IBM

Cognitive Systems

What's new for RPG in 7.3 and later TRs

Barbara Morris

Agenda

- The most recent enhancements available with 7.3 PTFs, also for earlier releases
- Other 7.3 enhancements
  - Fully free-form RPG (available as 7.1 & 72 PTFs)
  - Other 7.3 enhancements that were available as 7.1 and 7.2 PTFs
  - Other enhancements new in 7.3

The most recent enhancements available through PTFs

7.1, 7.2 & 7.3 PTF: Compile from Unicode

7.1. 7.2 and 7.3 PTFs - 2016

Previously, it was not possible to compile from Unicode IFS source.

If you edit your source using Orion, the source is in IFS, and depending on your system's CCSID, that source may be Unicode (UTF-8). (For example, in Japan)

You may prefer to have Unicode source for other reasons.

A new TGTCCSID parameter has been added to CRTBNDRPG and CRTRPGMOD that lets you tell the compiler what CCSID to use to compile your source.

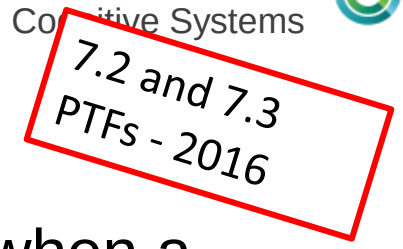
7.1, 7.2 & 7.3 PTF: Compile from Unicode

- TGTCCSID(*SRC) – the default, can't be used with Unicode
- TGTCCSID(*JOB) – compile using the job CCSID
- TGTCCSID(number) – for example TGTCCSID(37)

For CRTSQLRPGI, use the COMPILEOPT parameter along with RPGPPOPT(*LVL1) or *LVL2

```
CRTSQLRPGI MYLIB/MYPGM SRCSTMF(mypgm.sqlrpgle)
                       RPGPPOPT(*LVL1)
                       COMPILEOPT('TGTCCSID(*JOB)')
```

7.2 & 7.3 PTF: ON-EXIT



Sometimes you want to run some cleanup code when a procedure ends

- You allocated some storage and you want to deallocate it
- You created a temporary file and you want to delete it
- I'm sure you can think of a million things

Now, you can put that code into the ON-EXIT section of your procedure, and the code will run no matter how your procedure ends

7.2 & 7.3 PTF: ON-EXIT

```
dcl-proc myproc;
   dcl-s p pointer;

   p = %alloc(10);
   ...
   average = total / num;
   ...

   on-exit;
      dealloc p;

   end-proc;
```

This code always runs.

The pointer is deallocated even if NUM is zero for the division

7.2 & 7.3 PTF: ON-EXIT

Your procedure might end

- Because it reached the end of its code
- Because it reached a RETURN operation
- Because it crashed due to an unhandled exception
- Because the subsystem ended
- Because you pressed F3 while debugging
- …

No matter why your procedure ended, the code in the ON-EXIT section will run

7.2 & 7.3 PTF: ON-EXIT

When your procedure returns a value, it can return a new value within the ON-EXIT section

If a RETURN operation is not reached in the ON-EXIT section, the original return value is used

```
x = myproc2 (*off);
```

doReturn is off, so x = 5

```
x = myproc2 (*on);
```

doReturn is on, so x = 6

```
dcl-proc myproc2;
   dcl-pi *n packed(5);
      doReturn ind const;
   end-pi;
   return 5;
on-exit;
   if doReturn;
      return 6;
   endif;
end-proc;
```

7.2 & 7.3 PTF: Nested data structures

7.2 and 7.3 PTFs - 2017

You can now code data structure subfields directly when using free-form declarations

Before

```
dcl-ds orders_t qualified template;
   id char(10);
   price packed(7 : 2);
   quantity int(10);
end-ds;

dcl-ds order_info qualified;
   num_orders int(10);
   orders likeds(orders_t) dim(100);
   discount packed(7 : 2);
end-ds;
```

Now

```
dcl-ds order_info qualified;
   num_orders int(10);
   dcl-ds orders dim(100);
      id char(10);
      price packed(7 : 2);
      quantity int(10);
   end-ds;
   discount packed(7 : 2);
end-ds;
```

7.2 & 7.3 PTF: Nested data structures

- Nested data structures can only be defined in free-form. LIKEDS must still be used for fixed-form definitions
- The top-level data structure must be qualified
- The nested data structure subfields are automatically qualified

```
if order_info.orders(3).quantity > MAX_QUANTITY;
...
```

7.2 & 7.3 PTF: Nested data structures

There will still be cases where you will use LIKEDS to define data structure subfields

- When you use the same structure for several different subfields
- When the data structure subfield is also used as a top-level data structure
- …

But now you have a choice!

