Free-form RPG

Barbara Morris
IBM Software Group, Rational

Another big step forward for RPG – a totally free-form program

```
ctl-opt bnddir('ACCRCV');

dcl-f custfile usage(*update);
dcl-ds custDs likerec(custRec);
dcl-f report printer;

read custfile custDs;
dow not %eof;
   if dueDate > %date(); // overdue?
      sendOverdueNotice();
      write reportFmt;
      exec sql insert :name, :duedate into
               mylib/myfile;
   endif;
   read custfile custDs;
enddo;
*inlr = '1';

dcl-proc sendOverdueNotice;
   sendInvoice (custDs : %date());
end-proc;
```

RPG programmers will find this new syntax easy to learn

Non-RPG programmers will find this new syntax much easier to learn than fixed form

How far RPG has come

Let's take a look at the last 25 years of RPG syntax

RPG III (OPM RPG) System-38 – V2R3

```
FCUSTFILEIF  E                    DISK
FREPORT  O   E                    PRINTER
ICUSTDS    E DSCUSTFILE
 /COPY GETCURDAT
 /COPY INVOICE
C                     READ CUSTFILE
C           *INLR     DOWNE*ON
C           DUEDAT    IFGT CURDAT
C                     EXSR SNOVDU
C                     WRITEREPORTFM
C/EXEC SQL INSERT :NAME, :DUEDATE INTO
C+              MYLIB/MYFILE
C/END-EXEC
C                     ENDIF
C                     READ CUSTFILE                 LR
C                     ENDDO
C*
C           SNOVDU    BEGSR
C                     CALL 'SNDINVCE'
C                     PARM           CUSTDS
C                     PARM ISOVDU    OVERDU 10
C                     ENDSR
```

Limit of 6 character names. "Send overdue notice" = SNOVDU

All code is upper case

V3R1

```
H bnddir('ACCRCV') dftactgrp(*no)
Fcustfile  uf   e             disk
Freport    o    e             printer
D custDs         e ds                    extname(custfile)
D today            s                 d   datfmt(*iso)
 /copy invoices
C                   time                        today
C                   read      custfile
C                   dow       not %eof
C                   if        dueDate > today
C                   exsr      sendOvrNtc
C                   read      custfile
C                   write     reportFmt
C/exec sql insert :name, :duedate into
C+              mylib/myfile
C/end-exec
C                   endif
C                   enddo
C                   eval      *inlr = '1'

C     sndOvrNtc     begsr
C                   call      'SNDINVCE'
C                   parm                    custDs
C                   parm      IS_OVERDUE    overdue         10
C                   endsr
```

Mixed case

Up to 10 characters for names. "Send overdue notice" = SendOvrNtc

Date/time support

V3R2 – V4R4

```
H bnddir('ACCRCV') dftactgrp(*no)
Fcustfile  uf   e             disk
Freport    o    e             printer
D custDs        e ds                    extname(custfile)
D today           s                d    datfmt(*iso)
D sendOverdueNotice...
D                 pr
C                   time                    today
C                   read      custfile
C                   dow       not %eof
C                   if        dueDate > today
C                   callp     sendOverdueNotice (custDs)
C                   write     reportFmt
C/exec sql insert :name, :duedate into
C+              mylib/myfile
C/end-exec
C                   endif
C                   read      custfile
C                   enddo
C                   eval      *inlr = '1'

P sendOverdueNotice...
P                 b
 ...
```

Subprocedures

Long names. "Send overdue notice" = SendOverdueNotice

V5R1 – V5R2

```
H bnddir('ACCRCV') dftactgrp(*no)
Fcustfile  uf   e             disk
Freport    o    e             printer
D custDs          ds                  extname(custfile)

D sendOverdueNotice...
D                 pr
 /free
    read custfile custDs;
    dow not %eof;
       if dueDate > %date(); // overdue?
          sendOverdueNotice ();
          write reportFmt;
 /end-free
C/exec sql insert :name, :duedate into
C+             mylib/myfile
C/end-exec
 /free
       endif;
       read custfile custDs;
    enddo;
    *inlr = '1';
 /end-free
...
```

Free form calculations

Indentation!

