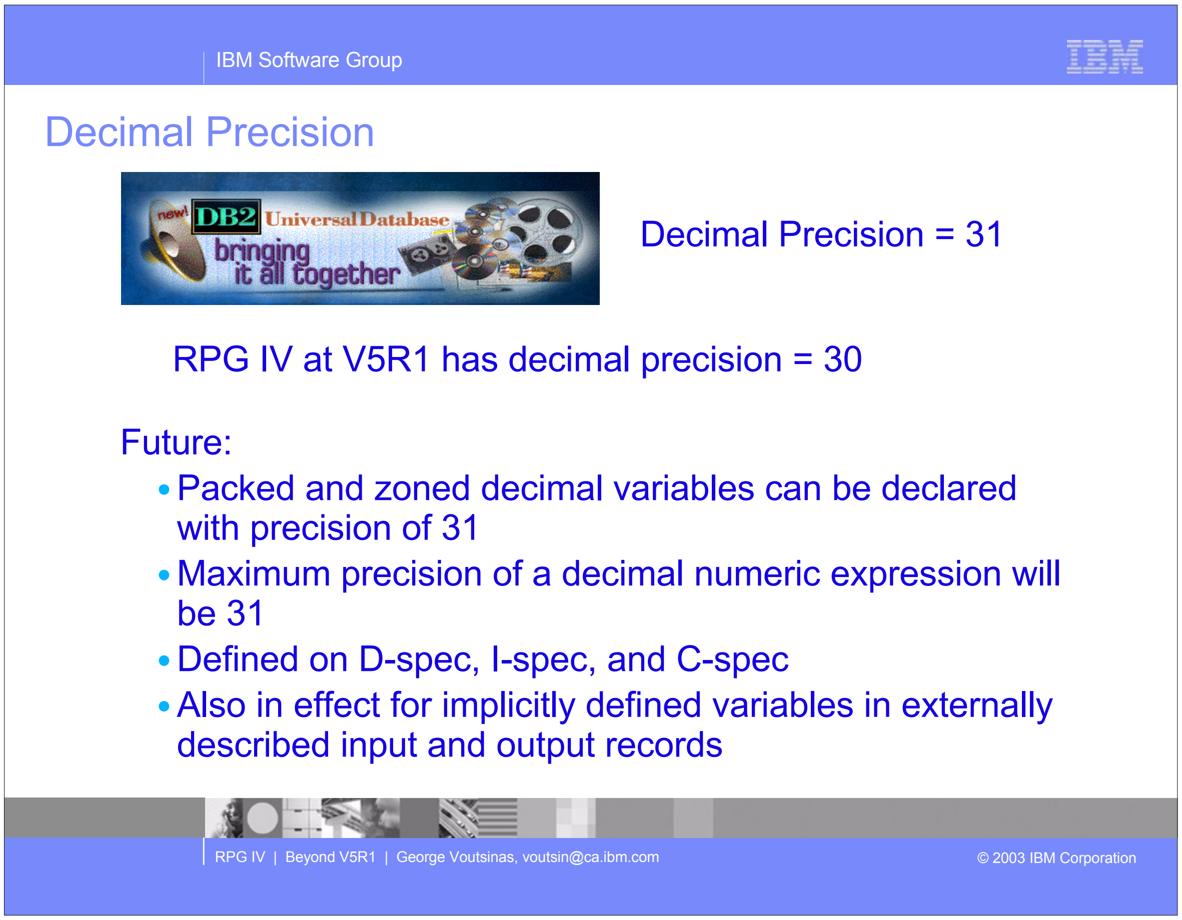
- You Still Wanted MORE!!

- V5R1 and V5R2 have had the biggest enhancements since the inception of RPG IV back in V3R1.
- The enhancements in this release were targeting the interoperability of RPG with Java.
- This can be clearly seen with introduction of having RPG easily instantiate a class in Java, and call its methods
- We will present the detail for each of these enhancements in upcoming slides.
- The way IBM selects function to add to the language is by going out and collecting requirements from customers. We at IBM have a database of all these requirements and we go out and do a survey to see what customers want.
- For V5R1 we received about 455 notes back on what enhancements to include. What you see in this list is what made it for the V5R1 release

Agenda

- V5R2 Enhancements
  - Enhanced Decimal Precision
  - Conversion of Character to Numeric
  - Data Structure Enhancements
  - Nested Data Structures
  - Improved Keyed I/O Operations
  - Qualified Data Areas
  - Short Form Operations
  - Other Enhancements
- Future
- Summary





Decimal Precision

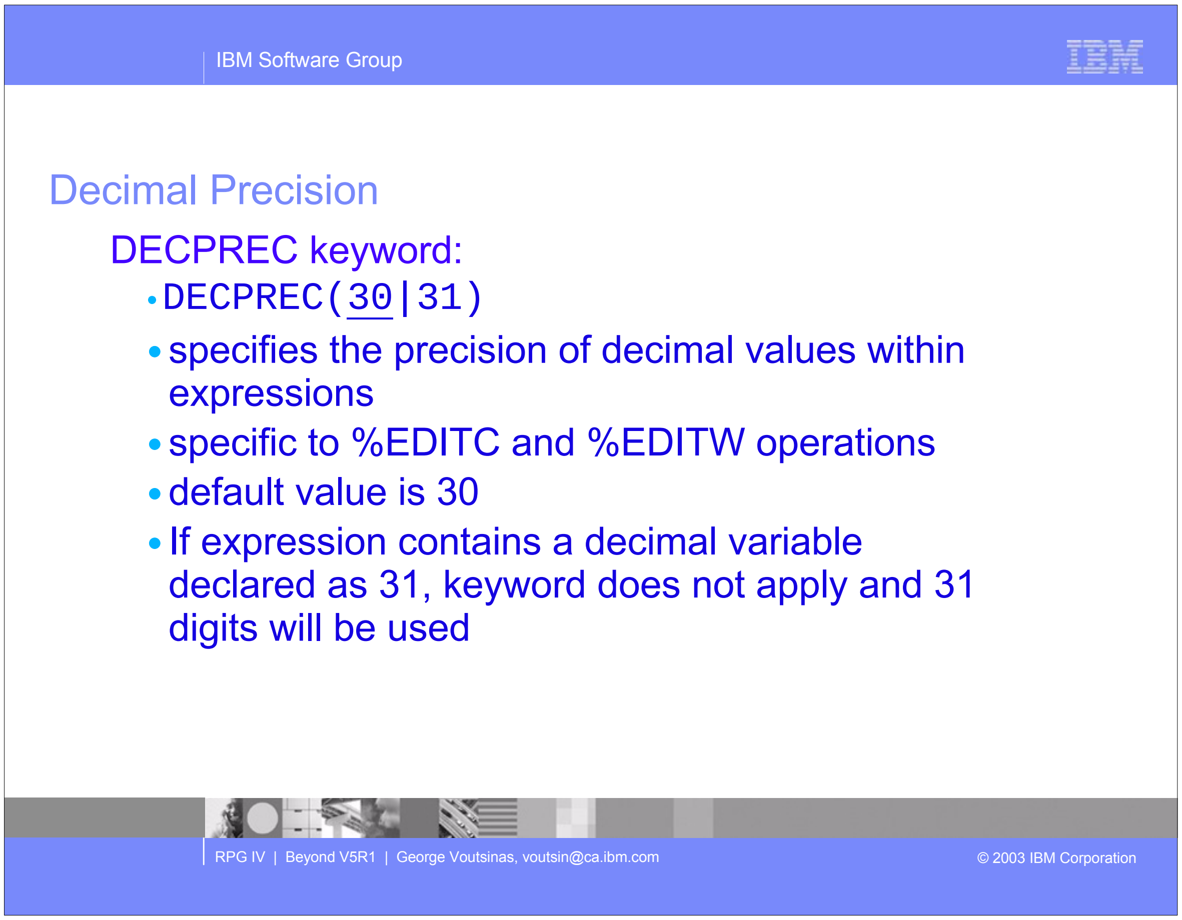
Decimal Precision = 31

RPG IV at V5R1 has decimal precision = 30

Future:

- Packed and zoned decimal variables can be declared with precision of 31
- Maximum precision of a decimal numeric expression will be 31
- Defined on D-spec, I-spec, and C-spec
- Also in effect for implicitly defined variables in externally described input and output records

We added support for 31-digit packed and zoned variables
We also added a new DECPREC option, similar to the current INTPREC, with similar rules.
Packed and zoned fields can now be declared with a precision of 31 digits
Why? Well the system sometimes generates fields that are 31 digit which RPG could not handle, now it can!

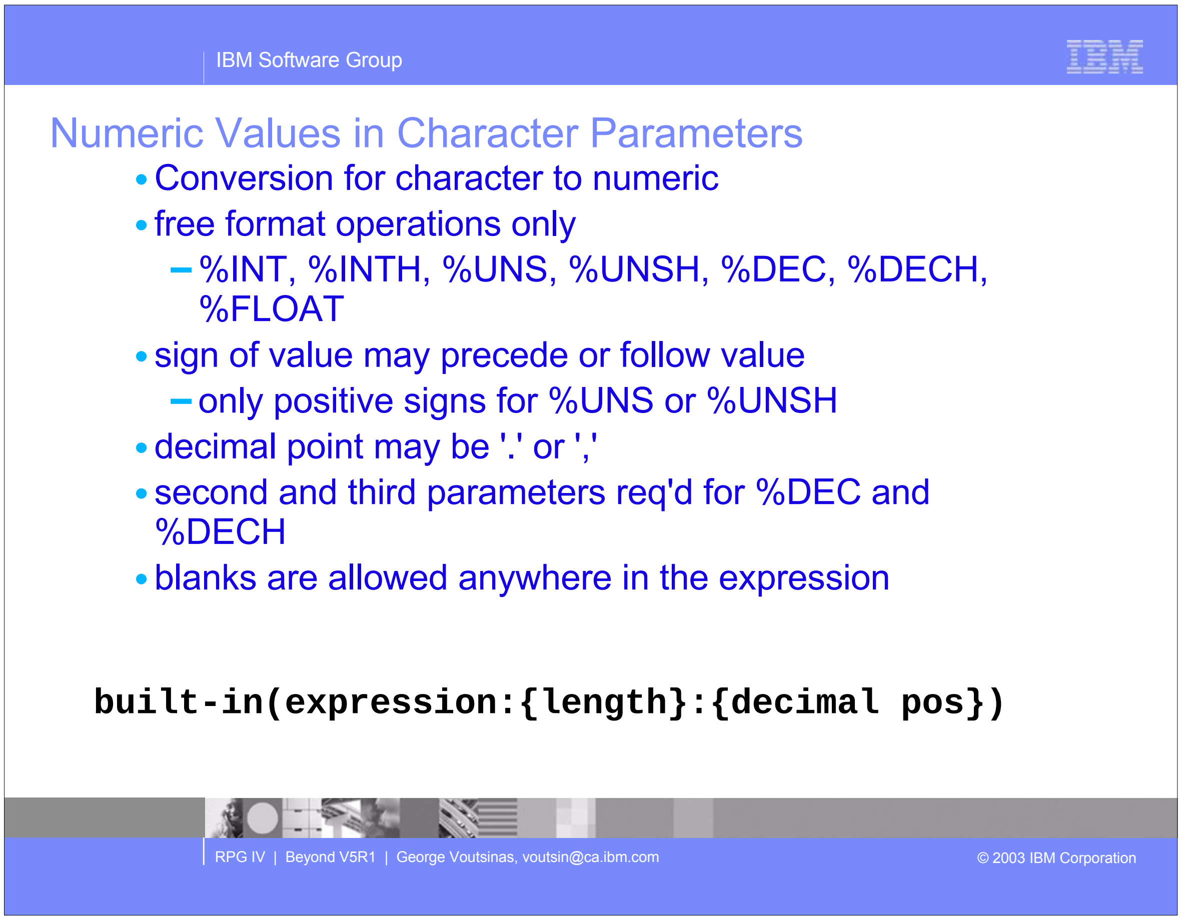


Decimal Precision

DECPREC keyword:

- `DECPREC(30|31)`
- specifies the precision of decimal values within expressions
- specific to %EDITC and %EDITW operations
- default value is 30
- If expression contains a decimal variable declared as 31, keyword does not apply and 31 digits will be used

- This slide describes the rules.

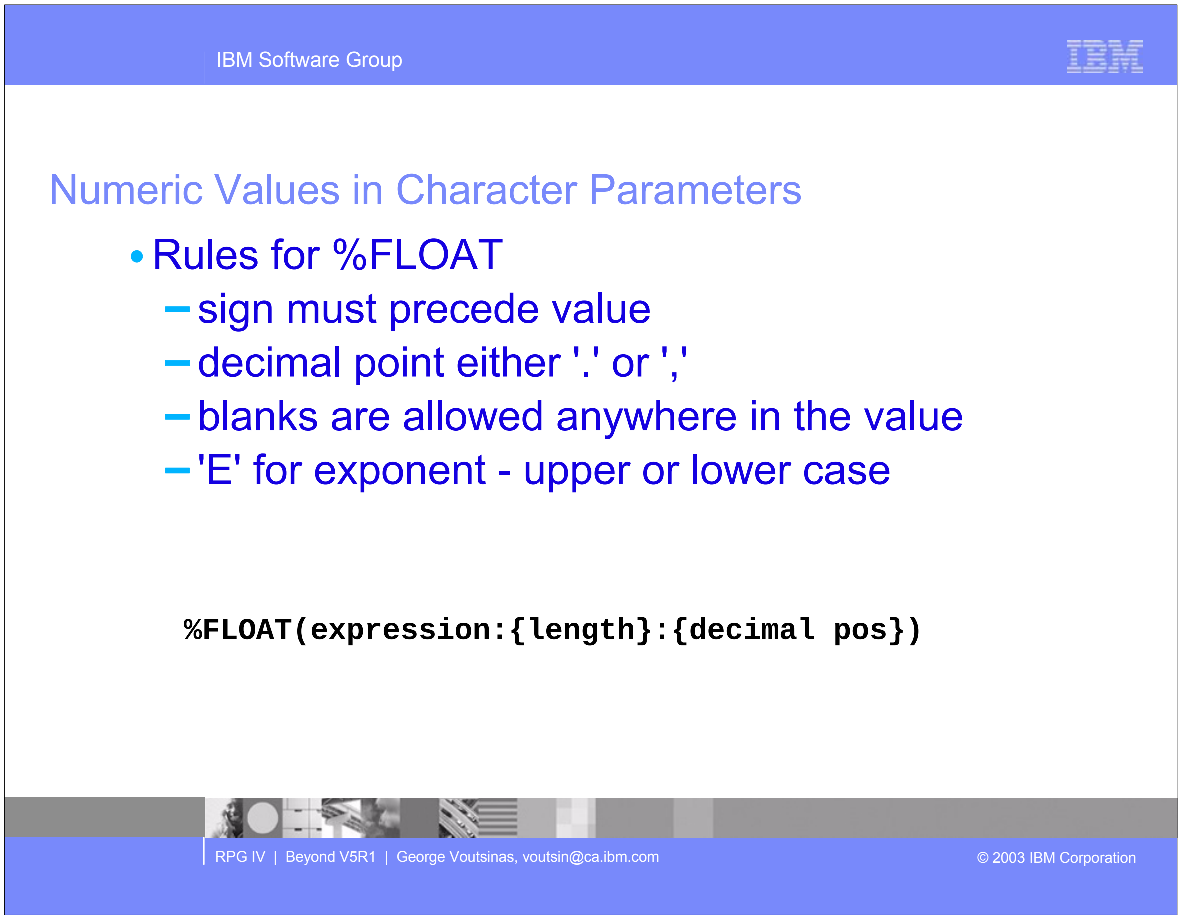


Numeric Values in Character Parameters

- Conversion for character to numeric
- free format operations only
  - %INT, %INTH, %UNS, %UNSH, %DEC, %DECH, %FLOAT
- sign of value may precede or follow value
  - only positive signs for %UNS or %UNSH
- decimal point may be '.' or ','
- second and third parameters req'd for %DEC and %DECH
- blanks are allowed anywhere in the expression

```
built-in(expression:{length}:{decimal pos})
```

- This slide describes the rules.