7.2 & 7.3 PTF: %MAX and %MIN

7.2 and 7.3 PTFs - 2017

New built-in functions %MAX and %MIN

```
max_dim = %MIN(length : height : width);

earliest_move_in_date = %MIN(electrical_ready
                           : appliances_ready
                           : painting_done
                           : furniture_delivered);
```

In calculations, there must be at least two operands, and they can have any type as long as they can be compared to each other. There is no maximum number of operands.

7.2 & 7.3 PTF: %MAX and %MIN

In keywords, there can only be two operands and they must be numeric.

```
dcl-ds ds1 extname('FILE1') end-ds;
dcl-ds ds2 extname('FILE2') end-ds;

// Define the file with a record length big enough
// to handle data from both FILE1 and FILE2
dcl-f testfile disk(%MAX(%size(ds1)
                         : %size(ds2)));
```

To handle more than two operands in keywords, nest %MAX:

```
dcl-c max %MAX(a
               : %MAX(b
                      : %MAX(c : d)));
```

7.2 & 7.3 PTF: ALIGN(*FULL)

7.2 and 7.3 PTFs - 2017

Use ALIGN(*FULL) to define a data structure the way a C API defines it

Background

The ALIGN keyword causes RPG to position integer and float subfields so that they are aligned on an x-byte storage boundary, where x is the size of the subfield

Pointer subfields are always aligned on a 16-byte boundary

An array of data structures with aligned subfields sometimes has some padding bytes added at the end

7.2 & 7.3 PTF: ALIGN(*FULL)

Background continued

Consider this data structure

```
dcl-ds ds1 dim(2) ALIGN qualified;
   int_subf int(10);
   char_subf char(1);
end-ds;
```

The “int_subf” subfield must be aligned on a 4-byte boundary, so 3 bytes of padding is needed at the end of each element.

Unfortunately, %SIZE(ds1) returns 5, not 8.

To get the actual size of each element, use this ugly code

%SIZE(ds1 : *ALL) / %ELEM(ds1)

7.2 & 7.3 PTF: ALIGN(*FULL)

Use ALIGN(*FULL) to have %SIZE reflect the true size of each element

- Consider this data structure

```
dcl-ds ds1 dim(2) ALIGN(*FULL) qualified;
   int_subf int(10);
   char_subf char(1);
end-ds;
```

- Now, %SIZE(ds1) returns 8.

7.2 & 7.3 PTF: ALIGN(*FULL)

What about ordinary data structures that aren't arrays?

In that case, the data structure does not have the padding bytes at the end.

```
dcl-ds ds1 ALIGN;
   int_subf int(10);
   char_subf char(1);
end-ds;
```

The actual size of the data structure is 5.

The problem

If this data structure is passed to a C API that expects an aligned data structure, the API might expect the data structure to have a length of 8.

Storage corruption!

7.2 & 7.3 PTF: ALIGN(*FULL)

This is a real issue with the regular-expression APIs. When an RPG programmer codes a data structure to match the regex_t structure in the C include regex.h, the RPG version of the data structure is 12 bytes too short.

With ALIGN(*FULL), the RPG data structure is correct.

7.2 & 7.3 PTF: ALIGN(*FULL)

Recommendations

- Whenever you code ALIGN, always code ALIGN(*FULL).
- If your data structure has pointers so you don’t need the ALIGN keyword, code ALIGN(*FULL) if
  - You are passing the data structure to a program written in C
  - The data structure is an array and you are using %SIZE to find out the distance between the elements

Fully free-form RPG

Fully free-form RPG

7.1 and 7.2 PTFs

Now it is possible to code free-form RPG starting in column 1 and going to the end of the line.