Many new built-in functions

V5R3 – 7.1

```
H bnddir('ACCRCV') dftactgrp(*no)
Fcustfile  uf   e             disk
Freport    o    e             printer
D custDs        e ds                    extname(custfile)
D sendOverdueNotice...
D                 pr
/free
    read custfile custDs;
    dow not %eof;
       if dueDate > %date(); // overdue?
          sendOverdueNotice ();
          write reportFmt;
          exec sql insert :name, :duedate into
                   mylib/myfile;
       endif;
       read custfile custDs;
    enddo;
    *inlr = '1';
 /end-free

P sendOverdueNotice...
P                 b
 /copy invoices
...
```

Free-form SQL

7.1 TR7, and a future release of RDI

```
ctl-opt bnddir('ACCRCV');

dcl-f custfile usage(*update);
dcl-ds custDs likerec(custRec);
dcl-f report printer;

read custfile custDs;
dow not %eof;
   if dueDate > %date(); // overdue?
      sendOverdueNotice ();
      write reportFmt;
      exec sql insert :name, :duedate into
             mylib/myfile;
   endif;
   read custfile custDs;
enddo;
inlr = '1';

dcl-proc sendOverdueNotice;
   /copy invoices
   sendInvoice (custDs : IS_OVERDUE);
 end-proc;
```

No /FREE, /END-FREE

All free-form statements

Better colorization options in the editor

What is wrong with fixed-form code?

- Most programmers today have never seen fixed form code
- When they see RPG code like this, it looks like gibberish

```
H bnddir('ACCRCV') dftactgrp(*no)
Fcustfile  if   e             disk
Freport    o    e             printer
```

- Here's what happens when a non-RPG programmer tries to make a change

```
H bnddir('ACCRCV')
Fcustfile  if   e             disk
Freport   o    e             printer
RNF0289E Entry contains data that is not valid; only valid data is used.
RNF2013E The Device entry is not PRINTER, DISK, SEQ, WORKSTN or SPECIAL;
     defaults to DISK.
RNF2003E The File Type is not I, O, U, or C; defaults to O if File
     Designation is blank, otherwise to I.
RNF2005E The Sequence entry is not blank, A, or D; defaults to blank.
... more error messages
```

RPG is still not 100% free

There are still some areas where RPG is not yet free

- Free-form code is still restricted to columns 8 – 80
- I specs and O specs must still be coded in fixed-form
  - I and O specs are considered deprecated by many RPG programmers in favor of externally-described files
- Code related to the RPG cycle must be coded in fixed-form
  - The cycle is considered deprecated by many RPG programmers in favor of using SQL for scenarios where the cycle formerly shone

What will an all-free RPG mean for non-RPG programmers?

- Learn RPG much more easily than with fixed-form
  - Fewer "secret codes" to remember ("E in column 19 means externally-described")
  - Better token-colorization in the RDI editor, allowing programmers to have the same look-and-feel for RPG code as for other languages like Java or PHP
  - New programmers will only have to learn how to use RPG, without having to struggle with how it is coded
- Learn from RPG programmers about
  - Business programming
  - Attributes of IBM i that make it a dream for programmers, such as the automatic logging of messages (joblog)

What will an all-free RPG mean for RPG programmers?

- Indented code is more maintainable
- Removal of many frustrations:
  - /FREE and /END-FREE in every procedure
  - Two lines for many definitions in fixed-form

```
D getNextCustomer...
D                 pr
```

vs

```
dcl-pr getNextCustomer;
```

  - Insufficient room in D-spec keywords for long strings

```
D HSSFCellStyle   c
D                                 'org.apache.poi.hssf.-
D                                 usermodel.HSSFCellStyle'
```

vs

```
dcl-c HSSFCellStyle 'org.apache.poi.hssf.usermodel.HSSFCellStyle';
```

What will an all-free RPG mean for businesses?