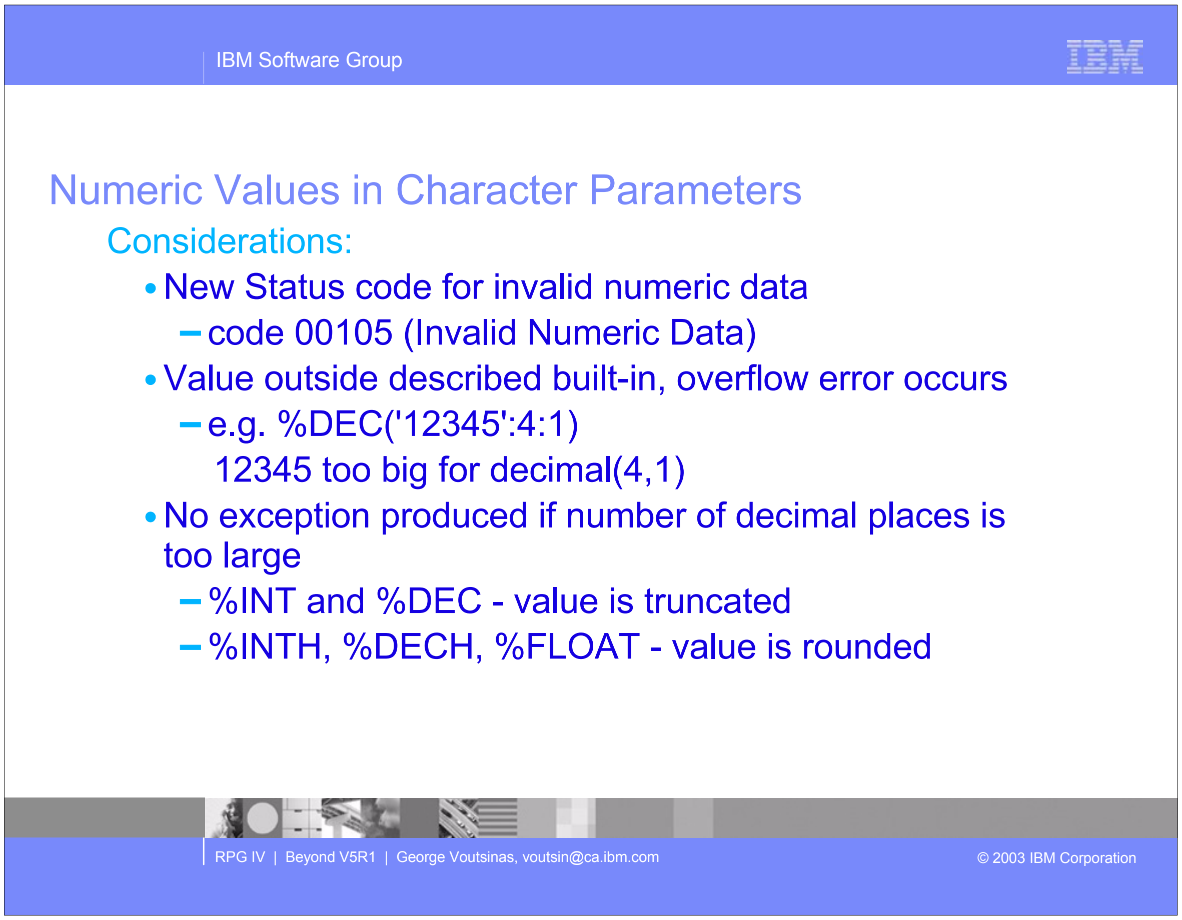


Numeric Values in Character Parameters

- Rules for %FLOAT
  - sign must precede value
  - decimal point either '.' or ','
  - blanks are allowed anywhere in the value
  - 'E' for exponent - upper or lower case

```
%FLOAT(expression:{length}:{decimal pos})
```

- This slide describes the rules.

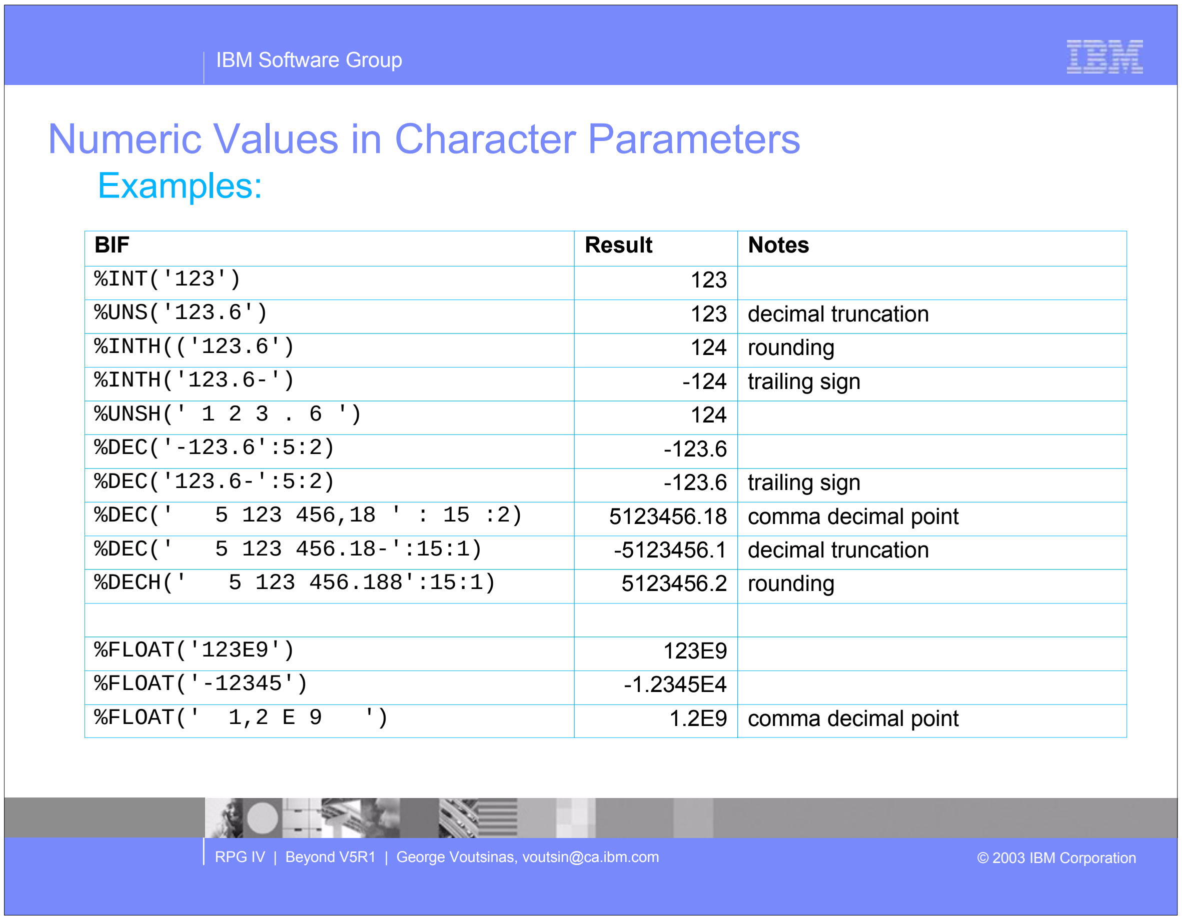


Numeric Values in Character Parameters

Considerations:

- New Status code for invalid numeric data
  - code 00105 (Invalid Numeric Data)
- Value outside described built-in, overflow error occurs
  - e.g. %DEC('12345':4:1)
    12345 too big for decimal(4,1)
- No exception produced if number of decimal places is too large
  - %INT and %DEC - value is truncated
  - %INTH, %DECH, %FLOAT - value is rounded

▸ This slide describes the rules.

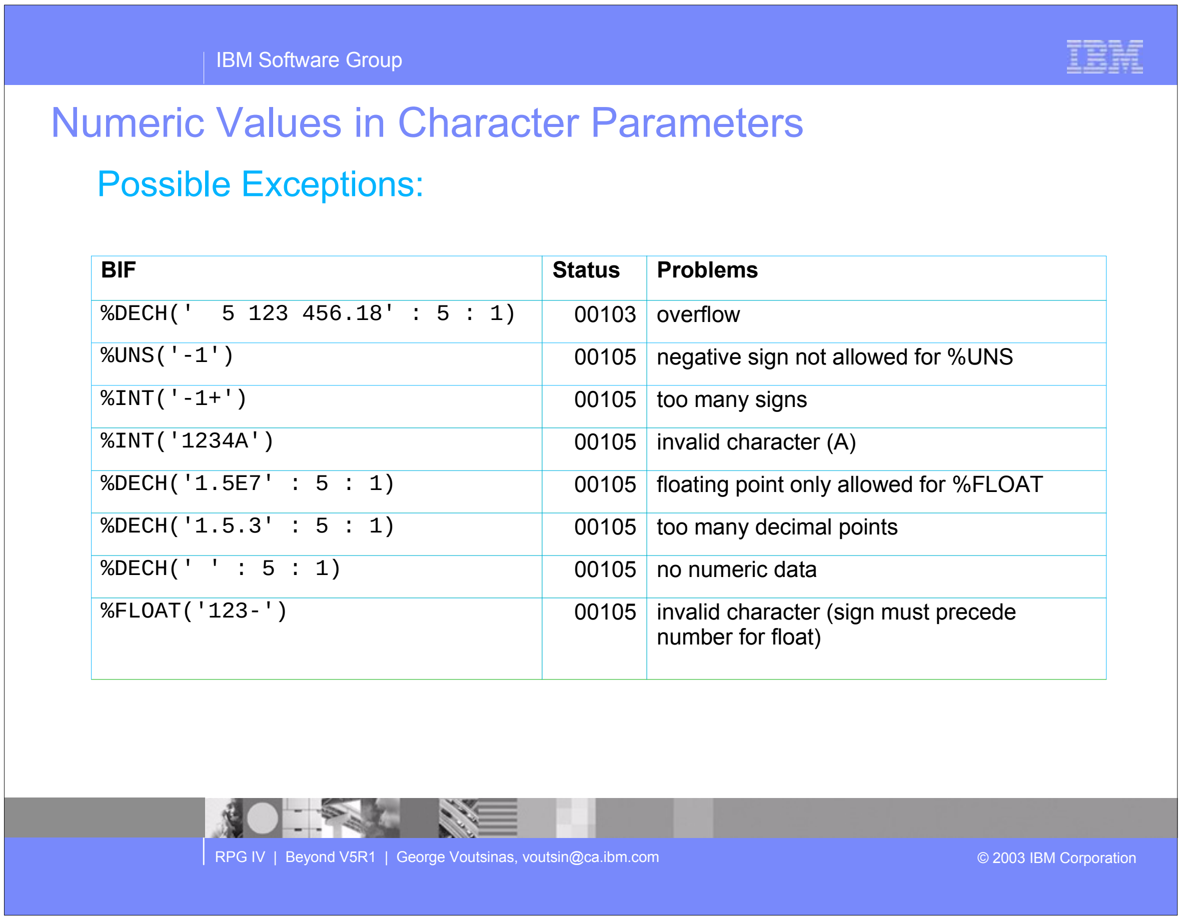


Numeric Values in Character Parameters

Examples:

BIF	Result	Notes
%INT('123')	123	
%UNS('123.6')	123	decimal truncation
%INTH(('123.6')	124	rounding
%INTH('123.6-')	-124	trailing sign
%UNSH(' 1 2 3 . 6 ')	124	
%DEC('-123.6':5:2)	-123.6	
%DEC('123.6-':5:2)	-123.6	trailing sign
%DEC(' 5 123 456,18 ' : 15 :2)	5123456.18	comma decimal point
%DEC(' 5 123 456.18-':15:1)	-5123456.1	decimal truncation
%DECH(' 5 123 456.188':15:1)	5123456.2	rounding
%FLOAT('123E9')	123E9	
%FLOAT('-12345')	-1.2345E4	
%FLOAT(' 1,2 E 9 ')	1.2E9	comma decimal point

- The first column shows the BIF and the parameter used, the result column shows what is the result, and the notes just FYI.
- Review the program line by line describing each lines purpose. For example, in line three, because it is half adjust, the result of %INTH('123.6') is 124!