There is no practical limit on the length of a source line.

Fully free-form RPG – source must start with **FREE

Any source member that contains fully-free code must have **FREE in column 1 of the first line of the source.

```
**free

ctl-opt main(greeting);

dcl-proc greeting;
   dsply 'Hello';
end-proc;
```

Fully free-form RPG

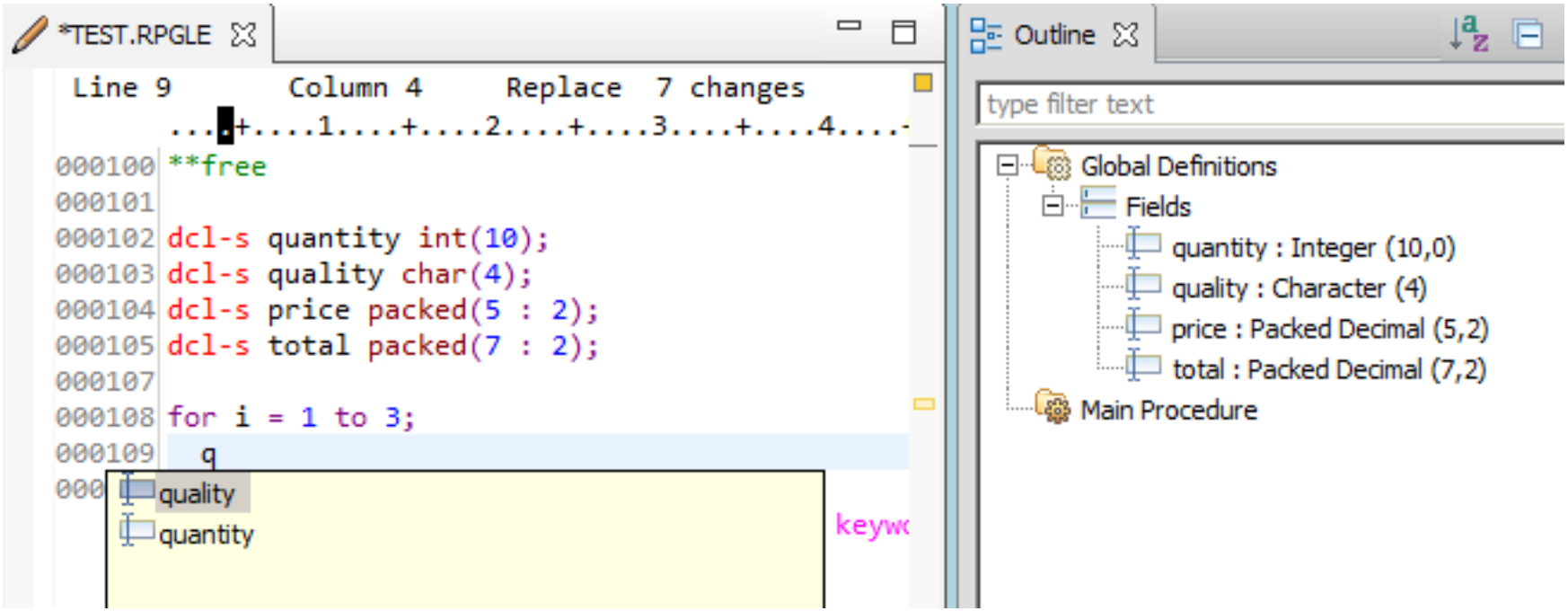
- All code in a **FREE source member must be free-form. If you need any fixed-form code, you can put it in a /COPY file
- Source lines must not begin with ** unless they are the special directives for compile-time data, file-translation, or alternate collating sequence.
- /FREE and /END-FREE are not allowed in a **FREE source member

Fully free-form RPG – copy files

- Each copy file has its own source mode
- A copy file is always assumed to have column-limited source mode unless it has **FREE in line 1

Fully free-form RPG – RDI

RDI V9.5 supports fully-free RPG code



Fully free-form RPG – Embedded SQL

The SQL precompiler supports fully-free RPG code

```
**free

dcl-s greeting char(10);
exec sql set :greeting = 'Hello';
dsply greeting;
return;
```

