

What's New and Exciting in RPG



(AKA Scott Goes Crazy.)

Presented by

Scott Klement

<http://www.scottklement.com>

© 2016, Scott Klement

*There's a band called 1023 MB.
They haven't had any gigs yet.*

This Session Is Organized by Feature



This session is organized by *feature...*

...not by release.

A chart at the end will detail the releases.



IBM No Longer Waits For Next Release



This is really, really cool!

- Prior to IBM i 7.1 ("V7R1") to get new RPG feature, you waited for next release.
- Couldn't install it right away? Had to wait longer.
- Needed to support both newer and older releases? Wait longer.
- Could be 5+ years before you could use a feature you wanted.

This is no longer true!

- Starting with Open Access (should've been a 7.2 feature) most new stuff is available on all supported releases!
- Improved XML support
- Improved ALIAS support
- Free format support
- No more column restrictions!

3

Is This Good Or Bad?



Definitely good for developers?

- especially software companies.

Does it look bad for IBM?

- why should people update releases?
- do people perceive IBM's RPG commitment as "less"?

For this reason, IBM holds back at least some of the updates to the next release.



4

Support in SEU



There is none.

SEU has not had updates since IBM i 6.1 was released.
IBM i 6.1 is no longer supported.

Use RDi 9.5 (or 9.5.1) to avoid getting errors in your source editor.

5

Free Format (Definitions)



Released in 7.2, PTF back to 7.1

- **CTL-OPT** replaces H-spec
- **DCL-S** replaces D-spec for standalone variables
- **DCL-F** replaces F-spec
- **DCL-DS** replaces D-spec for data structures.
- New **POS** keyword sometimes nicer than **OVERLAY(var:pos)**
- Sequence of F and D specs no longer matters.

```
.....1.....2.....3.....4.....5.....6.....7.....8
ctl-opt dftactgrp(*no) option(*srcstmt:*nodebugio);

dcl-s rrn packed(4: 0);

dcl-f MYFILE workstn sfile(SFL01:rrn)
      indds(inds);

dcl-ds inds qualified;
      Exit      ind pos(3);
      Cancel    ind pos(12);
      ClearSFL  ind pos(50);
      ShowSFL   ind pos(51);
end-ds;
```

6

Free Format (More DS Options)



Released in 7.2, PTF back to 7.1

- Use *N for the DS name for an "unnamed" DS
- **END-DS** can go on the same line if you have no subfields.
- **EXTNAME** or **EXT** for externally defined structures.

```
.....1.....2.....3.....4.....5.....6.....7.....8
dcl-ds *N;
    field1 char(10);
    field2 packed(9: 2);
    field3 zoned(7: 1);
end-ds;

dcl-ds CUSTREC ExtName('CUSTMAS') end-ds;

dcl-f PRINTFILE PRINTER(132);
dcl-ds DATA len(132) end-ds;
.
DATA = 'Print this';
write PRINTFILE DATA;
```

7

Free Format (Procedures)



Released 7.2, PTF back to 7.1

- **DCL-PROC** and **END-PROC** replaces P-spec.
- **DCL-PI** and **END-PI** replace D-spec with PI.
- You can use *N on the DCL-PI for same name as the DCL-PROC.
- Prototypes are only needed when there's no matching PI in the same module

```
.....1.....2.....3.....4.....5.....6.....7.....8
dcl-proc myProcedure;

    dcl-pi myProcedure int(10); ( --or-- dcl-pi *N int(10); )
    parm1 char(10) const;
    parm2 packed(7: 2) const;
end-pi;

dcl-s myVar like(parm2);

return myVar;

end-proc;
```

8

EXPORT(*DCLCASE)



Previously, when procedure name was case-sensitive, you had to repeat it in ExtProc:

```
.....1.....2.....3.....4.....5.....6.....7.....8
D MiXeDcAsEnaMe PR ExtProc('MiXeDcAsEnaMe')
D parm1 10a const

P MiXeDcAsEnaMe B Export
D MiXeDcAsEnaMe PI
D parm1 10a const
/free
... whatever procedure does ...
/end-free
P E
```

***DCLCASE** makes DCL-PROC case-sensitive. Released in 7.2, PTF back to 7.1

```
.....1.....2.....3.....4.....5.....6.....7.....8
dcl-proc MiXeDcAsEnaMe export(*dclcase);
dcl-pi *n;
parm1 char(10) const;
end-pi;
// whatever procedure does.
end-proc;
```