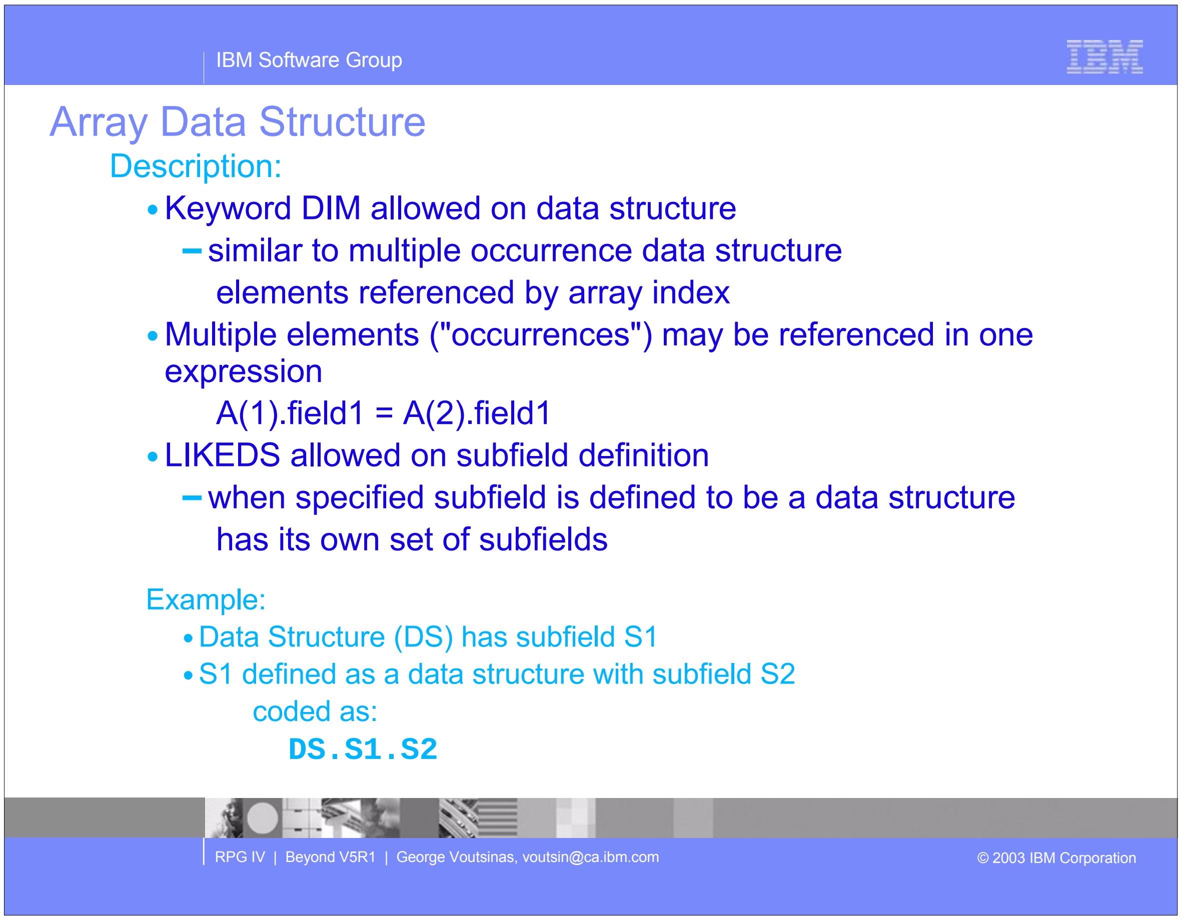


Numeric Values in Character Parameters

Possible Exceptions:

BIF	Status	Problems
`%DECH('  5 123 456.18' : 5 : 1)`	00103	overflow
`%UNS('-1')`	00105	negative sign not allowed for %UNS
`%INT('-1+')`	00105	too many signs
`%INT('1234A')`	00105	invalid character (A)
`%DECH('1.5E7' : 5 : 1)`	00105	floating point only allowed for %FLOAT
`%DECH('1.5.3' : 5 : 1)`	00105	too many decimal points
`%DECH(' ' : 5 : 1)`	00105	no numeric data
`%FLOAT('123-')`	00105	invalid character (sign must precede number for float)

- This is an example of an exception.
- Review the program line by line to understand why you would have a failure. The problem column highlights the reason for the problem.



Array Data Structure

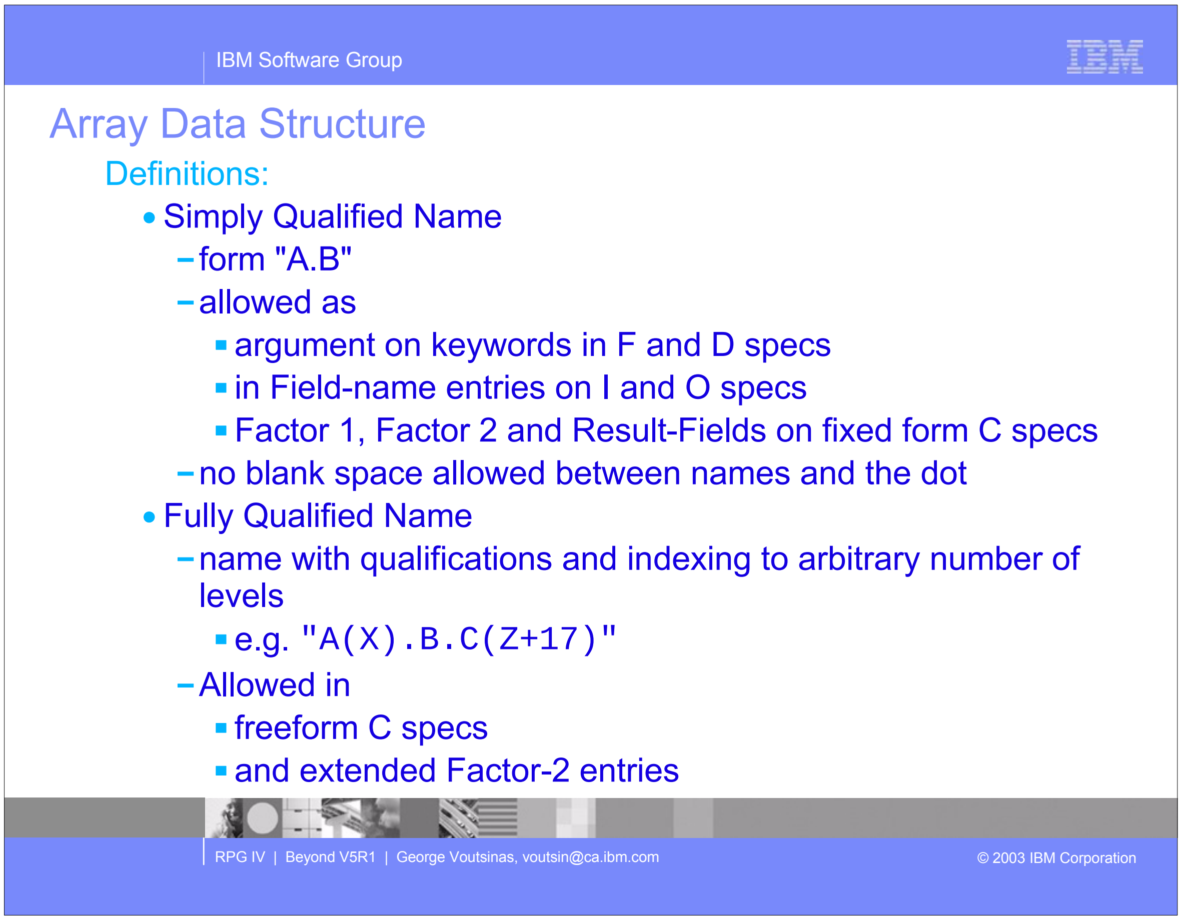
Description:

- Keyword DIM allowed on data structure
  - similar to multiple occurrence data structure
    elements referenced by array index
- Multiple elements ("occurrences") may be referenced in one expression
  A(1).field1 = A(2).field1
- LIKEDS allowed on subfield definition
  - when specified subfield is defined to be a data structure
    has its own set of subfields

Example:

- Data Structure (DS) has subfield S1
- S1 defined as a data structure with subfield S2
  coded as:
  DS.S1.S2

- These are the rules

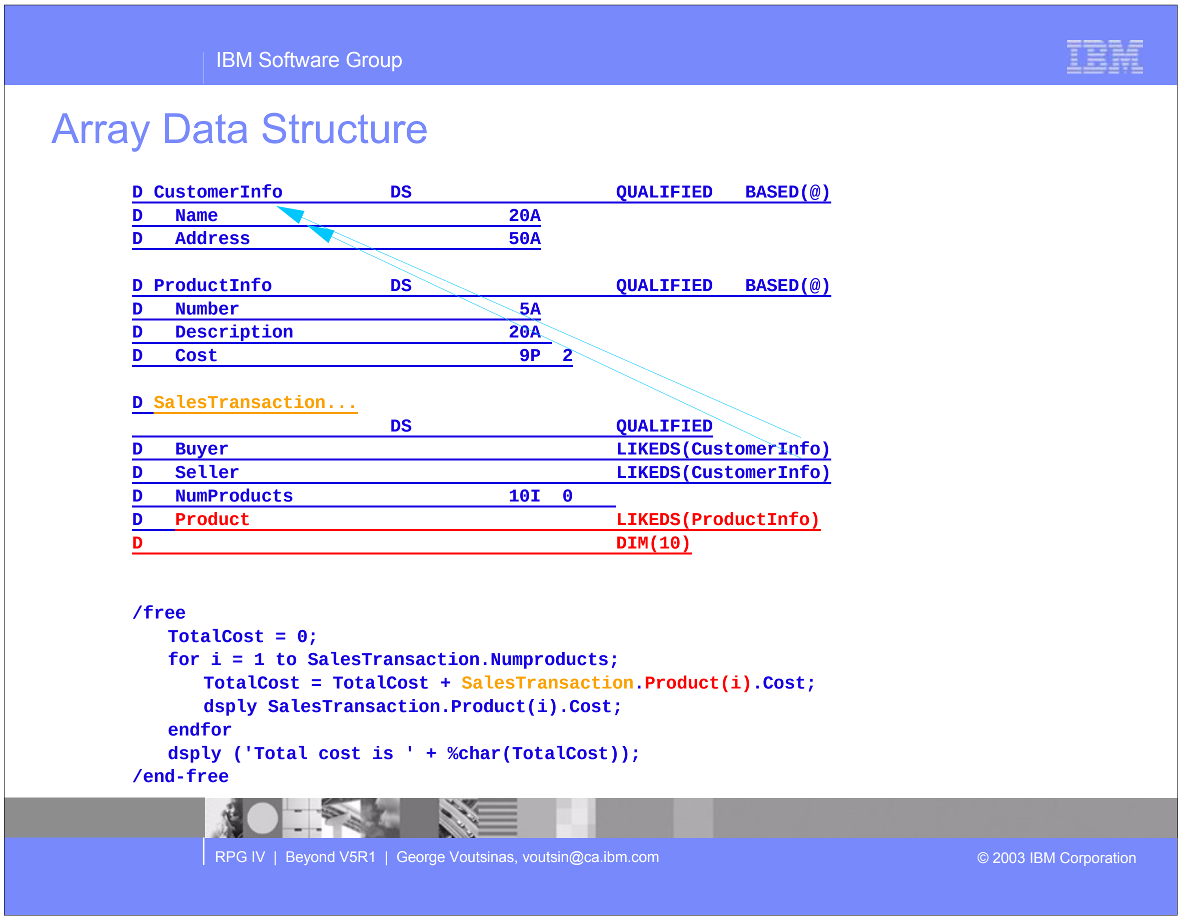


Array Data Structure

Definitions:

- Simply Qualified Name
  - form "A.B"
  - allowed as
    - argument on keywords in F and D specs
    - in Field-name entries on I and O specs
    - Factor 1, Factor 2 and Result-Fields on fixed form C specs
  - no blank space allowed between names and the dot
- Fully Qualified Name
  - name with qualifications and indexing to arbitrary number of levels
    - e.g. `"A(X).B.C(Z+17)"`
  - Allowed in
    - freeform C specs
    - and extended Factor-2 entries

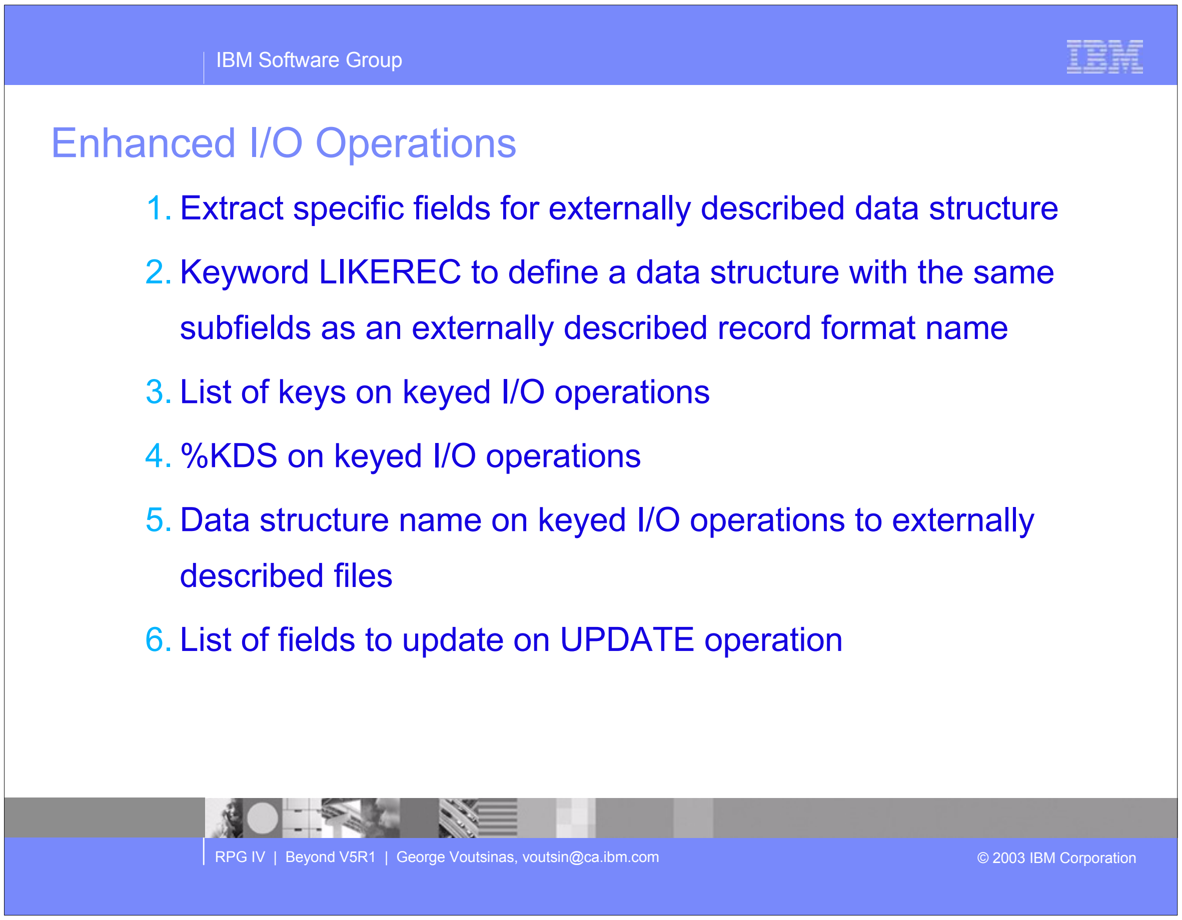
- These are the rules



Array Data Structure

```
D CustomerInfo          DS                    QUALIFIED   BASED(@)
D   Name                           20A
D   Address                        50A

D ProductInfo           DS                    QUALIFIED   BASED(@)
D   Number                          5A
D   Description                    20A
D   Cost                            9P  2

D SalesTransaction...
                        DS                    QUALIFIED
D   Buyer                                     LIKEDS(CustomerInfo)
D   Seller                                    LIKEDS(CustomerInfo)
D   NumProducts                    10I  0
D   Product                                   LIKEDS(ProductInfo)
D                                             DIM(10)


/free
   TotalCost = 0;
   for i = 1 to SalesTransaction.Numproducts;
      TotalCost = TotalCost + SalesTransaction.Product(i).Cost;
      dsply SalesTransaction.Product(i).Cost;
   endfor
   dsply ('Total cost is ' + %char(TotalCost));
/end-free
```

