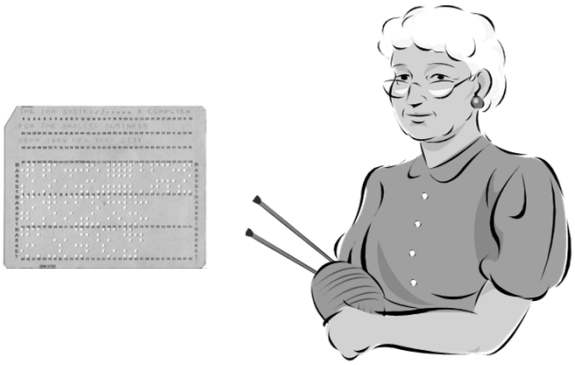
Get With The Program!

It's Not Your Grandma's RPG Anymore



Charles Guarino

Twitter @charlieguarino

Central Park Data Systems, Inc.

About The Speaker

With an IT career spanning over 30 years, Charles Guarino has been a consultant for most of them. Since 1995 he has been founder and President of Central Park Data Systems, Inc., a New York area based IBM midrange consulting company. In addition to being a professional speaker, he is a frequent contributor of technical and strategic articles and webcasts for the IT community. He is a proud member of COMMON's Speaker Excellence Hall of Fame and also Long Island Software and Technology Network's Twenty Top Techies of 2009. Charles currently serves as a member of COMMON's Strategic Education Team (SET) and is also Immediate Past President and monthly Q&A host of LISUG, a Long Island IBM i User's Group www.lisug.org.

Charles can be reached at cguarino@centralparkdata.com.

LinkedIn - http://www.linkedin.com/in/guarinocharles

Twitter - @charlieguarino

The Evolution of a Computer Language

Free Form RPG

RPG Open Access

RPG IV

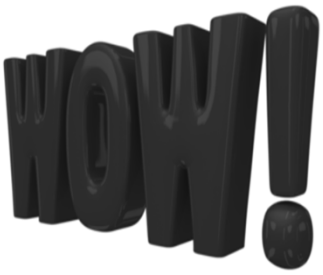
RPG 400

RPG III

RPG II

RPG

What's the BIG news?



Converted H, F, D and P specs to free-form.

/Free and /End-free are no longer required and are ignored by the compiler.

Works with any technology refresh of 7.1

File and Declaration specs can be intermingled.

Integrated new free-form enhancements into RDi.

What's important to know?

Statements always end with a semi-colon.

Free form calculations remain unchanged.

Source must be between columns 8 and 80.

I and O specs remain fixed format.

Can intermingle free form with fixed format.

What about the changes?

Works with ANY Technology Refresh of i7.1

Is NOT OS backward compatible – must be TGTRLS(*CURRENT).

Available with PTF SI51094 or later.
SQL Pre-Compiler available with DB2 PTF Group SF99701 level 26

Immediate Benefits

White Space

Non-RPG Programmers can understand

Code looks more modern

Can you figure out what these are?

CTL-OPT

DCL-F

DCL-S
DCL-DS
DCL-C
DCL-PR
DCL-PI
DCL-SUBF
DCL-PARM

DCL-PROC

Can you figure out what these are?

CTL-OPT	**control options – "H" spec**
DCL-F	**defines a file**
DCL-S	**defines a standalone field**
DCL-DS	**defines a data structure**
DCL-C	**defines a constant**
DCL-PR	**defines a prototype**
DCL-PI	**defines a procedure interface**
DCL-SUBF	**defines a subfield (*mostly* optional)**
DCL-PARM	**optional**
DCL-PROC	**defines a procedure**

Control specifications – KEY Points

CTL-OPT replaces H in column 6

Can be specified without any parameters.

Identical values from traditional H spec.

If keywords ACTGRP, BNDDIR or STGMDL are specified then DFTACTGRP(*NO) is assumed.

Control specifications (cont.)

Data Area Control Specifications

During program compilation a control spec is searched for in this sequence:

One included in your source code (including /copy)

A data area name RPGLEHSPEC in your library list

A data area named DFTLEHSPEC in library QRPGLE

Control specifications (cont.)