EXTPROC(*DCLCASE)



Can be used to call procedures with case-sensitive names without ExtProc.
Released in 7.2, PTF back to 7.1

```
.....1.....2.....3.....4.....5.....6.....7.....8

dcl-pr MiXeDcAsEnaMe int(10) extproc(*dclcase);
parm1 char(10) const;
end-pr;

MiXeDcAsEnaMe(myVariable);

... can also be used with APIs, such as the Unix-type ones ...

dcl-pr unlink int(10) extproc(*dclcase);
path pointer value options(*string);
end-pr;

unlink('/tmp/deleteMe.txt');
```

Mix Free Format With Fixed



Released in 7.2, PTF back to 7.1

- The /free and /end-free options are no longer required

```
.....1.....2.....3.....4.....5.....6.....7.....8
      dcl-s Y packed(9: 2);
      D X          s          9p 2

      x = 27.50;
      C          eval      Y = 25.50
```

- D/F spec sequence doesn't matter, even in fixed format

```
.....1.....2.....3.....4.....5.....6.....7.....8
      D DATA          ds          132
      FPRINTFILE 0    F 132      PRINTER
      C          eval  DATA = 'Print Me'
      C          write PRINTFILE  DATA
```

11

"Fully Free" Columns Support



Released in 7.3, PTF for 7.1 and 7.2

- Start with **FREE keyword in col 1 of line 1 of your source
- Start with column 1, no limit in line length (aside from record length)
- No length limit on line at all when using IFS files
- cannot use fixed format statements (except via copy book)
- copy books that want **FREE also code it on col 1, line 1 of copy book.
- After line 1 starting with ** is for CTDATA

```
.....1.....2.....3.....4.....5.....6.....7.....8
**FREE
dcl-s states char(12) dim(4) ctdata;
dcl-s x      int(10);

for x = 1 to %elem(states);
  dsply states(x);
endfor;

**
California
Ohio
Mississippi
Wisconsin
```

12

Free Format Is Awesome!



Free format for H, F, D and P specs

- Available in 7.2+, PTF available for 7.1 – no waiting!
- Much easier for new programmers to learn
- Adds new features like *DCLCASE and POS
- Fixed "bad-old" features with F/D spec order and /free /end-free
- Already extremely widely used in the RPG industry!!

The main complaint at the time this was released was

- Still limited to using columns 8 – 80
- But this was fixed later with *FREE support
- So many cool new things, I just can't handle it!

13

CCSIDs Are Really Important



Funny how many programmers, especially in the USA, seem to ignore CCSIDs.

You use CCSIDs every time you use text.

- Letters, symbols, punctuation, etc.
- Everything that isn't raw binary data like images, sound, or binary numbers!!
- That's the vast majority of what we do in business programming!

If you don't think about the CCSID of your data, the computer *assumes* the default.

- Shouldn't you, the programmer know what's going on?
- When the CCSID is different, don't you want to be in control of what's happening?

CCSIDs Are Not Just For Foreign Countries!!

- In the USA, we typically use CCSID 37, which is one flavor of EBCDIC
- There are many others – even the UK's is different!
- France, Germany, Italy, Spain, Mexico, French Canada, China, Korea, etc
- There are different EBCDICs for each, different ASCIIs for each

Unicode is the best!

- Most modern – today's solution, not yesterdays!
- Supports all characters for all languages all in one CCSID!

14

Char/Alpha CCSID Support



Released in 7.2 (Not available in 7.1)

- **CCSID** keyword when defining your fields
- Can be EBCDIC, ASCII or UTF-8 Unicode
- RPG now natively works with all of these character sets!!
- **/SET** and **/RESTORE** can be used to set the defaults for a group of definitions

```
.....1.....2.....3.....4.....5.....6.....7.....8
dcl-s var1 char(100);           // default is job CCSID (37 for me)
dcl-s var2 char(100) ccsid(*utf8); // special value *UTF8 = CCSID 1208

/set ccsid(*char: *utf8)

dcl-s var3 varchar(100);
dcl-s var4 char(200);

/restore ccsid(*char)

var1 = 'MEEEE SCOTT!';           // EBCDIC, default
var2 = var1;                     // converted to UTF-8
var3 = 'HIIIM SCOTT!';           // also converted.
var4 = var3;                     // no conversion needed
var1 = var3;                     // converted to EBCDIC
```