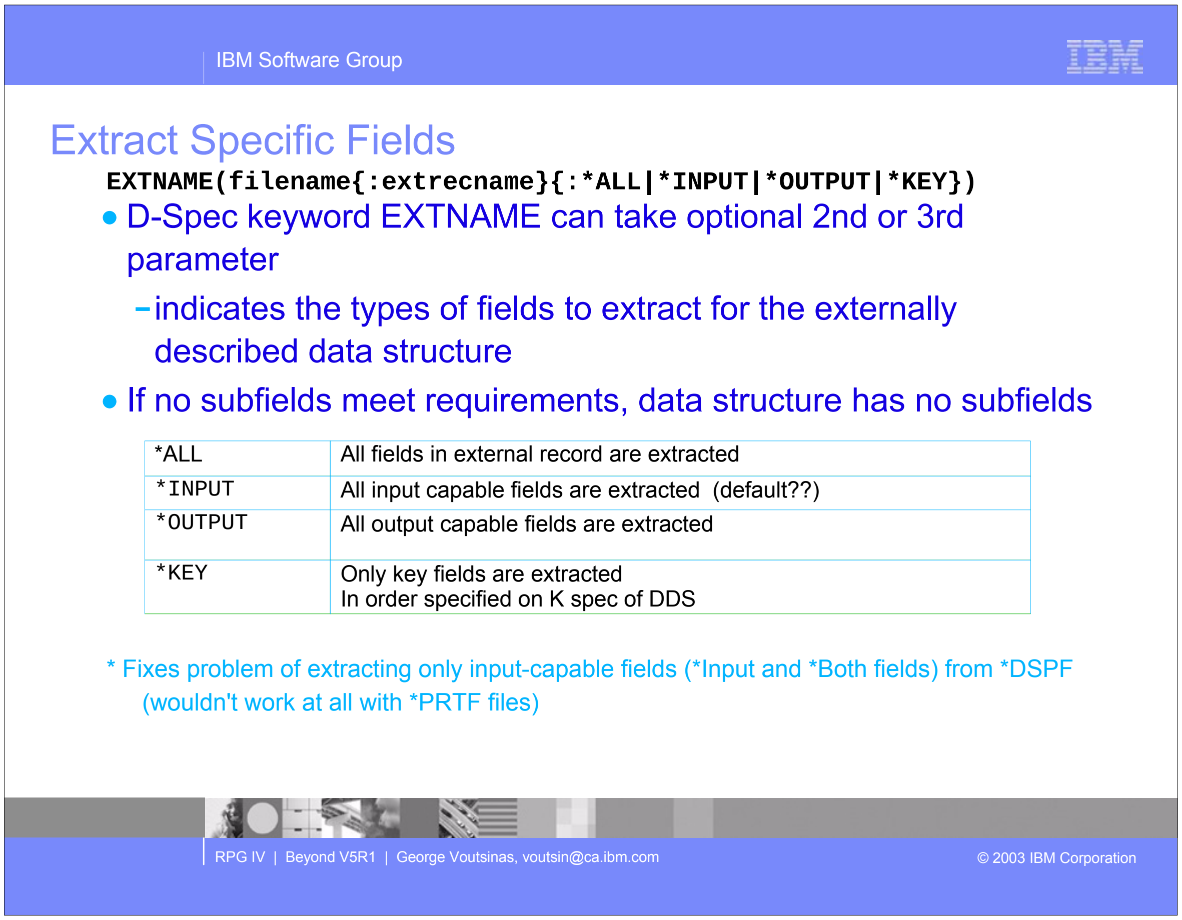
- This shows an example of the data structure enhancement
- Review the example at the SalesTransaction declaration. In that data structure you see the 'Product' subfield which is LIKEDS(ProductInfo) and it is an array of DIM 10. Now if you look at the C specification you will see we reference it by qualifying it and also using a dimension



Enhanced I/O Operations

1. Extract specific fields for externally described data structure
2. Keyword LIKEREC to define a data structure with the same subfields as an externally described record format name
3. List of keys on keyed I/O operations
4. %KDS on keyed I/O operations
5. Data structure name on keyed I/O operations to externally described files
6. List of fields to update on UPDATE operation

- These are the RPG IV enhancements to support I/O.



Extract Specific Fields

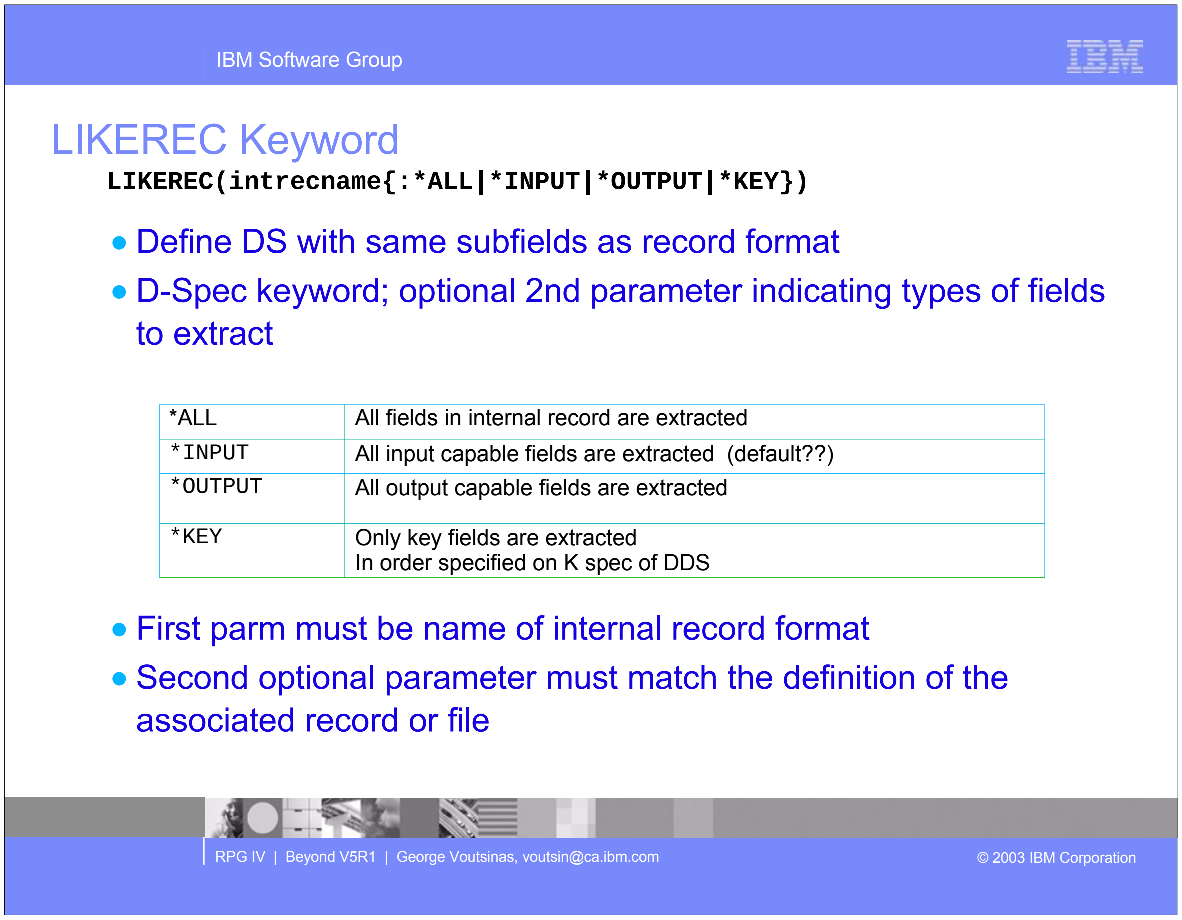
```
EXTNAME(filename{:extrecname}{:*ALL|*INPUT|*OUTPUT|*KEY})
```

- D-Spec keyword EXTNAME can take optional 2nd or 3rd parameter
  - indicates the types of fields to extract for the externally described data structure
- If no subfields meet requirements, data structure has no subfields

*ALL	All fields in external record are extracted
*INPUT	All input capable fields are extracted (default??)
*OUTPUT	All output capable fields are extracted
*KEY	Only key fields are extracted In order specified on K spec of DDS

* Fixes problem of extracting only input-capable fields (*Input and *Both fields) from *DSPF (wouldn't work at all with *PRTF files)

- Extract specific fields for externally described data structure
- D-Spec keyword EXTNAME can take an optional 2nd or 3rd parameter, which indicates the types of fields to extract for externally described data structures
- The option *ALL, *INPUT, *OUTPUT,*KEY are described on the chart
- If there are no subfields that meet the requirements, the data structure will have no subfields, and existing diagnostic messages will be issued.



LIKEREC Keyword

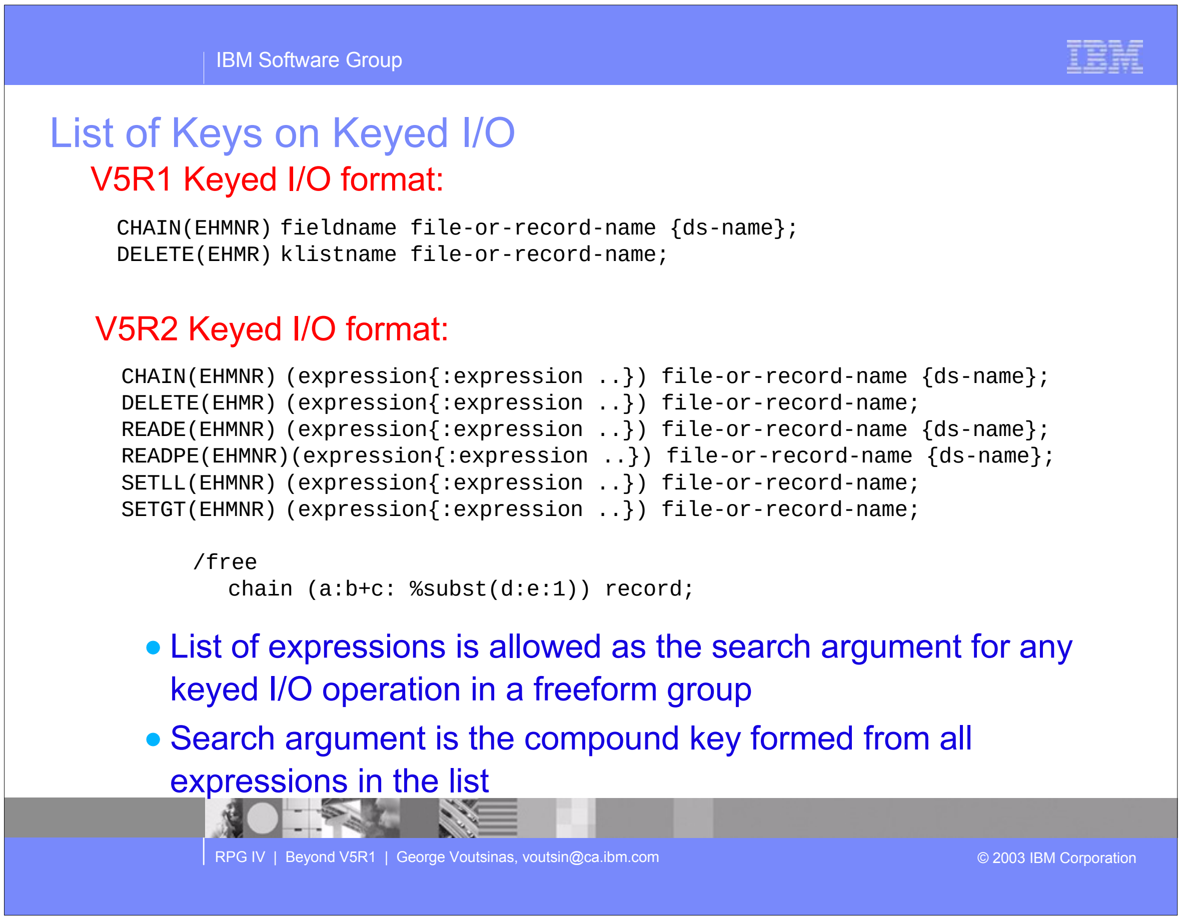
```
LIKEREC(intrecname{:*ALL|*INPUT|*OUTPUT|*KEY})
```

- Define DS with same subfields as record format
- D-Spec keyword; optional 2nd parameter indicating types of fields to extract

*ALL	All fields in internal record are extracted
*INPUT	All input capable fields are extracted (default??)
*OUTPUT	All output capable fields are extracted
*KEY	Only key fields are extracted In order specified on K spec of DDS

- First parm must be name of internal record format
- Second optional parameter must match the definition of the associated record or file

- Keyword LIKEREC is used like keyword LIKEDS, but instead of getting the subfield list from another data structure, the subfields are taken from the specified internal record name. D-Spec keyword LIKEREC can take an optional 2nd parameter, which indicated the types of fields to extract for the externally described data structure. ALL options are the same as the previous slide.