Before:

```
H bnddir('UTILITIES' : 'QC2LE') dftactgrp(*no) actgrp('QILE')
H option(*srcstmt : *nodebugio) debug(*input)
```

After:

```
ctl-opt bnddir('UTILITIES' : 'QC2LE') dftactgrp(*no) actgrp('QILE')
        option(*srcstmt: *nodebugio) debug(*input);
```

File specifications – KEY Points

DCL-F replaces F in column 6

Has some new built-in assumptions

Unless otherwise noted:
All files are assumed to be *DISK* files
All files are assumed to be *INPUT*
File are assumed to be *externally* defined

File specifications – A breakdown

Usage keywords:
*INPUT (default, not required)
*UPDATE (assumes INPUT)
*OUTPUT
*DELETE
(New! Implies *INPUT and UPDATE)

*UPDATE is not sufficient if you need to delete records
You MUST also specify *DELETE

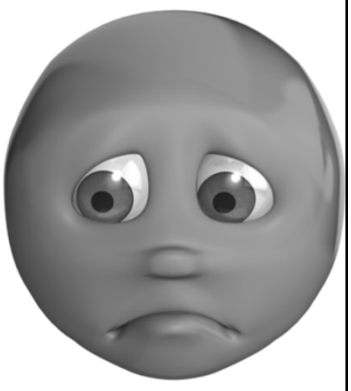
File specifications – Specify usage(*delete) to delete records

```
dcl-f custmast2 keyed  usage(*update);

 delete (01:123)  custmast2;
RNF5198: File in Factor 2 is not allowed for UPDATE or DELETE operation.


*inlr = *on;
return;
```

File specifications – A breakdown

What is NOT supported?

"IP" and "UP" – Primary files
"IS" and "US" – Secondary files
"IT" and "CT" – Table files
"O A" – Output with file addition



File specifications (cont.)

These are all the same to the compiler:

Before:

```
fcustmast  if   e           k disk
```

After:

```
dcl-f custmast disk(*ext) usage(*input) keyed;
```

- OR -

```
dcl-f custmast disk usage(*input) keyed;
```

- OR -

```
dcl-f custmast  usage(*input) keyed;
```

- OR -

```
dcl-f custmast  keyed;
```

File specifications (cont.)

Before:

```
 *  Custmast using procedure 'readproc' in service program 'DISKSVCPGM'
fcustmast  if   e           k disk    handler('DISKSVCPGM(readproc)')
```

After:

```
// Custmast using procedure 'readproc' in service program 'DISKSVCPGM'
 dcl-f  custmast  keyed  handler('DISKSVCPGM(readproc)');
```

File specifications (cont.)

Before:

```
 *  itemmast with ability to input and update and ADD records
fitemmast  uf a e           k disk    extdesc('devlib/itemmast')
f                                     extfile(*extdesc)
f                                     extmbr(production)
```

After:

```
dcl-f itemmast  usage(*update:*output) keyed
extdesc('devlib/itemmast')  extfile(*extdesc)   extmbr(production);
```

- OR -

```
dcl-f itemmast  usage(*update:*output)
                keyed
                extdesc('devlib/itemmast')
                extfile(*extdesc)
                extmbr(production);
```

File specifications (cont.)

Before:

```
 *  vendormst with ability to input and update AND DELETE records
fvendormst uf a e          k disk    prefix('vm') commit
```

After:

```
// vendormst with ability to input and update AND DELETE records
  dcl-f vendormst  usage(*update:*output:*delete)
                   keyed  prefix('vm')  commit;
```

- OR -

```
 dcl-f vendormst  usage(*update:*output:*delete)
                  keyed
                  prefix('vm')
                  commit;
```

File specifications (cont.)

Before:

```
fcusinqfm  cf   e           workstn usropn infds(cusinqds)
```

After:

```
dcl-f  cusinqfm workstn  usropn  infds(cusinqds);
```

- OR -

```
dcl-f  cusinqfm workstn
               usropn
               infds(cusinqds);
```

File specifications (cont.)

Before:

```
freport    o    e            printer oflind(*in98)
```

After:

```
dcl-f  report  printer oflind(*in98);
```

- OR -

```
dcl-f  report  printer
               oflind(*in98);
```

Definition specifications – KEY Points

DCL-xx replaces D in column 6

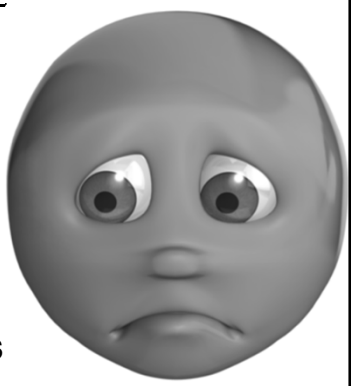
Several "blocks" have corresponding END-xx statement.

Ellipses are allowed but MUST continue a field name. You cannot specify ellipses and then follow with a keyword, the compile will combine the two to form a new field name.

Declaration specifications – A breakdown

What keywords are NOT supported?

CLASS
DATFMT
PROCPTR
TIMFMT
VARYING



***The above keywords have functional free form replacements**

FROMFILE (pre-runtime tables)
TOFILE (array or table file)

Definition specifications - Standalone fields

Character

```
D returndata       s                24
  dcl-s  returndata  char(24);
```

Character variable length

```
D returndata       s                24    varying
  dcl-s  returndata  varchar(24);
```

Zoned

```
D interestrate     S                5S 2 inz(5.5)
  dcl-s  interestrate   zoned(5:2) inz(5.5);
```

Packed

```
D vouchernumber    S                5P 2
  dcl-s  vouchernumber  packed(5:2);
```

Definition specifications - Standalone fields (cont.)