15

More Special Values



Released in 7.2 (Not available in 7.1)

- Already saw **CCSID(*UTF8)** is UTF-8, same as CCSID 1208
- **CCSID(*UTF16)** is UTF-16 same as CCSID 1200 (for double-byte fields)
- External data structures can use **CCSID(*EXACT)** to match the CCSIDs of the file the definition came from.
- **CCSID(*HEX)** when you never want conversion between CCSIDs
- **CCSID(*JOB RUN)** uses the job's default CCSID
- **CCSID(*JOB RUN MIX)** uses the mixed CCSID related to job CCSID

```
Create Table PSNFILE (
  Name      char(30) ccsid 1208,
  Address   char(40) ccsid 37
)
rcdfmt PSNFILEF;
```

```
.....1.....2.....3.....4.....5.....6.....7.....8
dcl-s var1 ucs2(10) ccsid(*utf16);
dcl-s var2 char(10) ccsid(*hex);
dcl-s var3 char(10) ccsid(*jobrun);

// Name will have CCSID 1208, Address will have CCSID 37
dcl-ds PSNFILE ext ccsid(*exact) end-ds;
```

16

CTL-OPT CCSID Keywords



Released in 7.2 (Not available in 7.1)

- **CTL-OPT** (H-Spec) **CCSID** keyword sets default for whole module
- You can also set **CCSID(*UCS2:1200)**, but we've had that for a long time.

```
.....+.....1.....+.....2.....+.....3.....+.....4.....+.....5.....+.....6
  H ccsid(*char: *utf8)
  -or-
  ctl-opt ccsid(*char: *utf8);
```

Released in 7.2, PTF for 6.1, 7.1

CTL-OPT (H-Spec) **CCSIDCVT** keyword has two functions

- ***EXCP** causes RPG to send an exception (error) when a character would be lost by CCSID conversion
- ***LIST** puts a listing of all of the automatic CCSID conversions that are taking place into your compile listing, so it's easy to see where stuff is being translated automatically.

```
.....+.....1.....+.....2.....+.....3.....+.....4.....+..
  H ccsidcvt(*list)
  -or-
  ctl-opt ccsidcvt(*excp);
```

17

CCSIDCVT(*EXCP)



What should happen if you convert a character that cannot exist in the result CCSID?

Default: A substitution character is inserted that shows that some character is missing.

With ***EXCP**: An exception/error is sent to your RPG program

```
.....+.....1.....+.....2.....+.....3.....+.....4.....+.....5.....+.....6.....+.....7.....+
01  **FREE
02  ctl-opt ccsidcvt(*excp: *list);
03  ctl-opt ccsid(*CHAR: *UTF8) ccsid(*ucs2: *utf16);
04
05  dcl-s var1 ucs2(2);
06  dcl-s var2 char(10);
07  dcl-s var3 char(10) ccsid(*jobrun);
08
09  var1 = u'4f605978';      // Chinese characters for "Ni Hao" in UTF-16
10  var2 = var1;            // converted to UTF-8
11  var3 = var2;            // converted to US-EBCDIC
```

```
RN0452 Some characters could not be converted from
      CCSID(1208) to CCSID(37)
```

18

CCSIDCVT(*LIST)



```
.....1.....2.....3.....4.....5.....6.....7...
01 **FREE
02 ctl-opt ccscidvt(*excp: *list);
03 ctl-opt ccscid(*CHAR: *UTF8) ccscid(*ucs2: *utf16);
04
05 dcl-s var1 ucs2(2);
06 dcl-s var2 char(10);
07 dcl-s var3 char(10) ccscid(*jobrun);
08
09 var1 = u'4f605978'; // Chinese characters for "Ni Hao" in UTF-16
10 var2 = var1; // converted to UTF-8
11 var3 = var2; // converted to US-EBCDIC
```

CCSIDCVT(*LIST) adds this to compile listing:

```
          C C S I D   C o n v e r s i o n s
From CCSID      To CCSID      References
1200            1208          10
1208            *JOBRUN      11
* * * * *   E N D   O F   C C S I D   C O N V E R S I O N S   * * * * *
```

19

Disabling Database CCSID Conversion



Historically, RPG didn't work with multiple CCSIDs in the same program.