List of Keys on Keyed I/O

V5R1 Keyed I/O format:

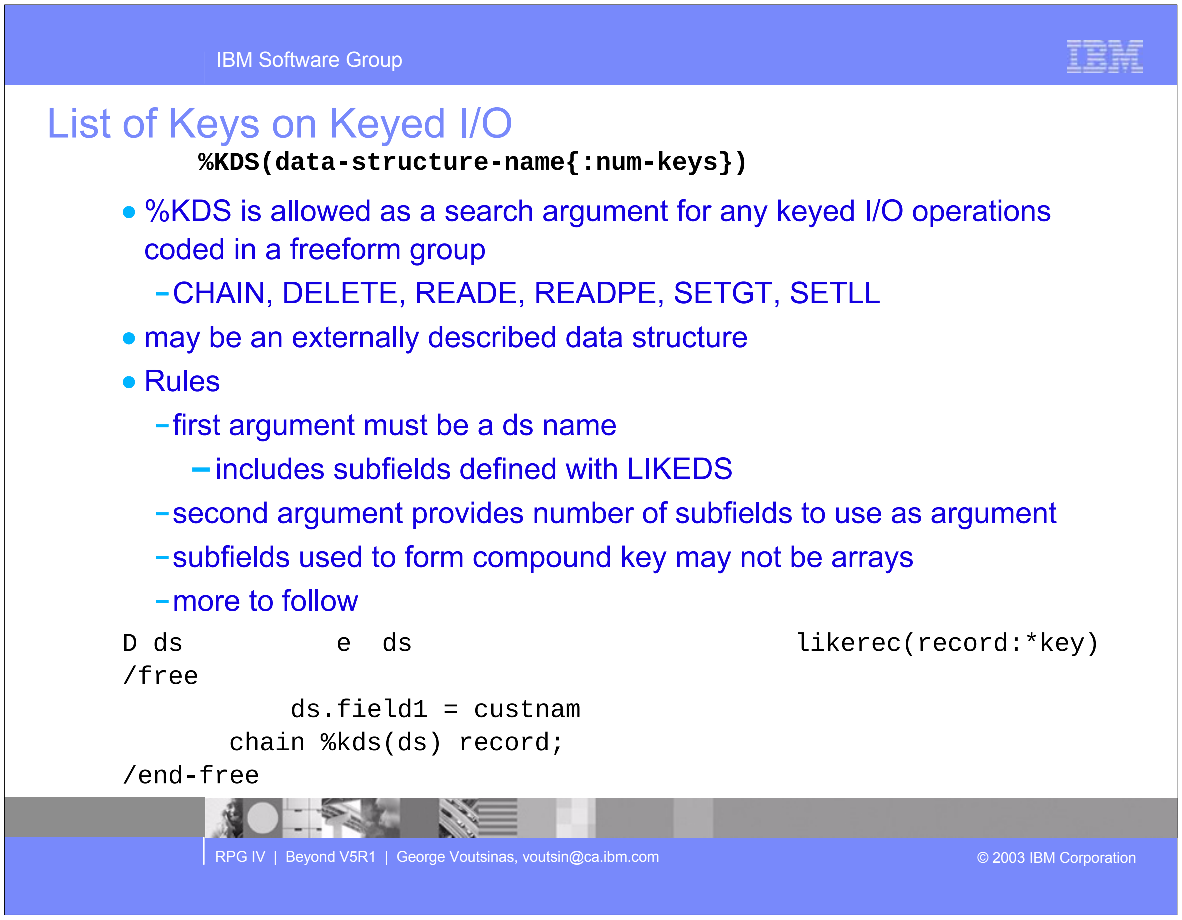
```
CHAIN(EHMNR) fieldname file-or-record-name {ds-name};
DELETE(EHMR) klistname file-or-record-name;
```

V5R2 Keyed I/O format:

```
CHAIN(EHMNR) (expression{:expression ..}) file-or-record-name {ds-name};
DELETE(EHMR) (expression{:expression ..}) file-or-record-name;
READE(EHMNR) (expression{:expression ..}) file-or-record-name {ds-name};
READPE(EHMNR)(expression{:expression ..}) file-or-record-name {ds-name};
SETLL(EHMNR) (expression{:expression ..}) file-or-record-name;
SETGT(EHMNR) (expression{:expression ..}) file-or-record-name;

      /free
        chain (a:b+c: %subst(d:e:1)) record;
```

- List of expressions is allowed as the search argument for any keyed I/O operation in a freeform group
- Search argument is the compound key formed from all expressions in the list

- A list of expressions is allowed as the search argument for any keyed I/O operation (Chain, Delete, READE, READPE, SETGT,and SETLL) in free-form group. The search argument is the compound key formed from all expressions specified in the list.

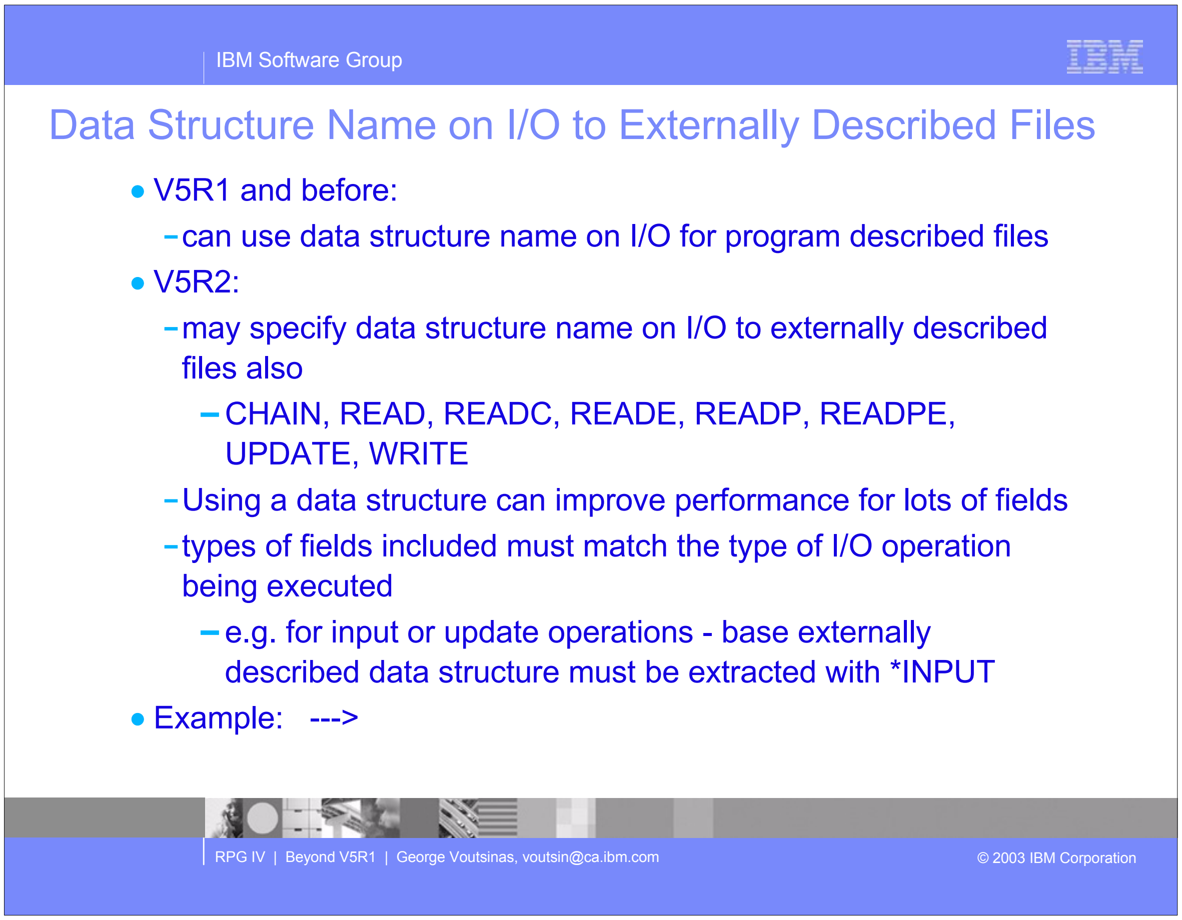


List of Keys on Keyed I/O

`%KDS(data-structure-name{:num-keys})`

- %KDS is allowed as a search argument for any keyed I/O operations coded in a freeform group
  - CHAIN, DELETE, READE, READPE, SETGT, SETLL
- may be an externally described data structure
- Rules
  - first argument must be a ds name
    - includes subfields defined with LIKEDS
  - second argument provides number of subfields to use as argument
  - subfields used to form compound key may not be arrays
  - more to follow

```
D ds           e  ds                        likerec(record:*key)
/free
           ds.field1 = custnam
       chain %kds(ds) record;
/end-free
```



Data Structure Name on I/O to Externally Described Files

- V5R1 and before:
  - can use data structure name on I/O for program described files
- V5R2:
  - may specify data structure name on I/O to externally described files also
    - CHAIN, READ, READC, READE, READP, READPE, UPDATE, WRITE
  - Using a data structure can improve performance for lots of fields
  - types of fields included must match the type of I/O operation being executed
    - e.g. for input or update operations - base externally described data structure must be extracted with *INPUT
- Example: --->

Before and after data structure name on I/O for program described files.

Data Structure Name on I/O to Externally Described Files

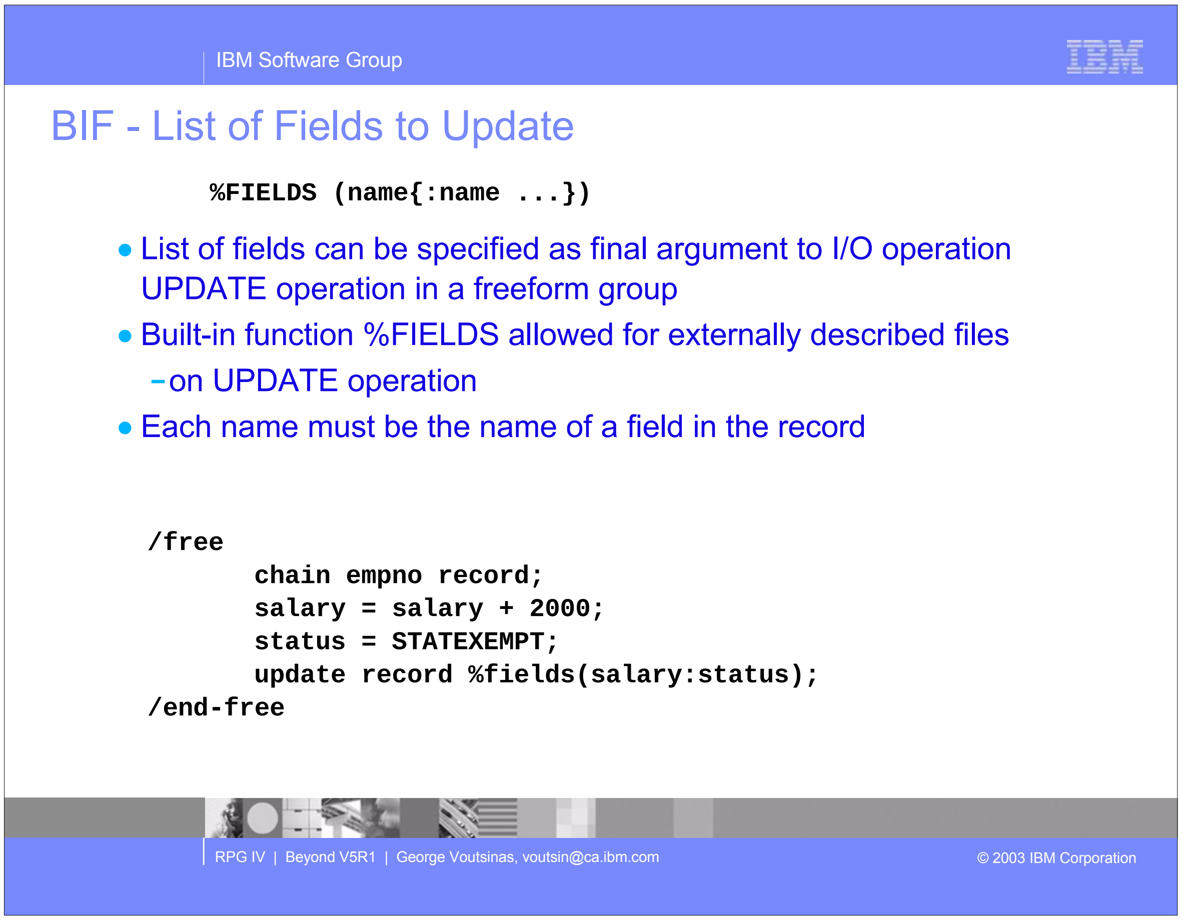
- Example:

DDS

```
A      R REC
A        FLD1    10A
```

RPG

```
FCUSTFILE    UF         E               DISK
D custDs           DS               LIKEREC(rec : *INPUT)

 /free
      read rec custDs;
      custDs.fld1 = 'some new value';
      update rec custDs;
```