Having some non-RPG programmers could be valuable for an RPG shop

- Bring knowledge of other programming languages and other platforms
- Modular coding is the norm
- Already familiar with modern editors and other modern tools that lead to greater productivity
- Accustomed to picking up new languages and finding out how to stretch each language to get it to do what they want
- Give a wake-up call to any existing RPG programmers who are not keeping up with everything that is available to RPG programmers

What doesn't change with each new improvement to RPG syntax

All versions of RPG have had

- Upward compatibility with earlier versions of the syntax
  - The RPG compiler will still support fixed-form code
- Unparalleled ease of coding record-level I/O
- Excellent support for decimal arithmetic which is so important for business programming

What doesn't change with each new improvement to RPG syntax

All versions of RPG have had

- Excellent integration with the system
  - File I/O is tightly integrated with the database
  - Support for data areas
  - Embedded SQL allows easy mixing of record-level I/O and SQL I/O
  - ILE RPG:
    - Support for calling between all OPM and all ILE languages
    - Support for calling between RPG and Java
    - Support for reading XML documents
    - Support, through Open Access, for devices and resources not directly supported by RPG

Additional support

Embedded SQL

The SQL precompiler will have support for the new free-form syntax

RDI

An upcoming release of RDI will have support for free-form RPG

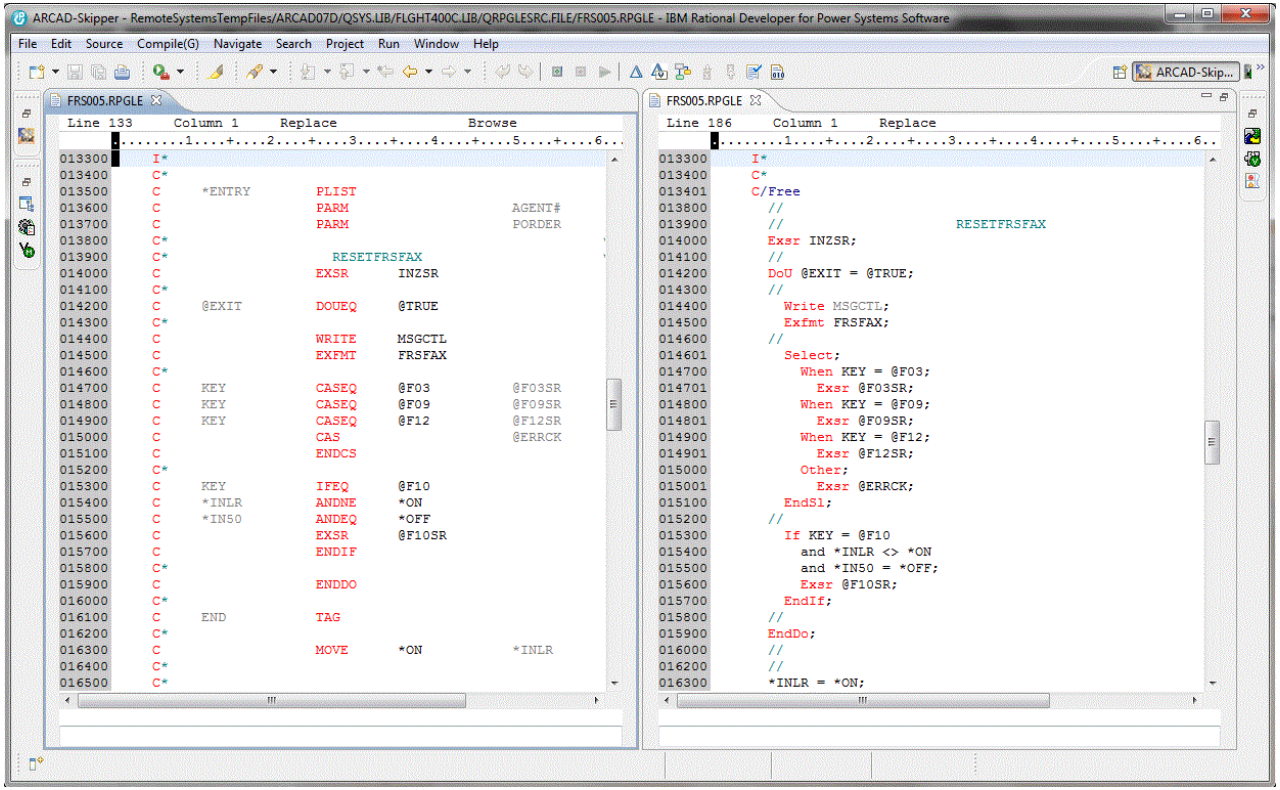
Conversion

- RDI will not do any conversion from H F D P to free-form
- ARCAD will be announcing a free-form conversion at the same time as TR7

ARCAD-Converter : Application Modernization

Convert to Free Format RPG

Help the latest generation of developers to manage and maintain your applications by converting to Free Format RPG



Rational Developer for i plug-in
Support for the latest compiler specs (C, H, F, D and P)

On-demand or bulk conversion

Let's look at the details

- General features
- Control (H)
- File declaration (F)
- Data declaration (D)
- Procedure (P)

Some general features

The new statements all

- Start with an "opcode"
- End with a semicolon

Just like calculation statements in RPG:

```
if duedate > today;
   sendAngryLetter (customer);
endif;
```

Some general features

Unlike free-form calculations, can have /IF, /ELSEIF, /ELSE, /ENDIF within a statement

```
dcl-s salary