Other 7.3 enhancements that were originally 7.1 and 7.2 PTFs

7.1 & 7.2 PTF: ALIAS is supported for all files

7.1 and 7.2 PTFs

Previously, using the alias names was not available for global files unless the F spec had the QUALIFIED keyword.
Now, the alias names are available for any externally-described file.

```
A          R CUSTREC
A            CUSTNM         25A          ALIAS(CUSTOMER_NAME)
A            CUSTAD         25A          ALIAS(CUSTOMER_ADDRESS)
A            ID             10P 0

Fmyfile    o    e             DISK       ALIAS

   customer_name = 'John Smith';
   customer_address = '123 Mockingbird Lane';
   id = 12345;
   write myfmt;
```

When there is no alternate name for a field, the short name is used.

7.1 & 7.2 PTF: Support for ALIAS names – a limitation

Limitation:

If you code ALIAS for a file, you can't code I specs or O specs for that file.

The compiler will still generate I and O specs into the listing. If the name is too long to fit in the generated spec, the name will be listed on the next line.

```
12=O                            COMPANY                  6A  CHAR
13=O                            *ALIAS                  31A  CHAR
       MAILING_ADDRESS
14=O                            STATUS                   2A  CHAR
```

7.1 & 7.2 PTF : Easier to use data structures for I/O

The problem:

RPG was very strict about which data structures could be use for I/O:

- For an input operation, it had to be defined with *INPUT (the default for LIKEREC)
- For a WRITE operation, it had to be defined with *OUTPUT
- For an UPDATE operation, it could be defined with either *INPUT or *OUTPUT

7.1 & 7.2 PTF: Easier to use data structures for I/O

Before:

For a file that allowed both READ and WRITE, it could be very awkward to use data structures for I/O to the file.

```
dcl-f myfile usage(*input : *output);

dcl-ds inDs likerec(myfmt : *input);
dcl-ds outDs likerec(myfmt : *output);

read myfmt inDs;
...
eval-corr outDs = inDs;  // copy over the fields
write outDs;
```

7.1 & 7.2 PTF: Easier to use data structures for I/O

Now, with new 7.1 and 7.2 PTFs

Now, if you use LIKEREC with no type parameter for a DISK record, you can use the data structure for any operation.

```
dcl-f myfile usage(*input : *output);
dcl-ds ds likerec(myfmt);

read myfmt ds;
...
write ds;
```

(The same PTFs as the ALIAS PTFs)

7.1 & 7.2 PTF: Easier to use data structures for I/O

You can also use a LIKEREC data structure with no type parameter for a PRINTER file.

```
dcl-f myprtf printer;
dcl-ds ds likerec(myprtfmt);

write myprtfmt ds;
```

7.1 & 7.2 PTF: Easier to use data structures for I/O

If the data structure is defined with *ALL (E-DS or LIKEREC), you can use it for any I/O operation.

```
dcl-f diskf usage(*update : *output);
dcl-f prtf printer;

dcl-ds diskDs extname('DISKF' : *all) end-ds;
dcl-ds prtDs extname('PRTF' : *all) end-ds;

read diskfmt diskDs;
write diskfmt diskDs;
update diskfmt diskDs;

write prtfmt prtDs;
```

(*ALL was already supported in 6.1 for WORKSTN files, including subfile formats)

7.1 & 7.2 PTF: PCML with mixed-case names

By default, the RPG compiler generates PCML with the names in uppercase.

```
dcl-proc newOrder export;
  dcl-pi *n;
    qty packed(15) const;
    itemName car(30) const;
```

```
<pcml version="4.0">
 <program name="NEWORDER" entrypoint="NEWORDER">
  <data name="QTY" type="packed" length="15" ...
  <data name="ITEMNAME" type="char" length="30"
...
```

Anything using the PCML must also use the uppercase names:

```
pcd.setValue ("NEWORDER.QTY", ... );
pcd.setValue ("NEWORDER.ITEMNAME", ... );
pcd.callProgram ("NEWORDER");
```

7.1 & 7.2 PTF: PCML with mixed-case names

New: Specify PGMINFO(*DCLCASE) in the H spec to have the PCML generated with the same case as the RPG source

The generated PCML:

```
<pcml version="4.0">
 <program name="newOrder" entrypoint="NEWORDER">
  <data name="qty" type="packed" length="15" ...
  <data name="itemName" type="char" length="30" ...
```

The Java code using the PCML can use the same mixed-case names:

```
pcd.setValue ("newOrder.qty", ... );
pcd.setValue ("newOrder.itemName", ... );
pcd.callProgram ("newOrder");
```

7.1 & 7.2 PTF: More granular PCML

7.1 and 7.2 PTFs

By default, the RPG compiler generates PCML for all exported procedures.