- but the database did!
- therefore, RPG converted all database CCSIDs to/from the job's CCSID
- for backward compatibility, it still does!

Now that it does work with lot of CCSIDs (as we just discussed)

- It's often better to keep the original database CCSIDs in RPG
- We can use RPG's features to convert them if needed.

```
.....1.....2.....3.....4.....5.....6.....7...
**FREE
ctl-opt openopt(*nocvtdata); // Default for whole program

dcl-f MYFILE workstn sfile(SFL01:rrn)
02 ctl-opt ccscidvt(*excp: *list);
03 ctl-opt ccscid(*CHAR: *UTF8) ccscid(*ucs2: *utf16);
04
05 dcl-s var1 ucs2(2);
06 dcl-s var2 char(10);
07 dcl-s var3 char(10) ccscid(*jobrun);
08
09 var1 = u'4f605978'; // Chinese characters for "Ni Hao" in UTF-16
10 var2 = var1; // converted to UTF-8
11 var3 = var2; // converted to US-EBCDIC
```

20

Expanded Timestamps



Released 7.2

You can now specify how many fractional seconds you want. You can have up to 12 digits of fractional seconds in a timestamp field.

```
.....1.....2.....3.....4.....5.....6.....7.....8
dcl-s ts    timestamp    inz(*sys); // YYYY-MM-DD-hh.mm.ss.ffffff (default)
dcl-s ts0   timestamp(0)  inz(*sys); // YYYY-MM-DD-hh.mm.ss
dcl-s ts1   timestamp(1)  inz(*sys); // YYYY-MM-DD-hh.mm.ss.f
dcl-s ts3   timestamp(3)  inz(*sys); // YYYY-MM-DD-hh.mm.ss.fff
dcl-s ts12  timestamp(12) inz(*sys); // YYYY-MM-DD-hh.mm.ss.ffffffffffff
```

The INZ() keyword, TIME opcode and %timestamp() BIF still only set the first 3 fractional digits. The remainder of the timestamp is set to zeroes unless you initialize it yourself with an API or similar.

%SUBDT Digits/Decimals



Released in 7.2

The %SUBDT BIF can specify

- a number of digits returned in 3rd parameter
- a number of decimal places when working with *SECONDS

```
.....1.....2.....3.....4.....5.....6.....
**FREE
dcl-s ts    timestamp inz(z'2016-09-16-18.09.33.123456');
dcl-s year4 packed(4: 0);
dcl-s year2 packed(2: 0);
dcl-s secs  packed(4: 2);

year4 = %subdt(ts:*years:4); // year4 = 2016
year2 = %subdt(ts:*years:2); // year2 = 16
secs  = %subdt(ts:*seconds:4:2); // secs = 33.12
```

ALIAS Support on Files/Tables



Consider this physical file:

- short field names were hard to understand
- ALIAS allows longer names, but prior to 7.1, we couldn't use them in RPG!

```
.....1.....2.....3.....4.....5.....6.....+..
.
A          R CUSTREC
A          CUSTNO          4S 0          ALIAS(CUST_NUM)
A          CUBLAD          30A          ALIAS(CUST_BILLING_ADDRESS)
A          CUBLCT          20A          ALIAS(CUST_BILLING_CITY)
A          CUBLST          2A          ALIAS(CUST_BILLING_STATE)
A          CUBLZP          10A          ALIAS(CUST_BILLING_ZIP)
A          CUSHNM          30A          ALIAS(CUST_SHIPPING_ADDRESS)
A          CUSHCT          20A          ALIAS(CUST_SHIPPING_CITY)
A          CUSHST          2A          ALIAS(CUST_SHIPPING_STATE)
A          CUSHZP          10A          ALIAS(CUST_SHIPPING_ZIP)
A          K CUSTNO
```