   /if defined(large_vals)
      packed(13 : 3)
   /else
      packed(7 : 3)
   /endif
      ;
```

Some general features

Can mix fixed-form and free-form
- /FREE and /END-FREE are not required

Example: Defining the TAG for SQL "whenever"

```
     exec sql whenever sqlerror goto err;
     ...
     return;
C         err           tag
     ok = *off;
     reportSqlError ();
```

Control statements

CTL-OPT (Control Option) statement

- Start with CTL-OPT
- Zero or more keywords
- End with semicolon

```
ctl-opt option(*srcstmt : *nodebugio)
        dftactgrp(*no);
```

Control statements

- Can have multiple CTL-OPT statements
- The rules about not repeating keywords apply across all statements

```
   ctl-opt; // no keywords
   ctl-opt option(*srcstmt : *nodebugio)
           dftactgrp(*no); // two keywords
 H datfmt(*iso) text('My Program')
    ctl-opt alwnull(*usrctl); // free again
```

Control statements

One little enhancement for free-form H:

If there is at least one free-form control statement, you don't need DFTACTGRP(*NO) if you have one of the ACTGRP, BNDDIR, or STGMDL keywords

File statements

DCL-F (Declare file) statement

- Start with DCL-F
- File name
- Keywords
- End with semicolon

File statements

- Only full-procedural and output – no cycle, RAF or table files
- The name can be longer than 10 as long as there's an EXTFILE keyword (and an EXTDESC keyword if externally-described)

```
dcl-f year_end_report printer
          oflind(overflow)
          extdesc('YERPT')
          extfile(*extdesc);
```

File statements – the device

Device keyword or LIKEFILE must be the first keyword

DISK, PRINTER, SEQ, SPECIAL, WORKSTN

- Defaults to DISK

Externally-described: *EXT (default)

Program-described: record-length

```
dcl-f orders; // defaults to DISK(*EXT)
dcl-f qprint printer(132);
dcl-f screen workstn; // defaults to *EXT
```

File statements – the usage

USAGE keyword

*INPUT, *OUTPUT, *UPDATE, *DELETE

Equivalent of fixed-form File Type (I, O, U, C) and File-Addition

Default for USAGE depends on the device

```
dcl-f orders disk;      // *INPUT
dcl-f report printer;   // *OUTPUT
dcl-f screens workstn; // *INPUT : *OUTPUT
```

- SEQ and SPECIAL default to USAGE(*INPUT)

File statements – the usage

Some usage values imply other values

- *UPDATE implies *INPUT
- *DELETE implies *UPDATE and *INPUT

```
// USAGE(*INPUT : *UPDATE)
dcl-f orders disk usage(*update);

// USAGE(*INPUT : *UPDATE : *DELETE)
dcl-f arrears disk usage(*delete);
```