Some procedures have parameter or return value types that make it impossible to generate PCML.

This causes the compile to fail.

7.1 & 7.2 PTF:

New: P-spec keyword PGMINFO(*YES | *NO)

Either

- Specify PGMINFO(*YES) for all the procedures that should have PCML generated

Or

- Specify PGMINFO(*NO) for all the procedures that should not have PCML generated

These PGMINFO keywords are ignored if PCML is not being generated.

7.1 & 7.2 PTF: DCLOPT(*NOCHGDSLEN)

7.1 and 7.2 PTFs

New: H-spec keyword DCLOPT(*NOCHGDSLEN)

By default, RPG has a feature that allows the length of a data structure to be set by a C spec or program-described I spec, or by having a field in an externally-described file with the same name as the data structure.

```
D ds1             ds
D   fld1                      10a

C                   move      'x'           ds1                 20
```

In the example above, it looks like data structure DS1 should have a length of 10. But the assignment to DS1 in the MOVE operation sets the length to 20.

This means that a data structure cannot be considered to be “defined” until the end of the procedure.

7.1 & 7.2 PTF: DCLOPT(*NOCHGDSLEN)

Free-form declaration statements can use built-in functions as parameters to keywords, as long as the built-in function is defined.

```
dcl-ds ds1;
  fld1 char(10);
end-ds;
dcl-s fld2 char(%size(ds1)); // not valid
```

RNF3320: The keyword parameter is not defined; keyword is ignored.

When %size is encountered, data structure DS1 is not yet defined, so %size cannot be used with the CHAR keyword.

7.1 & 7.2 PTF: DCLOPT(*NOCHGDSLEN)

When new keyword DCLOPT(*NOCHGDSLEN) is added to the Control options, data structure DS1 is defined as soon as all its subfields are defined, and %SIZE can be used.

```
ctl-opt dclopt(*nochgdslen);

dcl-ds ds1;
  fld1 char(10);
end-ds;
dcl-s fld2 char(%size(ds1)); // valid
```

Other enhancements new in 7.3

%SCANR (Scan reverse)

```
%SCANR(search argument : source string { : start { : length } } )
```

- %SCANR is similar to %SCAN. They differ only in the direction of the search
- The “start position” represents the beginning of the substring to be searched
- %SCANR starts at the end of the string and searches backwards until it finds a match or it reaches the start position
- The start and length parameters represent a substring in the source string. The “start” parameter doesn’t indicate the starting position for the search.

New “length” parameter for %SCAN

```
%SCAN(search argument : source string { : start { : length } } )
```

- An optional “length” parameter has been added to %SCAN. It indicates the length to search in the source string.
- %SCAN and %SCANR are identical except for the direction of the search within the substring identified by the “source string”, “start”, and “length” parameters.

%SCANR example

Find the filename in a path:

```
path = '/home/mydir/other/whatever/a.txt';
lastSlash = %SCANR('/' : path);
if lastSlash = 0;
     fileName = path;
else;
     fileName = %subst(path : lastSlash + 1);
endif;
```

```
lastSlash = 27
fileName = 'a.txt‘
```

%SCANR example with a start position

Find the filename suffix. First, find the last slash. Then find the last period in the substring starting at the last slash

```
path = '/home/mydir/other/what.ever/myfile';
p = %SCANR('/' : path);
if p = 0;
     p = 1; // start the next search at the beginning
endif;

dot = %SCANR('.' : path : p); // search “/myfile”
if dot = 0;
     suffix = '';
else;
     suffix = %subst(path : dot + 1);
endif;
```