23

ALIAS Support in SQL



Long names are enabled by default in SQL

- You can use "for" to also give a short name (optional)
- If you don't use "for", SQL will generate a short name like CUS00001

```
.....1.....2.....3.....4.....5.....6.....+..
Create Table CUST (
  CUST_NUM          for CUSTNO numeric(4, 0) not null,
  CUST_BILLING_ADDRESS for CUBLAD char(30) not null,
  CUST_BILLING_CITY   for CUBLCT char(20) not null,
  CUST_BILLING_STATE  for CUBLST char(2) not null,
  CUST_BILLING_ZIP    for CUBLZP char(10) not null,
  CUST_SHIPPING_ADDRESS for CUSHAD char(30) not null,
  CUST_SHIPPING_CITY   for CUSHCT char(20) not null,
  CUST_SHIPPING_STATE  for CUSHST char(2) not null,
  CUST_SHIPPING_ZIP    for CUSHZP char(10) not null,
  primary key (CUST_NUM)
)
rcdfmt CUSTREC;
```

24

Original 7.1 Alias Support



ALIAS keyword would generate long names

- but only worked with data structure I/O
- required "qualified" keyword on file to avoid I-specs and O-specs

```
.....1.....2.....3.....4.....5.....6.....+...
FCUST      UF  E          K DISK  QUALIFIED ALIAS

D CUST_IN          E DS          extname(CUST:*input)
D                  qualified alias
D CUST_OUT        E DS          extname(CUST:*output)
D                  qualified alias

D Key              s           like(CUST_IN.CUST_NUM)

/free
key = 1000;
chain key CUST CUST_IN;
if %found;
  eval-corr CUST_OUT = CUST_IN;
  if cust_out.cust_billing_address = *blanks;
    cust_out.cust_billing_address = cust_out.cust_shipping_address;
    cust_out.cust_billing_city    = cust_out.cust_shipping_city;
    cust_out.cust_billing_state   = cust_out.cust_shipping_state;
    cust_out.cust_billing_zip     = cust_out.cust_shipping_zip;
  update CUST.CUSTREC CUST_OUT;
endif;
endif;
```

25

Improved Alias Support



Released 7.3, PTF for 7.1, 7.2

- ALIAS now works without data structures
- Compiler-generated I-specs/O-specs now support the longer names

```
.....1.....2.....3.....4.....5.....6.....+...
FCUST      UF  E          K DISK  ALIAS

D Key              s           like(CUST_NUM)

/free
key = 1000;
chain key CUST;
if %found;
  if cust_billing_address = *blanks;
    cust_billing_address = cust_shipping_address;
    cust_billing_city    = cust_shipping_city;
    cust_billing_state   = cust_shipping_state;
    cust_billing_zip     = cust_shipping_zip;
  update CUSTREC;
endif;
endif;
```

26

Relaxed DS I/O Rules



Prior to this update, if you use data structures for I/O, you must have separate ones for *INPUT and *OUTPUT.

Released 7.3, PTF for 7.2, 7.1

- Can use **EXTNAME(MYFILE:*ALL)** for any type of I/O
- Or can use **LIKEREC(MYFILE)** (with no usage) for any type of I/O

```
**FREE
dcl-f CUST disk keyed usage(*input: *output: *update);
dcl-ds rec extname('CUST':*ALL) qualified alias end-ds;

chain 1000 CUST rec;

if %found;
  rec.cust_billing_address = '123 Main Street';
  update CUSTREC rec;
else;
  rec.cust_num = 1000;
  rec.cust_shipping_address = '123 Main Street';
  write CUSTREC rec;
endif;
```

Same DS on
chain, update and
write operations

27

Improved Null Support



For some time, RPG has had support for database nulls in its native I/O (i.e. "F-spec files") using **ALWNULL(*USRCTL)** and the **%NULLIND BIF**.

In version **7.3** (no PTFs for older versions), RPG has extended this support with the **NULLIND** keyword. This keyword enables you to:

- Define your own (standalone) fields as null-capable.
- Define your own indicators to replace the **%NULLIND BIF**
- Associate a null map data structure with a record data structure to handle nulls in data structure I/O

Like the **%NULLIND BIF**, the **NULLIND** keyword requires **ALWNULL(*USRCTL)** to be specified on the CTL-OPT or H-spec.