Can specify implied values explicitly too

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

File statements – difference for *DELETE

In fixed form, U enables update and delete

In free form, *UPDATE does not enable delete

- *DELETE must be coded explicitly

File statements – Keyed files

For externally-described files, KEYED keyword

```
dcl-f orders disk keyed;
```

For program-described files, KEYED(*CHAR:len)

```
dcl-f generic disk(2000) keyed(*CHAR:100);
```

File statements – Program-described keyed files

Only character keys supported for program-described

For other types, use a data structure

```
dcl-f generic disk(2000) keyed(*CHAR:7);

dcl-ds key len(7) qualified;
   item_num packed(12);
end-ds;

key.item_num = 14;
chain key generic;
```

File statements

F specs can be mixed with D specs (even in fixed form)

Group related items together

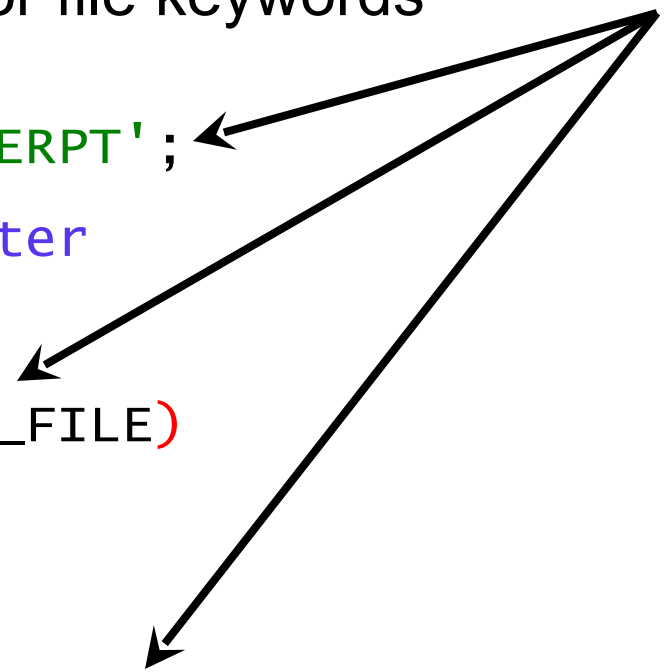
```
dcl-f orders
      usage (*update : *output) keyed;
dcl-ds orders_dsi
      likerec (ordersR:*input);
dcl-ds orders_dso
      likerec (ordersR:*output);
dcl-s num_orrders int(10);
```

```
dcl-f report printer;
dcl-ds report_ds
      likerec (reportR:*output);
```

File statements

Named constants can be used for file keywords

```
dcl-c YEAR_END_RPT_FILE 'YERPT';
dcl-f year_end_report printer
      oflind(overflow)
      extdesc(YEAR_END_RPT_FILE)
      extfile(*extdesc);
dcl-ds report_ds
      extname(YEAR_END_RPT_FILE:*output);
```



Data definition statements

- Start with DCL-x
- Item name – can be *N if not named
- Keywords
- End with semicolon

```
dcl-s name like(other_name);
```

Standalone fields

The first keyword must be a data-type keyword or the LIKE keyword

```
dcl-s salary packed(9:2) inz(0);
dcl-s annual_salary like(salary : +2);
```

Data-type keywords

Some data-type keywords match the Data-Type entry exactly

```
CHAR, INT, POINTER ...
```

Some merge the Data-Type entry with another keyword

```
VARCHAR = A + VARYING

DATE = D + DATFMT

OBJECT = O + CLASS
```

Data-type keywords

String data types

Fixed length	`CHAR(characters)`
	`GRAPH(characters)`
	`UCS2(characters)`

Varying length	`VARCHAR(characters)`
	`VARGRAPH(characters)`
	`VARUCS2(characters)`