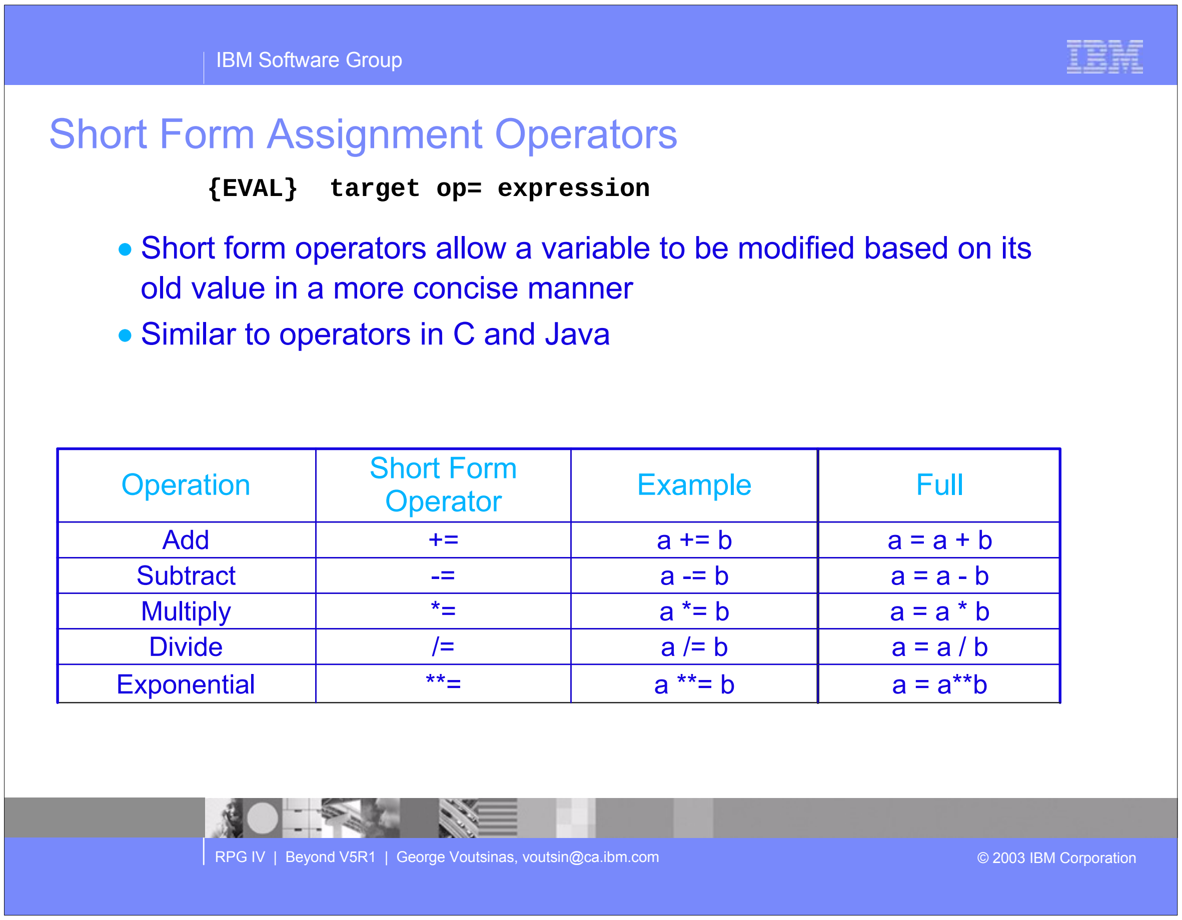
BIF - List of Fields to Update

```
%FIELDS (name{:name ...})
```

- List of fields can be specified as final argument to I/O operation UPDATE operation in a freeform group
- Built-in function %FIELDS allowed for externally described files
  - on UPDATE operation
- Each name must be the name of a field in the record

```
/free
       chain empno record;
       salary = salary + 2000;
       status = STATEXEMPT;
       update record %fields(salary:status);
/end-free
```

A list of fields can be specified as the final argument to I/O operation UPDATE coded in a free-form group.
Only the fields specified are updated into the I/O buffer.
Many customers wanted this enhancement



Short Form Assignment Operators

```
{EVAL}  target op= expression
```

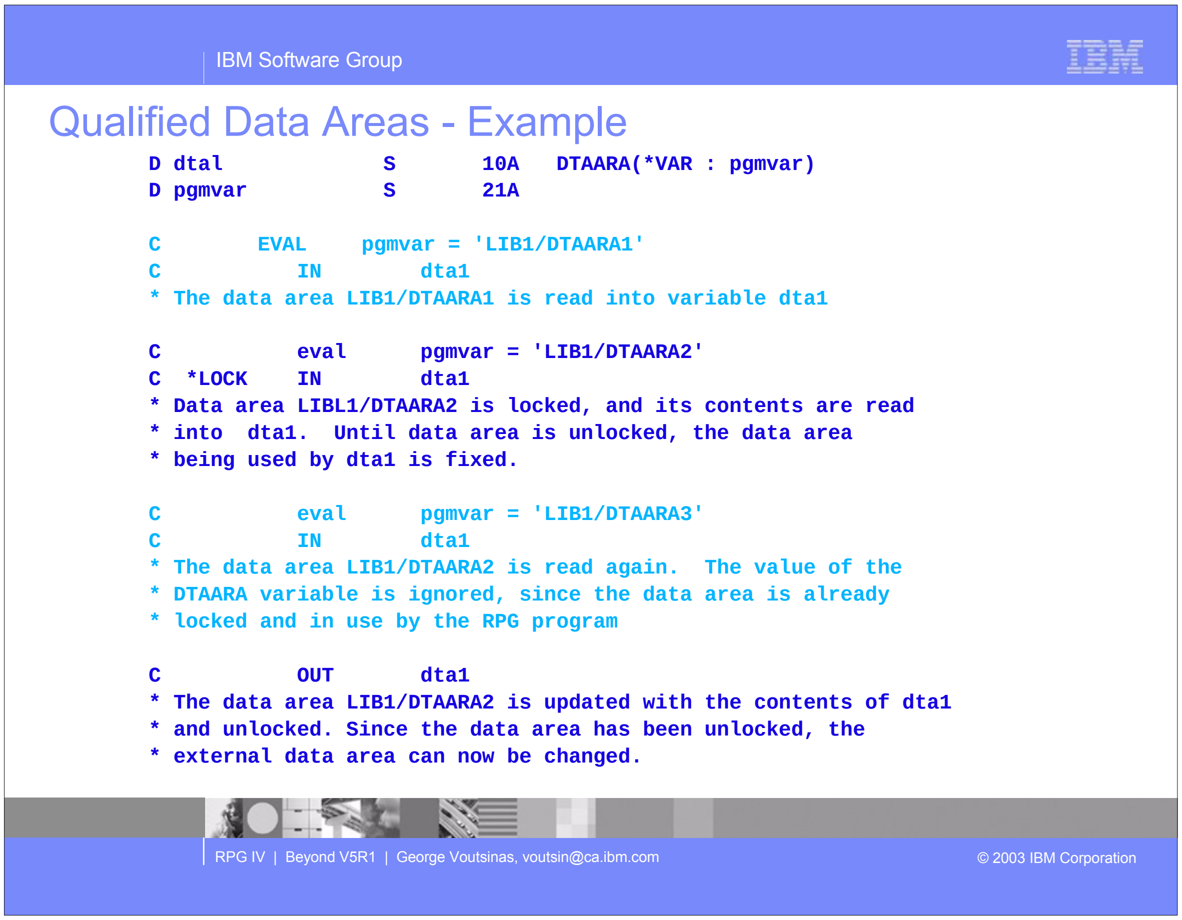
- Short form operators allow a variable to be modified based on its old value in a more concise manner
- Similar to operators in C and Java

Operation	Short Form Operator	Example	Full
Add	+=	a += b	a = a + b
Subtract	-=	a -= b	a = a - b
Multiply	*=	a *= b	a = a * b
Divide	/=	a /= b	a = a / b
Exponential	**=	a **= b	a = a**b

- Customers when they see this slide always say that we are trying to make RPG look like Java! The answer to that question is no. We are just responding to programmers who used both RPG and Java and got used to these operators and wanted IBM to do the same for RPG.
- The example column and the Full column are identical in function. They cause no additional performance work if one or the other is used. Just use any form you prefer.
- You can still use the good old op-codes as well. All are supported now.

Qualified Data Areas

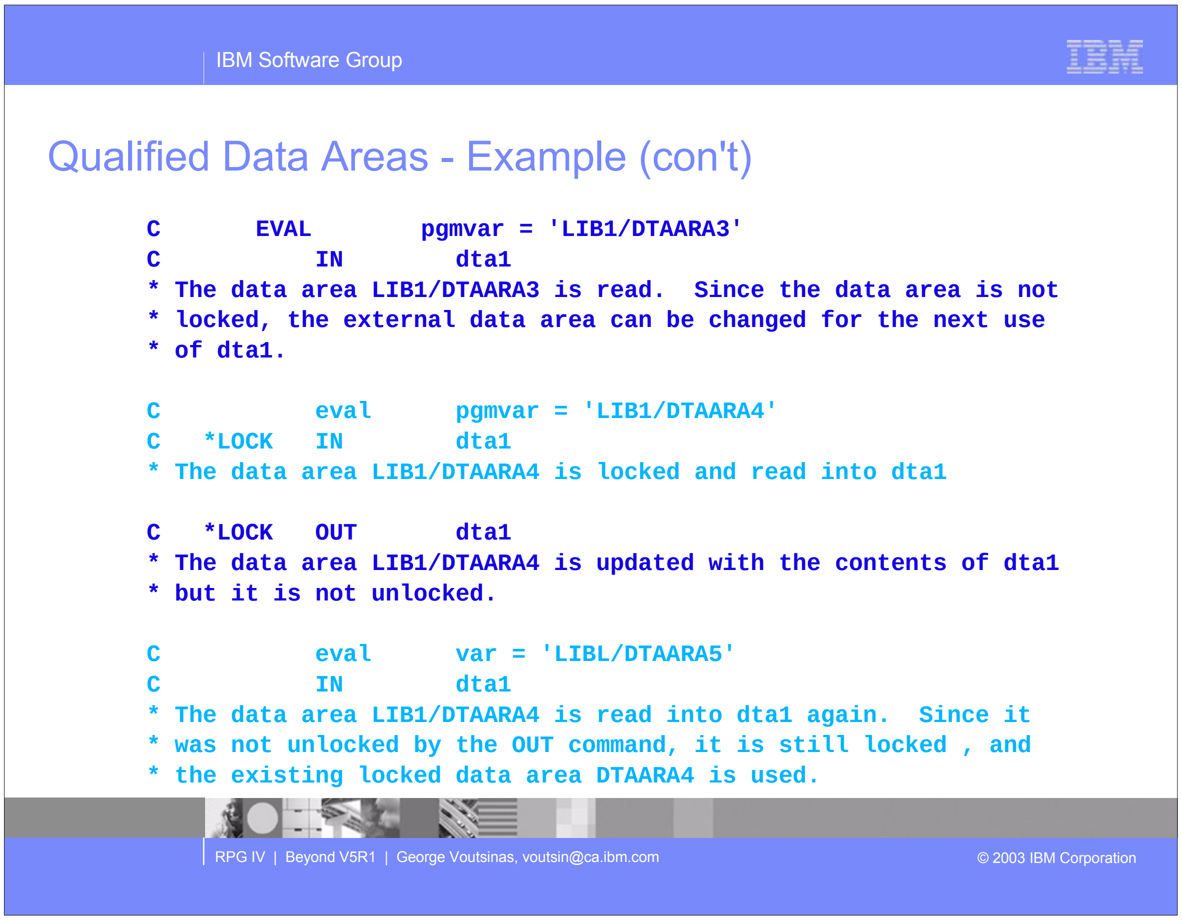
- Name of data area specified in character constant or variable must be in following form:
  - 'NAME'
  - 'LIBRARY/NAME'
  - '*LIBL/NAME'
- valid for single parameter form of DTAARA or using new DTAARA(*VAR:dtaname) form
  - ★ DTAARA keyword used to have ONLY un-quoted name of data area
- You cannot specify *CURLIB as the library name.
- If you specify a data area name without a library name, *LIBL is used.
- The name must be in the correct case.
  - DTAARA(*VAR:dtaname) and dtaname='qtemp/mydta',
  - the data area will not be found. Should be 'QTEMP/MYDTA'.



Qualified Data Areas - Example

```
D dtal             S       10A    DTAARA(*VAR : pgmvar)
D pgmvar           S       21A

C        EVAL     pgmvar = 'LIB1/DTAARA1'
C           IN         dta1
* The data area LIB1/DTAARA1 is read into variable dta1

C           eval      pgmvar = 'LIB1/DTAARA2'
C  *LOCK    IN         dta1
* Data area LIBL1/DTAARA2 is locked, and its contents are read
* into  dta1.  Until data area is unlocked, the data area
* being used by dta1 is fixed.

C           eval      pgmvar = 'LIB1/DTAARA3'
C           IN         dta1
* The data area LIB1/DTAARA2 is read again.  The value of the
* DTAARA variable is ignored, since the data area is already
* locked and in use by the RPG program

C           OUT       dta1
* The data area LIB1/DTAARA2 is updated with the contents of dta1
* and unlocked. Since the data area has been unlocked, the
* external data area can now be changed.
```

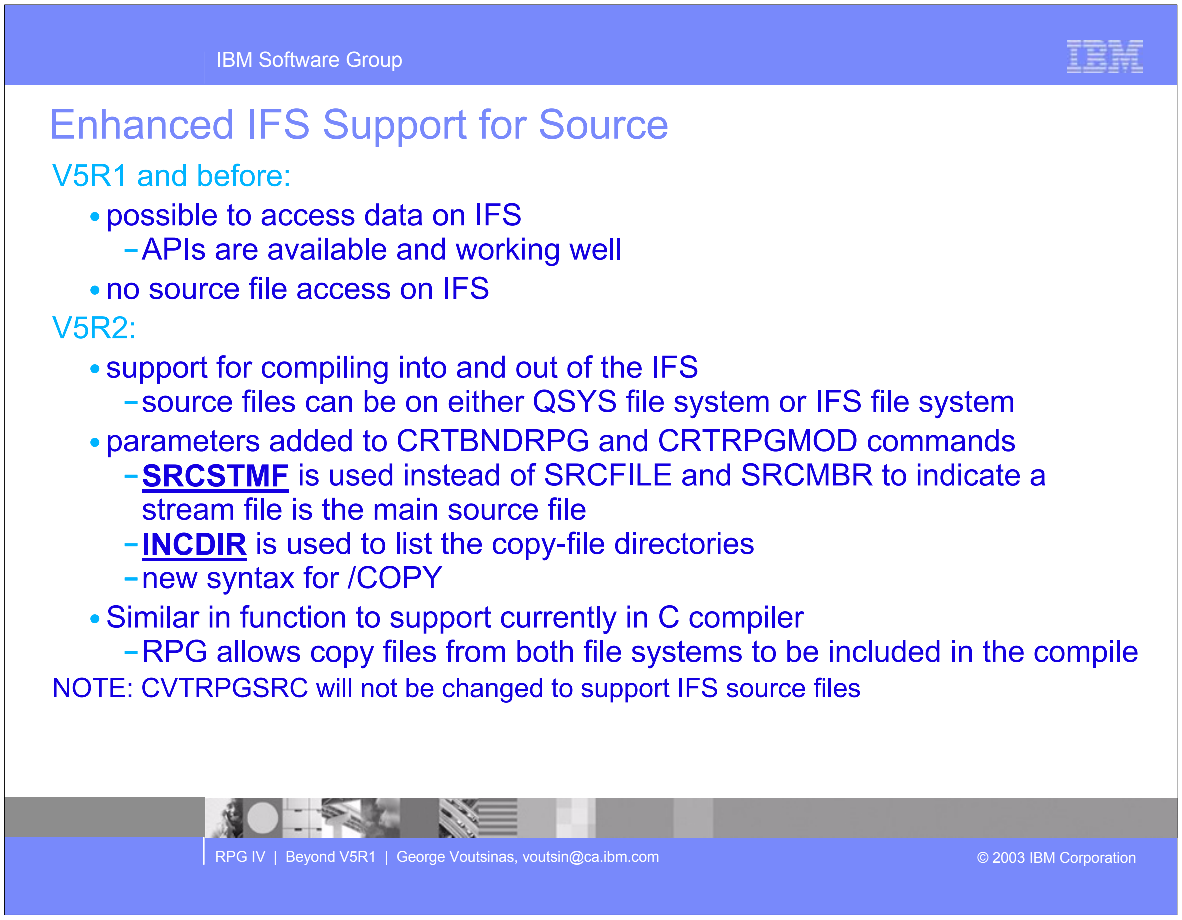
First example is simple, it shows how to dynamically read in a data area.
The next one shows the capabilities of doing the *LOCK on the data area when the IN op-code is used.
Lastly the out op-code writes the data area and unlocks it (If it was locked to start with)