Standalone Null-Capable Field



I've wanted this for a long time!!

```
H alwnull(*usrctl)

D ShipDate          S          D nullind
```

Also free format:

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

dcl-s ShipDate date NULLIND;
```

Use %NULLIND to check/set the null indicator (same as a database field)

```
if %nullind(ShipDate) = *OFF;
  msg = 'Order was shipped on ' + %char(ShipDate:*USA);
else;
  msg = 'Order has not been shipped';
endif;

%nullind(ShipDate) = *ON;
```

29

Works with OPTION(*NULLIND)



Null capable fields can be passed between subprocedures as well

```
dcl-s ShipDate date NULLIND;
.
.
TestNull(ShipDate);
.
.
dcl-proc TestNull;
  dcl-pi *N;
    TheDate Date options(*nullind);
  end-pi;
  if %nullind(TheDate);
    dsply 'is null';
  else;
    dsply 'is not null';
  endif;
end-proc;
```

30

Use Your Own Indicator for Nulls



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

dcl-s NullShipDate ind;
dcl-s ShipDate date NULLIND(NullShipDate);

// This is the same as %nullind(ShipDate) = *ON
NullShipDate = *ON;

// This is the same as IF %NULLIND(ShipDate)=*ON
if NullShipDate = *ON;
    // not shipped
endif;
```

Also works in fixed format (in case you were wondering)

```
      H alwnull(*usrctl)

      D NullShipDate      S              N
      D ShipDate         S              D nullind(NullShipDate)
```

31

Using NULLIND for Files



Released in 7.3, LIKERECE/EXTNAME can be used with *NULL to build a data structure that has the null indicators for a database record.

For example, consider this file

```
Create Table NULLTEST (
  CustNo numeric(4, 0) not null,
  FirstOrd date,
  AmtOwed decimal(9, 2),
  SalesRep char(30)
)
rcdfmt NULLTESTF;
```

FirstOrd, AmtOwed and SalesRep are null capable (but CustNo is not)

RPG code example on next slide

32

LIKEREC/EXTNAME *NULL



Released in 7.3

- Requires ALWNULL(*USRCTL)
- Still need to say *INPUT/*OUTPUT/*ALL so it knows which fields to include
- Add *NULL to make it the null map for the record
- Add NULLIND to the "normal" record DS to associate the null map to it.

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

dcl-f NULLTEST disk usage(*input:*update);

dcl-ds NULL likerec(NULLTESTF:*ALL:*NULL);
dcl-ds REC likerec(NULLTESTF:*ALL) nullind(NULL);

read NULLTEST rec;

if null.FirstOrd = *ON;
  null.FirstOrd = *off;
  Rec.FirstOrd = %date();
  update NULLTESTF rec;
endif;
```

33

Use NULLIND in Prototypes



You can use NULLIND on prototypes/procedure interfaces...

- must specify an indicator parameter, NULLIND(field) not just NULLIND
- Indicator parameter must be passed in the same parameter list
- works with both standalone fields and data structures

```
CheckFields(rec : null);
.
.
dcl-proc CheckFields;
  dcl-pi *n;
    r likeds(rec) nullind(n);
    n likeds(null);
  end-pi;

  if n.AmtOwed = *off;
    dsply (%char(r.CustNo) + ' owes ' + %char(r.AmtOwed));
  endif;

  if n.SalesRep = *ON;
    dsply (%char(r.CustNo) + ' has no sales rep');
  endif;
end-proc;
```

34

Reverse Scanning



Released in 7.3

- %SCANR is like scan, but scans right-to-left
- it will find the last occurrence of a string rather than the first

```
Pathname = '/home/sklement/test/sales2015.csv';  
  
pos = %scanr('/', Pathname);  
  
// pos = 20  
  
Stmf = %subst(Pathname:pos+1);  
  
// Stmf = sales2015.csv  
  
DirName = %subst(Pathname:1:pos-1);  
  
// dirname = /home/sklement/test  
  
*inlr = *on;
```

35

Limit Scanning



Released in 7.3

- %SCAN now has an optional length parameter
- previously had start position, but not length
- works with %SCANR as well.