Varying length with specific prefix-size	`VARCHAR(characters : 4)`
	`VARGRAPH(characters : 4)`
	`VARUCS2(characters : 4)`

Indicator	`IND`

Data-type keywords

Numeric data types

("BINDEC" is explained on the next slide)

Decimal types with default zero decimal positions	`PACKED(digits)`
	`ZONED(digits)`
	`BINDEC(digits)`

Decimal types with specific decimal positions	`PACKED(digits : decimals)`
	`ZONED(digits : decimals)`
	`BINDEC(digits : decimals)`

Integer, Unsigned	`INT(digits)  3,5,10,20` `UNS(digits)  3,5,10,20`
Float	`FLOAT(bytes) 4,8`

BINDEC keyword – reduce confusion over RPG's "binary" type

RPG's "binary" type is a decimal type stored in binary form, not a "true binary".

```
D binfld          S                   9B 3
```

- Values between -999999.999 and 999999.999

RPG programmers see "binary" in API documention and think they should code B in their RPG programs

Non-RPG programmers see "binary" as the RPG data type, and think it means true binary

- When they want an 4 byte binary, they code 4B which is a 2-byte binary with 4 digits

Data-type keywords

Other data types

Date, Time, Timestamp	`DATE`
	`TIME`
	`TIMESTAMP`

Date, Time with format	`DATE(*YMD-)` `TIME(*HMS-)`
Pointer	`POINTER`
Procedure pointer	`POINTER(*PROC)`
Object	`OBJECT(*JAVA:class)` (Parameters are optional for the return type of a constructor method)

Tip for remembering the data-type keywords

If there is a related built-in function, the data-type keyword has the same name:

%CHAR	- CHAR and VARCHAR
%GRAPH	- GRAPH and VARGRAPH
%UCS2	- UCS2 and VARUCS2
%DATE	- DATE
%TIME	- TIME
%TIMESTAMP	- TIMESTAMP
%INT	- INT
%UNS	- UNS
%FLOAT	- FLOAT

Exception: %DEC. The decimal data types are PACKED, ZONED, BINDEC.

Data structures

Data-structures end the subfield list with END-DS

- not used for LIKEDS or LIKEREC data structures

END-DS is optionally followed by the DS name

```
dcl-ds info;
   name varchar(25);
   price packed(4 : 2);
end-ds info;
```

If no subfields, code END-DS on the DCL-DS line

```
dcl-ds prt_ds len(132) end-ds;
```

Prototypes and procedure interfaces

Prototypes and procedure interfaces are similar

```
dcl-pr qcmdexc extpgm;
   cmd char(3000);
   cmd_len packed(15 : 5);
end-pr;

dcl-pr init end-pr; // no parameters

dcl-pr init;
end-pr;       // can be a separate statement

dcl-pi *n varchar(25); // no need for a
    name
   id int(10);
end-pi;
```

Bonus feature:
EXTPGM parameter
is optional

*DCLCASE for external procedure names

A common bug:

- EXTPROC is needed for the mixed-case name
- The programmer uses copy-paste and forgets one change

```
D Qc3EncryptData...
D                 pr     extproc('Qc3EncryptData')
D Qc3DecryptData...
D                 pr     extproc('Qc3EncryptData')
```

Bug!

Use *DCLCASE to avoid retyping the name:

```
dcl-pr Qc3EncryptData extproc(*dclcase);
dcl-pr Qc3DecryptData extproc(*dclcase);
```

- Less error prone when coding
- Easier for code reviewers to see that it's correct

Subfields

Subfields officially start with the DCL-SUBF opcode

The opcode is optional unless the name is the same as a free-form opcode

```
dcl-ds info;
   name char(25);
   dcl-subf select int(10);
end-ds info;
```

DCL-SUBF must be used because "select" is an opcode supported in free-form

Same as the rule for EVAL and CALLP

```
name = 'Sally';
eval select = 5;
```

Subfields

The POS keyword replaces

- From-and-to positions
- OVERLAY(dsname)

```
D info     DS
D  sub1        25   34A
D  sub2                D     OVERLAY(info:100)
D  sub3                5P 2  OVERLAY(info)
```

```
dcl-ds info;
   sub1 char(10) pos(25);
   sub2 date pos(100);
   sub3 packed(5 : 2) pos(1);
end-ds info;
```