`suffix = ''` because there is no period following the last slash

%SCANR example with a length

Find the final directory: First, find the last slash. Then find the previous slash

```
   path = '/home/mydir/v2/myfile';
   p = %SCANR('/' : path);     // last slash
   if p = 0;
      finalDir = '';           // path has no directory
   else;
      prv = %SCANR('/' : path : 1 : p - 1); // length to search is "p – 1"
      if prv = 0;              // path starts with the final directory
         finalDir = %SUBST(path : 1 : p – 1);
      else;                    // path has more than one directory
         finalDir = %SUBST(path : prv + 1 : p - prv - 1);
      endif;
   endif;
finalDir = "v2"
```

%SCAN and %SCANR together

Find the last word in a quoted section of a string.

```
string = 'The message is "The file was not found". The sender is X.';
      //  ....+....1....+....2....+....3....+....4....+....5....+...

// Find the quoted string within the string

p1 = %SCAN('"' : string);                    // Start quote (16)
p2 = %SCANR('"' : string : p1 + 1);          // End quote (39)
len = p2 - p1 - 1;                           // Length between quotes (22)

// Find the last word in the quoted string

p3 = %SCANR(' ' : string : p1 + 1 : len);    // Last blank between "" (33)
len = p2 - p3 - 1;                           // Length after last blank (5)
lastword = %subst(string : p3 + 1 : len);    // Last word ("found")
```

Increased maximum parameters for a bound call

Formerly, the maximum number of parameters for a bound call was 399.

The new maximum is 16,382.

Note that this is a general ILE enhancement that RPG supports. It is unlikely to be of interest to RPG programmers, since it is unlikely that RPG programmers have ever even run into the previous limit of 399.

Null-capable fields (background)

A null-capable field has

- Its value, 3.2, 'Jack Sprat' etc.
- An associated value that says whether it is null or not, called a "null indicator"

The I/O buffer for the file has a separate section called the "null-byte map" which has an indicator for each field in the file indicating whether it is null or not.

(If the field is not null-capable, the null-byte-map indicator for that field is always '0'.)

Null-capable fields (background)

Imagine a file with three fields

- NAME: not null-capable
- DUEDATE: null-capable
- PRVBAL: null-capable

Here is the I/O buffer for a sample record

```
Buffer: Jack Sprat      0001-01-010041.75

Null-byte map: 010
```

Null-capable field "DUEDATE" has the null-value. Its value of 0001-01-01 is meaningless.

Null-capable fields (background)

When you code the ALWNULL(*USRCTL) keyword in an RPG module, you can work with the null-capable fields in your file

The associated value is an internal variable maintained by the RPG compiler.

- Prior to 7.3, this was always the case.
- Starting in 7.3 this is the default

You refer to the null-indicator using the %NULLIND built-in function

```
dueDate = curDate + %days(30);
%nullind(dueDate) = *off;  // not null
```

Null-capable fields (background)

When you read a record containing null-capable fields

- The buffer values get moved into the program fields
- The null-byte map values get moved into the associated null-indicators of the null-capable fields

```
Buffer: Jack Sprat      0001-01-010041.75
Null-byte map: 010
```

```
ctl-opt alwnull(*usrctl);

dcl-f custfile;
read custrec;
```

```
> EVAL name
  NAME = 'Jack Sprat    '
> EVAL duedate
  DUEDATE = '0001-01-01'
> EVAL _QRNU_NULL_DUEDATE
  _QRNU_NULL_DUEDATE = '1'
> EVAL PRVBAL
  PRVBAL = 41.75
> EVAL _QRNU_NULL_PRVBAL
  _QRNU_NULL_PRVBAL = '0'
```

Null-capable fields (background)

When you write or update a record containing null-capable fields

- The program field values get moved into the buffer
- The associated null-indicators of the null-capable fields get moved into the null-byte map. The null-byte map for the non-null-capable fields is set to '0'.

```
dueDate = curDate + %days(30);
%nullind(dueDate) = *off;  // not null
update custrec;
```

```
Buffer: Jack Sprat      2016-06-150041.75
Null-byte map: 000
```

Null-capable fields (background)

Prior to 7.3, the only null-capable fields available for RPG were

- From externally-described files
- From externally-described data structures

You could not define your own null-capable fields

NULLIND: More control over null indicators

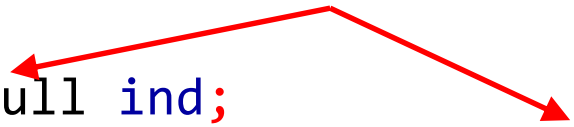
Use the NULLIND keyword without a parameter to define the field as null-capable.

```
dcl-s qty int(10) nullind;
```

Field qty is null-capable, but the null-indicator is still maintained internally by the RPG compiler.