```
// from last slide:  
// Pathname = /home/sklement/test/sales2015.csv  
// pos = 20  
  
if %scan('test': PathName: 1: pos) > 0;  
    // "test" was found in the directory portion  
    dsply 'test found';  
endif;  
  
if %scan('sales': PathName: 1: pos) = 0;  
    // "sales" was not found in the directory portion  
    dsply 'sales not found';  
endif;  
  
x = %scanr('/', PathName: 1: pos-1);  
// x = 15
```

36

Sooooo Many Enhancements



There are so many enhancements, I don't even have time to cover them all in-depth! Here are some that I didn't cover in this talk:

Not covered, because these are already well known:

- XML-INTO was enhanced with **countprefix**, **datasubf** (7.1, ptf for 6.1)
- XML-INTO support for **case=convert and namespaces** (7.2 ptf for 7.1, 6.1)
- Open Access and the **HANDLER** keyword (7.2 ptf for 7.1, 6.1)

Not covered because "too trivial"

- **PGMINFO** can be used to control which procedures are included in PCML
- **VALIDATE(*NODATETIME)** slightly improves date/time performance by eliminating error checking (don't use this, please)
- **DCLOPT(*NOCHGDSLEN)** allows %SIZE to figure out the size of a data structure in some definitions by guaranteeing that input, output and calcs won't change the length.

During 7.1 Development Cycle



	<u>6.1</u>	<u>7.1</u>	<u>7.2</u>	<u>7.3</u>
Sort and Search Qualified DS Arrays		X	X	X
Sort in Ascending or Descending		X	X	X
%SCANRPL BIF		X	X	X
%LEN(*MAX)		X	X	X
ALIAS Support		X	X	X
RTNPARM		X	X	X
%PARMNUM		X	X	X
Prototypes Are Optional		X	X	X
Procedures Convert CCSIDs when CONST		X	X	X
Teraspace Storage model		X	X	X
Default ACTGRP is now *STGMDL		X	X	X
H-spec option to change ALLOC to teraspace		X	X	X
Encrypted Debugging View		X	X	X
XML-INTO datasubf and countprefix	ptf	X	X	X

During 7.2 Development Cycle



	<u>6.1</u>	<u>7.1</u>	<u>7.2</u>	<u>7.3</u>
XML-INTO namespaces options	ptf	ptf	X	X
XML-INTO case=convert	ptf	ptf	X	X
CCSIDCVT keyword to list or give exceptions during auto-converts	ptf	ptf	X	X
Date/Time efficiency VALIDATE(*NODATETIME)	ptf	ptf	X	X
Open Access	ptf	ptf	X	X
CCSID support on A data type (and UTF-8)			X	X
CCSID(*EXACT)			X	X
CCSID(*HEX) and hex literals			X	X
Conversion During Concatenation			X	X
OPENOPT(*NOCVTDATA) and DATA(*NOCVT)			X	X
/SET and /RESTORE			X	X
Control %SUBDT Length			X	X
Timestamps up to 12 fractional digits (why??)			X	X
Free-Format H, F, D and P specs		ptf	X	X
No more need for /free and /end-free		ptf	X	X
File/Definitions Can be In Any Sequence (Fixed/Free)		ptf	X	X
EXPORT(*DCLCASE)			X	X

39

During 7.3 Development Cycle



	<u>6.1</u>	<u>7.1</u>	<u>7.2</u>	<u>7.3</u>
**FREE ("fully free form")		ptf	ptf	X
%SCANR BIF				X
Length Parameter to %SCAN				X
Improved ALIAS support		ptf	ptf	X
Relaxed DS I/O rules		ptf	ptf	X
Improvements to DB Null Support -- NULLIND				X
Improvements to PCML generation		ptf	ptf	X
DCLOPT(*NOCHGDSLEN)		ptf	ptf	X

40

Get The Latest PTFs



You can any of the features (if available for your release of IBM i) by installing these PTFs.

There is no need to install separate PTFs for each feature, these PTFs (and their pre-requisites) include it all.

- 7.1
- *CURRENT compiler: **SI61169**
 - TGTRLS(V6R1M0) compiler: **SI53442**
 - runtime: **SI58913**
- 7.2:
- *CURRENT compiler: **SI62208**
 - TGTRLS(V7R1M0) compiler: **SI60690**
 - TGTRLS(V6R1M0) compiler: **SI52577**
 - runtime: **SI58915**
- 7.3:
- *CURRENT compiler: **SI61030**
 - TGTRLS(V7R2M0) compiler: **SI61083**
 - TGTRLS(V7R1M0) compiler: **SI61125**
 - runtime: **SI59831**

41

This Presentation



You can download a PDF copy of this presentation:

<http://www.scottklement.com/presentations/>

Thank you!

42