Subfields

Free-form OVERLAY only overlays subfields

- Use POS to overlay the data structure.
- No free-form equivalent for OVERLAY(ds:*NEXT)
- OVERLAY(ds:*NEXT) means "after all previous subfields" which is the same as not having the OVERLAY keyword at all
- SUB3 starts at position 101, after all previous subfields.

```
D info     DS
D  sub1       1   100A
D  sub2      11    20A
D  sub3             5A    OVERLAY(info:*next
)
```

Equivalent:

```
dcl-ds info;
   sub1 char(100) pos(1);   // 1-100
   sub2 char(10) pos(11);   // 11-20
   sub3 char(5);            // 101-105
```

Parameters

Parameters officially start with DCL-PARM

DCL-PARM is optional. Same rule as for subfields

```
dcl-pr proc;
   name char(25) const;
   dcl-parm clear ind value;
end-pr;
```

Can use named constants for keywords

Use named constants for keywords

```
dcl-c SYS_NAME_LEN 10;

dcl-ds sys_obj qualified;
   obj char(SYS_NAME_LEN);
   lib char(SYS_NAME_LEN);
end-ds;
```

Can use named constants for keywords

Some keywords allow literals to be specified without quotes in fixed form: DTAARA, EXTNAME, EXTFLD

This is not allowed in free-form

```
D name      C                'MYLIB/DTAARA1'
D data1a    S          10A   DTAARA(name)
D data1b    S          10A   DTAARA(*VAR:nameFld)

  dcl-s data2a char(10) dtaara('NAME');
  dcl-s data2b char(10) dtaara(nameFld);
  dcl-s data2c char(10) dtaara(name);
```

- DATA1A and DATA2A use *LIBL/NAME
- DATA1B and DATA2C use the value in nameFld
- DATA2C uses MYLIB/DTAARA1'

Procedure statements

Begin a procedure

- DCL-PROC
- Procedure name
- Keywords
- End with semicolon

```
dcl-proc myProc export;
```

End a procedure

- END-PROC
- Optional procedure name
- End with semicolon

```
end-proc myProc;
```

or

```
end-proc;
```

Procedure example

```
dcl-proc getCurUser export;
   dcl-pi *n char(10) end-pi;

   dcl-s curUser char(10) inz(*user);

   return curUser;
end-proc;
```

- The PI uses the place-holder *N for the name
- END-PI is specified as a keyword at the end of the DCL-PI statement

Gotchas

- Update does not imply delete
- END-DS, END-PR, END-PI needed at the end of a subfield or parameter list (even when there are no subfields or parameters)
- Keywords like DTAARA and EXTNAME that assume unquoted names are named constants or variables

(These have already been discussed)

Gotchas

If you are in the habit of using ellipsis at the end of D and P spec names

```
P customerName...
P             S                   50A
```

That will not work for free-form declarations

```
dcl-s customerName...
       char(50);
```

The name is customerNamechar, and "(50)" is found where the compiler expects to find the data type.

More information

PTFs

- PTF SI51094 for RPG compiler support
- DB2 Group PTF SF99701 level 26 for SQL precompiler support https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/IBM%20i%20Technology%20Updates/page/DB2%20for%20i%20TR7%20timed%20enhancements

Documentation

- There is a new PDF in the 7.1 Info Center with full documentation for the new free-form syntax
  - http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/topic/books/sc092508a.pdf
  - In the PDF, start at "What's New Since 7.1" in the "What's New" section

RPG Café wiki page

https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/We13116a562db_467e_bcd4_882013aec57a

Summary

We had two goals when designing the new free-form syntax

- Easy for non-RPG programmers to learn
- Easy for existing RPG programmers to learn

We hope we have accomplished those goals!

www.ibm.com/software/rational

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.

Revised September 26, 2006

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.

Revised December 2, 2010