NULLIND: More control over null indicators

Use the NULLIND keyword to associate your own indicator with another field to represent whether that field is null.



```
dcl-s qty_is_null ind;
dcl-s qty int(10) nullind (qty_is_null);
```

You can refer to the null indicator using its name, or using %NULLIND. These mean the same thing:

```
if qty_is_null;

if %nullind(qty);
```

NULLIND with a data structure

Use the NULLIND keyword to associate a data structure of null indicators with a data structure to represent whether the subfields are null.

The NULLIND data structure represents the null-byte map for the other data structure.

The next enhancement allows you to easily define the data structure of null indicators …

LIKEREC(rec:*NULL) and EXTNAME(file:*NULL)

*NULL defines a null-byte map data structure representing whether the matching fields or subfields are null.

The subfields have the same names as the fields in the file, but all the subfields are indicators.

The NULLIND keyword associates cust_null as the null-byte map for cust_ds.

```
dcl-ds cust_null likerec(custrec : *null);
dcl-ds cust_ds   likerec(custrec)
                 nullind(cust_null);

read custfile cust_ds;
if not cust_null.duedate; // duedate is not null
```

LIKEREC(rec:*NULL) and EXTNAME(file:*NULL)

If the main data structure has a specific extract type (*INPUT etc), define the nullind data structure the same way, adding *NULL.

```
dcl-ds cust_ds   likerec(custrec : *output) nullind(cust_null);
dcl-ds cust_null likerec(custrec : *output : *null);
```

Here, the two data structures represent the output record format.

*NULL – easier to work with trigger parameters

In trigger programs, there is a null byte map for the before and after record.

Before (error prone):

```
dcl-s nullmap1 char(100) based(pNullmap1);

pNullmap1 = %addr(trigger_buffer) + null_offset1;
if %subst(nullmap1 : 3 : 1) = '1';
```

Now:

```
dcl-ds nullmap1 extname('CUSTFILE':*NULL)
                qualified based(pNullmap1) end-ds;

pNullmap1 = %addr(trigger_buffer) + before_null_offset;
if nullmap1.duedate;
```

Bonus: duedate is an indicator, so there is no need to compare it to '1' now.

Where to find out about PTF enhancements for RPG

- Subscribe to the blog in the RPG Cafe
  - Whenever we provide an enhancement through PTFs, we post a blog entry in the Cafe blog
  - The blog entry will usually point to a new page in the Cafe wiki
- Regularly check the “welcome” page in the RPG Cafe wiki. There is a section for “Enhancements delivered through PTFs”, and the “Announcement” section at the top will have information about the most recent enhancement.
- Regularly check the “What’s New Since …” section in the ILE RPG Reference
  - Starting in 7.2, the ILE RPG manuals are updated with enhancements delivered through PTFs. If there are PTFs for 7.1, refer to the 7.2 documentation for that enhancement.

Where to request enhancements for RPG

RPG is now part of the RFE (Request for Enhancements) process.

- You can submit requirements
- You can vote on requirements that others have requested
  - Votes aren't the only consideration when IBM decides which RFEs to work on, but they are important
  - Be careful not to vote for too many RFEs. Just vote on the ones that you need the most
- The TGTCCSID enhancement was RPG's first delivered RFE.

Here is a link to the current RFEs for the RPG compiler:
http://ibm.biz/rpg_rfe

Brand:	Servers and Systems Software
Product family:	Power Systems
Product:	IBM i
Component:	Languages - RPG

Go to IBM

Special notices

This document was developed for IBM offerings in the United States as of the date of publication. IBM may not make these offerings available in other countries, and the information is subject to change without notice. Consult your local IBM business contact for information on the IBM offerings available in your area.