Indicator

```
D EndOfJob                          N   inz('0')
  dcl-s  EndOfJob  ind  inz('0');
```

Dates

```
D invoicedate                       D   datfmt(*mdy)
  dcl-s  invoicedate   date(*mdy);
```

*ISO is the default if *date* specified without a format

Zoned

```
D transtime                         T   timfmt(*usa)
  dcl-s   transtime  time(*usa);
```

*ISO is the default if *time* is specified without a format

Timestamp

```
D transtime                         Z
  dcl-s   transtime  timestamp;
```

Definition specifications - Constants

Before:

```
D arraysize       C                   const(200)
```

After:

```
dcl-c   arraysize  const(200);
```

- OR -

```
dcl-c   arraysize  200;
```

Definition specifications – Data structures

Requires a DCL-DS to start and END-DS to complete.

Unnamed data structures are represented with '*N"

END-DS statement can optionally contain name of data structure.

Program Status Data Structures are identified with new PSDS keyword

Definition specifications – Data structures (cont.)

External data structures are identified with EXT or EXTNAME keywords.

LDA previously defined as UDS is now identified as DTAARA(*AUTO)

Data areas processed with IN and OUT require *AUTO keyword.

In RDi, new data structure wizard still uses fixed-format.

Can optionally specify the name of the DS after END-DS

Definition specifications – Data structures (cont.)

Before:

```
D WeatherData     ds                  qualified
D Forecast                            likeds(WeatherDetail)
D                                     dim(7)
```

After:

```
dcl-ds  WeatherData     qualified;
   Forecast             likeds(WeatherDetail)
                        dim(7);
end-ds;
```

Definition specifications – When dcl-subf is required

Before:

```
D dspindicators   DS
D  endofjob                3       3n
D  return                 12      12n
D  subfileclear           60      60n
D  subfiledsp             62      62n
```

After:

```
dcl-ds dspindicators;
   endofjob          ind pos(3);
// optional dcl-subf is required when a field name is the same as an op code
   dcl-subf return   ind pos(12);
   subfileclear      ind pos(60);
   subfiledsp        ind pos(62);
end-ds;
```

Reserved words require dcl-subf keyword

Definition specifications – OVERLAY

Before:

```
D MonthlyData     ds
D  Month                          9
D  Sub1a                          1    overlay(month)
D  Sub1b                          1    overlay(month:*next)
D  Sub2a                          1    overlay(monthlydata)
D  Sub2b                          1    overlay(monthlydata:2)
D  Low                            3
D  High                           3
```

After:

```
dcl-ds MonthlyData;
  Month         char(9);
  sub1a         char(1) overlay(month);
  sub2a         char(1) overlay(month:*next);
  sub1b         char(1) pos(1);
  sub2b         char(1) pos(2);
  Low           char(3);
  High          char(3);
end-ds;
```

Definition specifications – Program Status Data Structure

Before:

```
D* Program status data structure
D programstatus  SDS                 QUALIFIED
D  ROUTINE         *ROUTINE                                  * Routine name
D  PARMS           *PARMS                                    * Num passed parms
D  EXCP_TYPE            40     42                           * Exception type
D  PGM_LIB              81     90                           * Program library
```

After:

```
dcl-ds  programstat  psds  qualified;
   routine       *routine;
   parms         *parms;
   excp_type     char(3)    pos(40);
   pgm_lib       char(10)   pos(81);

end-ds  programstat;
```

Definition specifications - Local data area

Here the LDA is updated automatically

```
d                 uds
d parameter1                1     10
```

- OR -

```
dcl-ds  *n dtaara(*auto);
  parameter1  char(10) pos(1);
 end-ds;

  parameter1 = '12345ABCDE';

 *inlr = *on;
 return;
```

Definition specifications – Updating a named data area

```
d nextordnum      ds            10     dtaara('NEXTORDNUM')
d counter                  1     10  0
```

- OR -

```
dcl-ds nextordnum   dtaara('NEXTORDNUM');
   counter    zoned(10);
end-ds;
```

```
/free
 in *lock  nextordnum;
   counter  +=1;

 out  nextordnum;
  *inlr = *on;
  return;
```

Definition specifications – Prototypes

Before:

```
D*------------------------------------------------
D* Program name: DisplayHourly
D* Purpose:       Launches www.weather.com with selected zip code
D*------------------------------------------------
D DisplayHourly   PR                  EXTPGM('HOURLYCL')
D  zipcodealpha                  5    const
```

After:

```
//* Purpose: Launches www.weather.com with selected zip code
//* Returns:
dcl-pr displayhourly extpgm('HOURLYCL');
zipcodealpha         char(5)  const;
end-pr;
```

Definition specifications – Prototypes (cont.)