Qualified Data Areas - Example (con't)

```
C       EVAL       pgmvar = 'LIB1/DTAARA3'
C           IN        dta1
* The data area LIB1/DTAARA3 is read.  Since the data area is not
* locked, the external data area can be changed for the next use
* of dta1.

C           eval      pgmvar = 'LIB1/DTAARA4'
C   *LOCK   IN        dta1
* The data area LIB1/DTAARA4 is locked and read into dta1

C   *LOCK   OUT       dta1
* The data area LIB1/DTAARA4 is updated with the contents of dta1
* but it is not unlocked.

C           eval      var = 'LIBL/DTAARA5'
C           IN        dta1
* The data area LIB1/DTAARA4 is read into dta1 again.  Since it
* was not unlocked by the OUT command, it is still locked , and
* the existing locked data area DTAARA4 is used.
```

- Just continuation of the previous example with more variations



Enhanced IFS Support for Source

V5R1 and before:

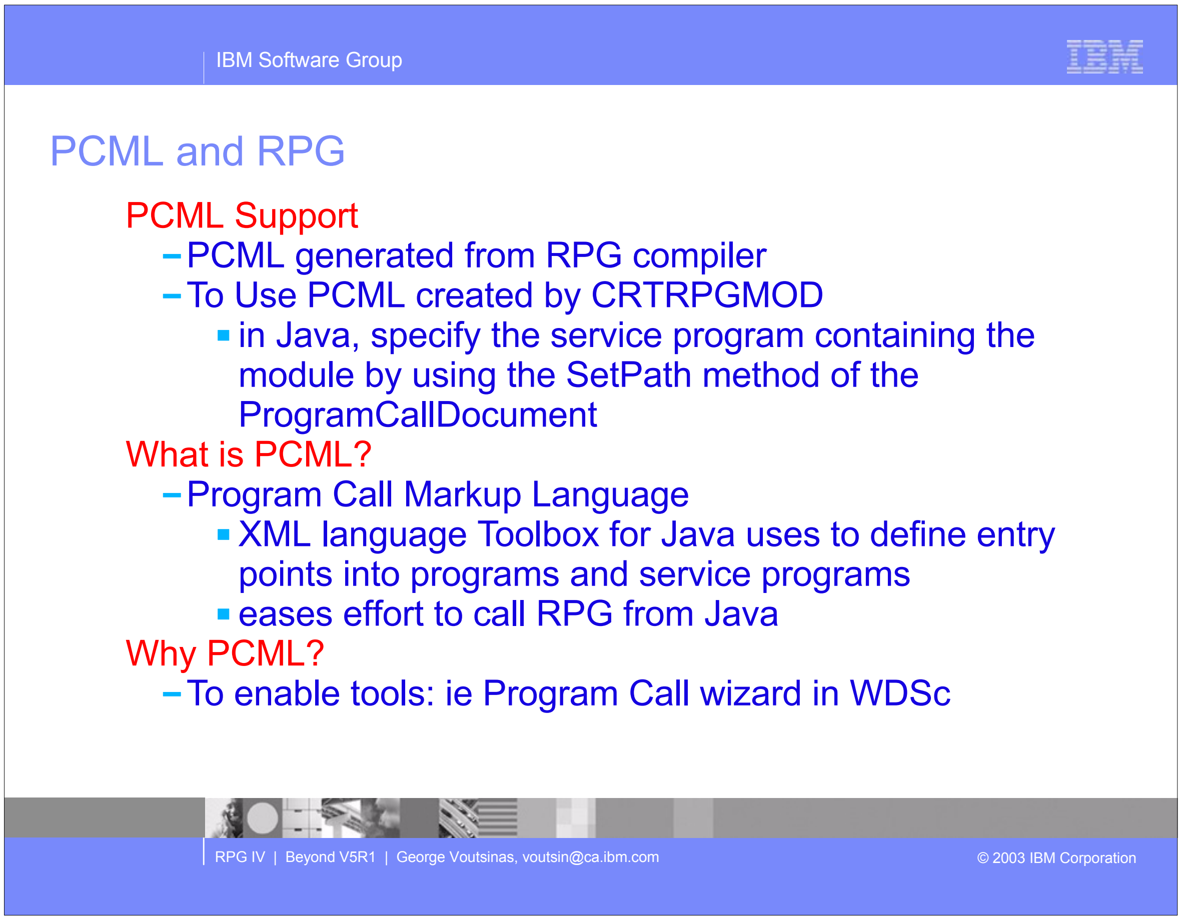
- possible to access data on IFS
  - APIs are available and working well
- no source file access on IFS

V5R2:

- support for compiling into and out of the IFS
  - source files can be on either QSYS file system or IFS file system
- parameters added to CRTBNDRPG and CRTRPGMOD commands
  - **SRCSTMF** is used instead of SRCFILE and SRCMBR to indicate a stream file is the main source file
  - **INCDIR** is used to list the copy-file directories
  - new syntax for /COPY
- Similar in function to support currently in C compiler
  - RPG allows copy files from both file systems to be included in the compile

NOTE: CVTRPGSRC will not be changed to support IFS source files

- These are the rules
- Note that the compiler will allow parameters to specify the name of the IFS file and the directory it resides in. This wil allow the compiler to retrieve the member and compile it.
- This enhancement is only for compiling source members. It does not include reading from IFS data files. For that you still need to use existing system APIs.



PCML and RPG

PCML Support

- PCML generated from RPG compiler
- To Use PCML created by CRTRPGMOD
  - in Java, specify the service program containing the module by using the SetPath method of the ProgramCallDocument

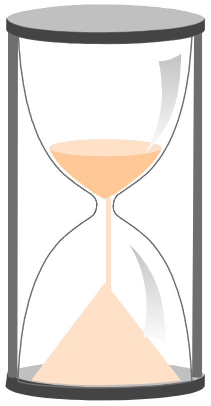
What is PCML?

- Program Call Markup Language
  - XML language Toolbox for Java uses to define entry points into programs and service programs
  - eases effort to call RPG from Java

Why PCML?

- To enable tools: ie Program Call wizard in WDSc

Agenda



- V5R2 Enhancements
  - Enhanced Decimal Precision
  - Conversion of Character to Numeric
  - Data Structure Enhancements
  - Nested Data Structures
  - Improved Keyed I/O Operations
  - Qualified Data Areas
  - Short Form Operations
  - Other Enhancements
- Future
- Summary

Future Enhancement Requests

File and Record:

- Prefix record format names
  - RECPREFIX keyword
- Special Files to handle all I/O opcode including keyed operations
  - extension of the SPECIAL file type
  - allow easier interaction in eBusiness environments

Future Enhancement Requests

Field Definition:

- RANGE and VALUE keywords
  - D Spec?
  - exception given if invalid value assigned
- External Definition to be used - defined for compile time
  - EXTDESC : identify name of externally described file that contains file description
- Allow format change on numerics
  - LIKE keyword

```
D f1          s        8s 0
D f2          s          p      like(fld1)
```

- New BIFs - %TESTT, %TESTD, %TESTZ
- Direct defintion to internal format of ext fields
  - H-Spec keyword EXTDEFS
  - Internal format must match external format
- Full Support for NULL fields - null capable in D-spec
- Allow keyword CONST for variables

Field Editing:

- Option *ZEROFILL on %EDITC to include leading zeros.
- Option to edit negative numbers using parentheses.

Future Enhancement Requests

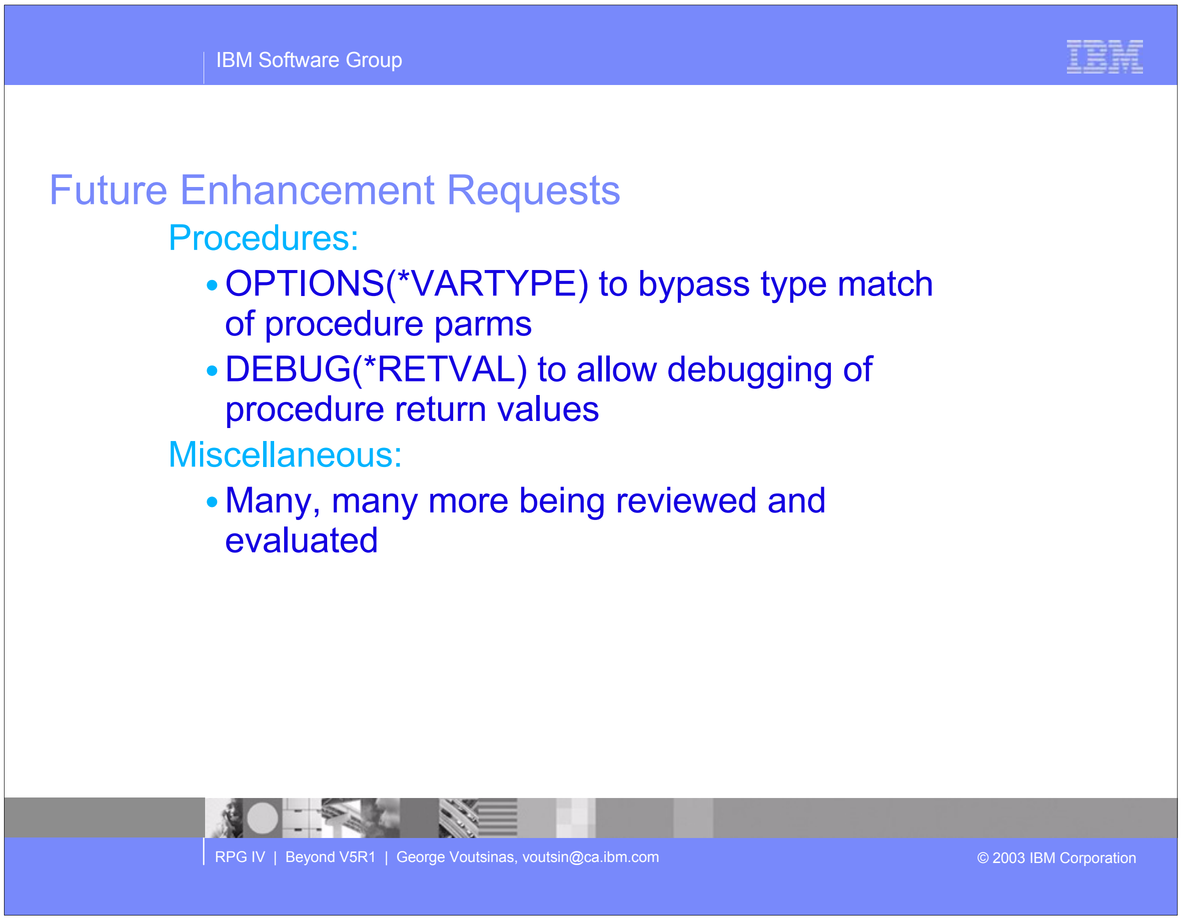
Arrays and DS:

- Multi-dimensional
  - expressions only?
- Dynamic Re-sizing of arrays and MODS
- Multiple array element initializing
  - INZ + DIM
- SORTA
  - SORTA arr1 {: arr2 ...} WITH key_arr1 {: key_arr2 ...}
- Extender on SORTA to reverse the order of sorting
- Retrieve starting position of subfields
  - %OFFSET builtin function returns offset of subfield
- Assign short-name for externally desrcibed data structures
  - extend ALIAS keyword
- Allow named data structure bigger than 64k bytes long
- Allow definiton of DS without allocating storage until specified
  - keyword TEMPLATE
- Runtime level-check for externally-described data structures

Future Enhancements?

Calculation Specs:

- Allow expressions as parameters to keywords.
- Procedure name overloading based on parm types
  - multiple procs with same name
  - selection based on number and type of parameters sent
- Extend INDDS to associate DS with specific record format
  - today one per program buffer area
- Move values from one data structure to another
  - EVALC operation
- Better precision on TIME operation (6 digits right of decimal)
- Built-in function %COND(condition:truevalue:falsevalue)
- Option to pass trimmed string as parameter
- Allow date fields to convert to numeric
  - %INT, %DEC extensions
- Second parameter on %TRIMx - number of chars to trim
- LEAVE-WHEN & ITER-WHEN: single-statement conditional leave or iter.