Information in this document concerning non-IBM products was obtained from the suppliers of these products or other public sources. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. Send license inquires, in writing, to IBM Director of Licensing, IBM Corporation, New Castle Drive, Armonk, NY 10504-1785 USA.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

The information contained in this document has not been submitted to any formal IBM test and is provided "AS IS" with no warranties or guarantees either expressed or implied.

All examples cited or described in this document are presented as illustrations of the manner in which some IBM products can be used and the results that may be achieved. Actual environmental costs and performance characteristics will vary depending on individual client configurations and conditions.

IBM Global Financing offerings are provided through IBM Credit Corporation in the United States and other IBM subsidiaries and divisions worldwide to qualified commercial and government clients. Rates are based on a client's credit rating, financing terms, offering type, equipment type and options, and may vary by country. Other restrictions may apply. Rates and offerings are subject to change, extension or withdrawal without notice.

IBM is not responsible for printing errors in this document that result in pricing or information inaccuracies.

All prices shown are IBM's United States suggested list prices and are subject to change without notice; reseller prices may vary.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

Any performance data contained in this document was determined in a controlled environment. Actual results may vary significantly and are dependent on many factors including system hardware configuration and software design and configuration. Some measurements quoted in this document may have been made on development-level systems. There is no guarantee these measurements will be the same on generally-available systems. Some measurements quoted in this document may have been estimated through extrapolation. Users of this document should verify the applicable data for their specific environment.

Special notices (cont.)

IBM, the IBM logo, ibm.com AIX, AIX (logo), AIX 5L, AIX 6 (logo), AS/400, BladeCenter, Blue Gene, ClusterProven, DB2, ESCON, i5/OS, i5/OS (logo), IBM Business Partner (logo), IntelliStation, LoadLeveler, Lotus, Lotus Notes, Notes, Operating System/400, OS/400, PartnerLink, PartnerWorld, PowerPC, pSeries, Rational, RISC System/6000, RS/6000, THINK, Tivoli, Tivoli (logo), Tivoli Management Environment, WebSphere, xSeries, z/OS, zSeries, Active Memory, Balanced Warehouse, CacheFlow, Cool Blue, IBM Systems Director VMControl, pureScale, TurboCore, Chiphopper, Cloudscape, DB2 Universal Database, DS4000, DS6000, DS8000, EnergyScale, Enterprise Workload Manager, General Parallel File System, , GPFS, HACMP, HACMP/6000, HASM, IBM Systems Director Active Energy Manager, iSeries, Micro-Partitioning, POWER, PowerExecutive, PowerVM, PowerVM (logo), PowerHA, Power Architecture, Power Everywhere, Power Family, POWER Hypervisor, Power Systems, Power Systems (logo), Power Systems Software, Power Systems Software (logo), POWER2, POWER3, POWER4, POWER4+, POWER5, POWER5+, POWER6, POWER6+, POWER7, System i, System p, System p5, System Storage, System z, TME 10, Workload Partitions Manager and X-Architecture are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries.

A full list of U.S. trademarks owned by IBM may be found at: http://www.**ibm.com**/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

AltiVec is a trademark of Freescale Semiconductor, Inc.

AMD Opteron is a trademark of Advanced Micro Devices, Inc.

InfiniBand, InfiniBand Trade Association and the InfiniBand design marks are trademarks and/or service marks of the InfiniBand Trade Association.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries or both.

Microsoft, Windows and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries or both.

NetBench is a registered trademark of Ziff Davis Media in the United States, other countries or both.

SPECint, SPECfp, SPECjbb, SPECweb, SPECjAppServer, SPEC OMP, SPECviewperf, SPECapc, SPEChpc, SPECjvm, SPECmail, SPECimap and SPECsfs are trademarks of the Standard Performance Evaluation Corp (SPEC).

The Power Architecture and Power.org wordmarks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

TPC-C and TPC-H are trademarks of the Transaction Performance Processing Council (TPPC).

UNIX is a registered trademark of The Open Group in the United States, other countries or both.

Other company, product and service names may be trademarks or service marks of others.