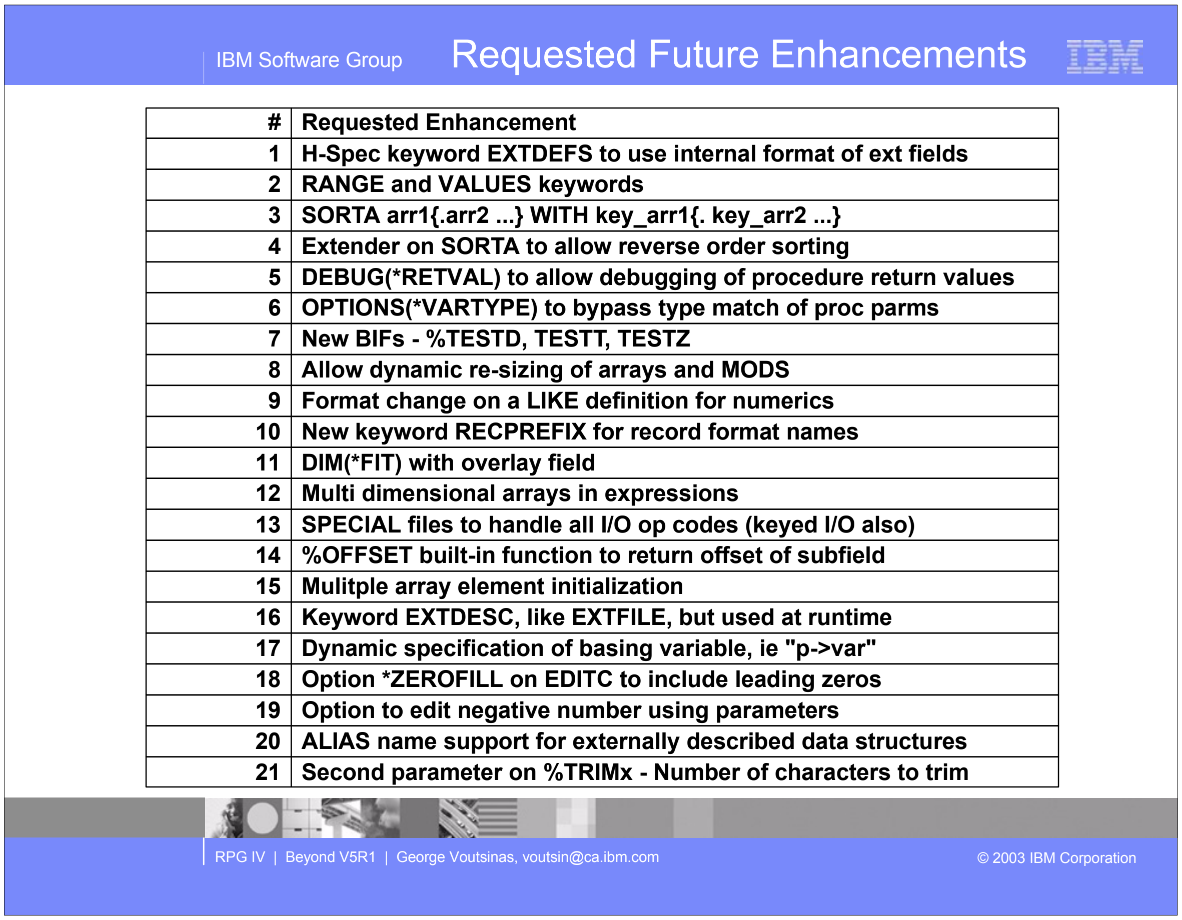
Future Enhancement Requests

Procedures:

- OPTIONS(*VARTYPE) to bypass type match of procedure parms
- DEBUG(*RETVAL) to allow debugging of procedure return values

Miscellaneous:

- Many, many more being reviewed and evaluated

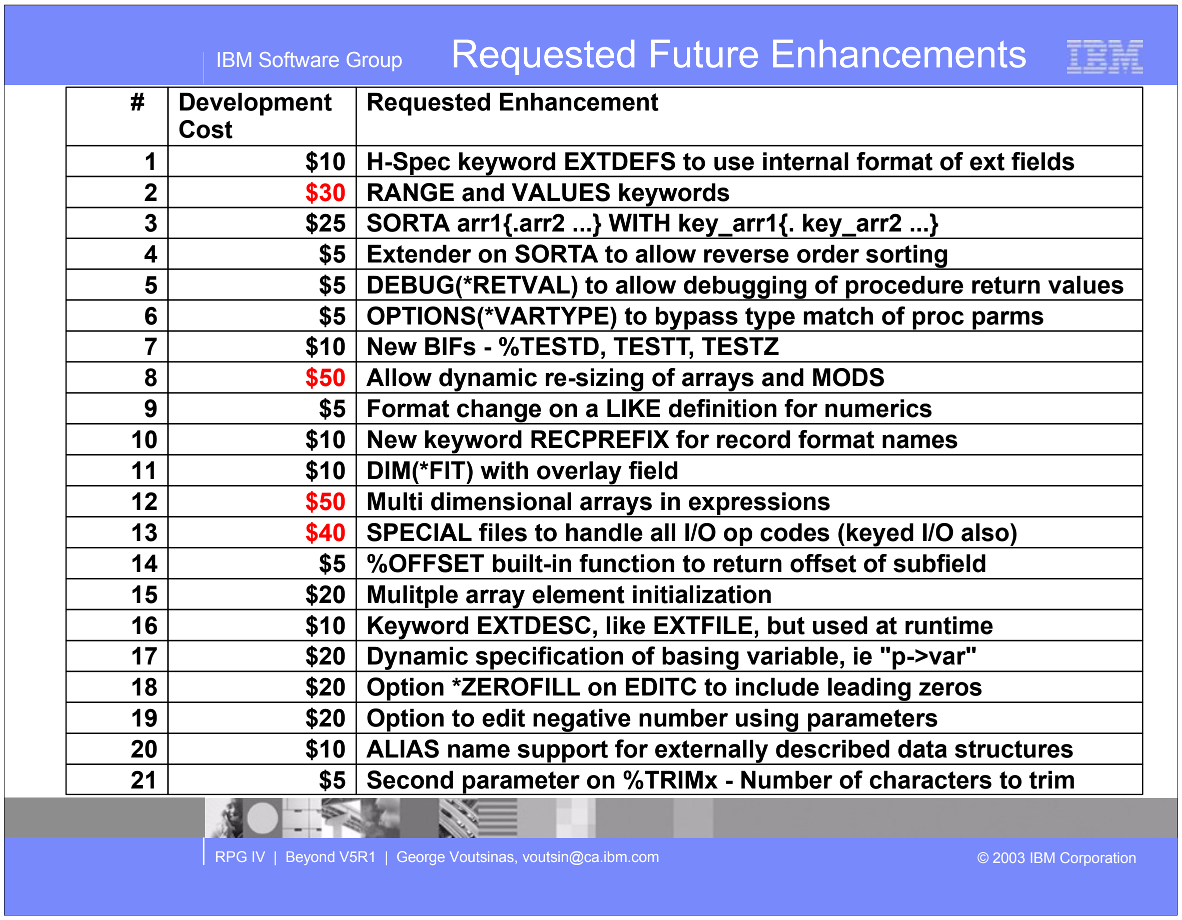


Requested Future Enhancements

#	Requested Enhancement
1	H-Spec keyword EXTDEFS to use internal format of ext fields
2	RANGE and VALUES keywords
3	SORTA arr1{.arr2 ...} WITH key_arr1{. key_arr2 ...}
4	Extender on SORTA to allow reverse order sorting
5	DEBUG(*RETVAL) to allow debugging of procedure return values
6	OPTIONS(*VARTYPE) to bypass type match of proc parms
7	New BIFs - %TESTD, TESTT, TESTZ
8	Allow dynamic re-sizing of arrays and MODS
9	Format change on a LIKE definition for numerics
10	New keyword RECPREFIX for record format names
11	DIM(*FIT) with overlay field
12	Multi dimensional arrays in expressions
13	SPECIAL files to handle all I/O op codes (keyed I/O also)
14	%OFFSET built-in function to return offset of subfield
15	Mulitple array element initialization
16	Keyword EXTDESC, like EXTFILE, but used at runtime
17	Dynamic specification of basing variable, ie "p->var"
18	Option *ZEROFILL on EDITC to include leading zeros
19	Option to edit negative number using parameters
20	ALIAS name support for externally described data structures
21	Second parameter on %TRIMx - Number of characters to trim

Requested Future Enhancements

#	Requested Enhancement
22	Built-in function %COND(condition:truevalue:falsevalue)
23	Allow expression as parameters to keywords
24	Full null field support
25	Procedure name overloading based on parameter types
26	/FREE longer than 80 characters
27	Allow EXTPGM to be coded without a parameter
28	Extend INDDS to associate DS with specific record format
29	Allow named data structure bigger than 64K bytes long
30	Keyword TEMPLET to define DS without allocating storage
31	EVALC move corresponding from one DS to another
32	Allow BASED(%ADDR(something))
33	Allow variable offset for data structure subfields
34	Provide support for inlingin procedures
35	TIME opcode to provide more precise time
36	Additional parameter passing mechanisms
37	Option to pass trimmed string as parameter
38	Allow keyword CONST for variables
39	Conversion of date/time/timestamp to numeric using %INT, %DEC
40	Runtime level check for externally described data structures
41	LEAVE-WHEN and ITER-WHEN: single statement conditional leave or iter



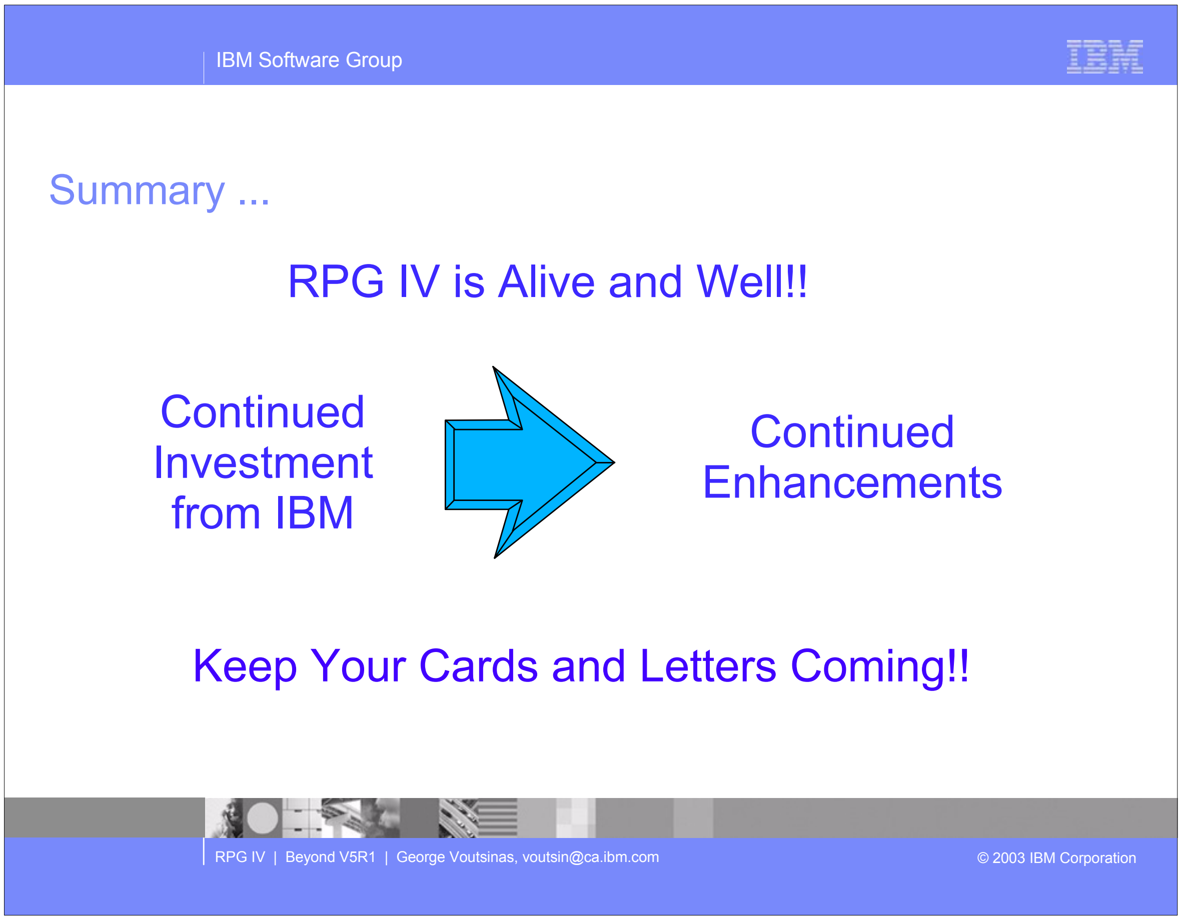
Requested Future Enhancements

#	Development Cost	Requested Enhancement
1	$10	H-Spec keyword EXTDEFS to use internal format of ext fields
2	$30	RANGE and VALUES keywords
3	$25	SORTA arr1{.arr2 ...} WITH key_arr1{. key_arr2 ...}
4	$5	Extender on SORTA to allow reverse order sorting
5	$5	DEBUG(*RETVAL) to allow debugging of procedure return values
6	$5	OPTIONS(*VARTYPE) to bypass type match of proc parms
7	$10	New BIFs - %TESTD, TESTT, TESTZ
8	$50	Allow dynamic re-sizing of arrays and MODS
9	$5	Format change on a LIKE definition for numerics
10	$10	New keyword RECPREFIX for record format names
11	$10	DIM(*FIT) with overlay field
12	$50	Multi dimensional arrays in expressions
13	$40	SPECIAL files to handle all I/O op codes (keyed I/O also)
14	$5	%OFFSET built-in function to return offset of subfield
15	$20	Mulitple array element initialization
16	$10	Keyword EXTDESC, like EXTFILE, but used at runtime
17	$20	Dynamic specification of basing variable, ie "p->var"
18	$20	Option *ZEROFILL on EDITC to include leading zeros
19	$20	Option to edit negative number using parameters
20	$10	ALIAS name support for externally described data structures
21	$5	Second parameter on %TRIMx - Number of characters to trim

This shows the voting list of what customers wanted us to do. This survey had 455 customer replies. Very good results. You see the voting results and what was at the top.
We will be doing the same for the next release and future one.

Requested Future Enhancements

#	Development Cost	Requested Enhancement
22	$10	Built-in function %COND(condition:truevalue:falsevalue)
23	$80	Allow expression as parameters to keywords
24	$80	Full null field support
25	$80	Procedure name overloading based on parameter types
26	$15	/FREE longer than 80 characters
27	$5	Allow EXTPGM to be coded without a parameter
28	$20	Extend INDDS to associate DS with specific record format
29	$20	Allow named data structure bigger than 64K bytes long
30	$10	Keyword TEMPLATE to define DS without allocating storage
31	$20	EVALC move corresponding from one DS to another
32	$10	Allow BASED(%ADDR(something))
33	$20	Allow variable offset for data structure subfields
34	$40	Provide support for inlining procedures
35	$20	TIME opcode to provide more precise time
36	$40	Additional parameter passing mechanisms
37	$5	Option to pass trimmed string as parameter
38	$15	Allow keyword CONST for variables
39	$10	Conversion of date/time/timestamp to numeric using %INT, %DEC
40	$15	Runtime level check for externally described data structures
41	$10	LEAVE-WHEN and ITER-WHEN: single statement conditional leave or iter



Summary ...

RPG IV is Alive and Well!!

Continued Investment from IBM → Continued Enhancements

Keep Your Cards and Letters Coming!!