Before:

```
D EventHandler    pr            10i 0
D commArea                        likeds(myCommArea)
D event                         10i 0 value
D return                          *   value
D stringlen                     20i 0 value
D exceptionID                   10i 0 value
```

After:

```
dcl-pr  EventHandler int(10);
  commArea              likeds(MyCommArea);
  event                 int(10)   value;
  dcl-parm return       pointer   value;
  dcl-parm stringlen    int(20)   value;
  exceptionID           int(10)   value;
  end-pr;
```

Definition specifications – Procedures and procedure interfaces

Procedures start with DCL-PR and end with END-PR.

Procedure interfaces require DCL-PI and END-PI.

The DCL and END statements can be on the same line.

If no name is used for the PI then *N must be specified.

Definition specifications – Procedures (cont.)

Before:

```
 * Procedure interface
p secretdata      b                   export
d                 pi            24
d datastring                    24    value
d direction                      1    value
d
```

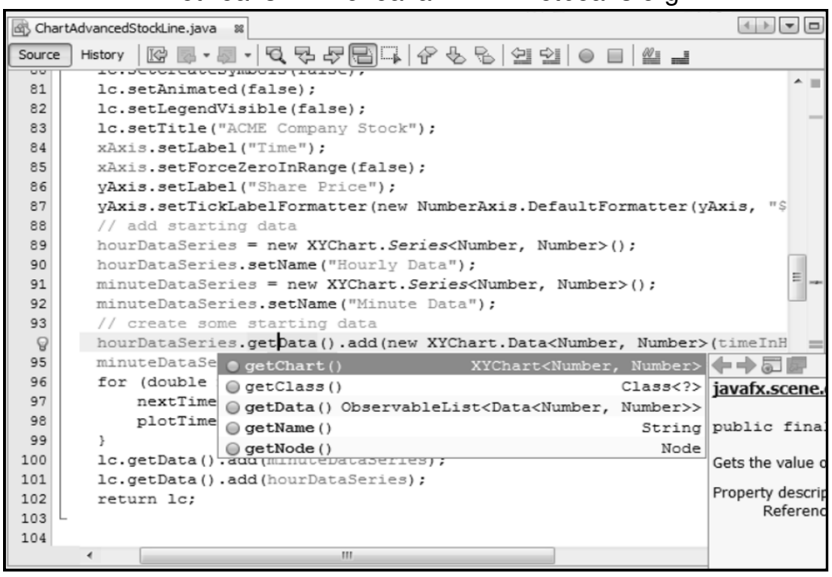
After:

```
 dcl-proc   secretdata export;
   dcl-pi  *n   char(24);
    datastring  char(24)  value;
    direction   char(1)   value;
  end-pi;

// Insert all your great code here

  end-proc  secretdata;
```

Is it just "compiler sugar" or is it something more?

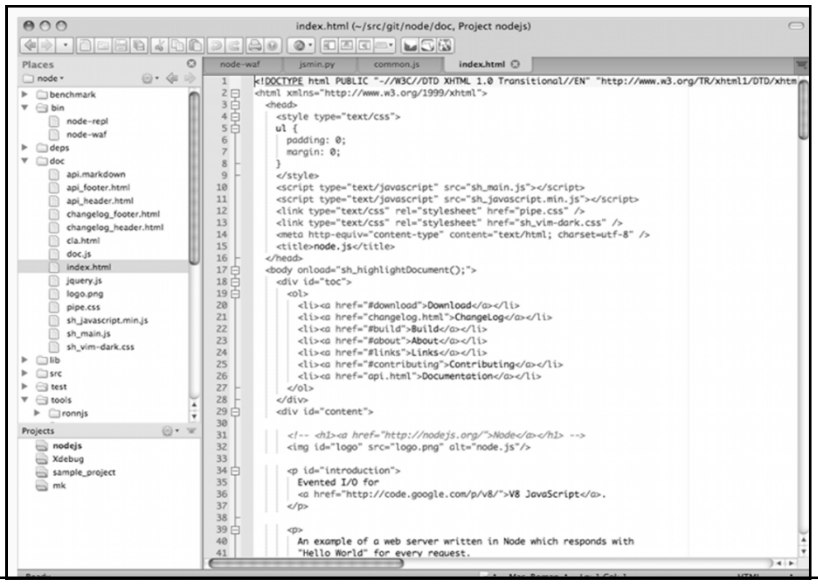
What are modern developers using today?

NetBeans IDE for Java – www.netbeans.org



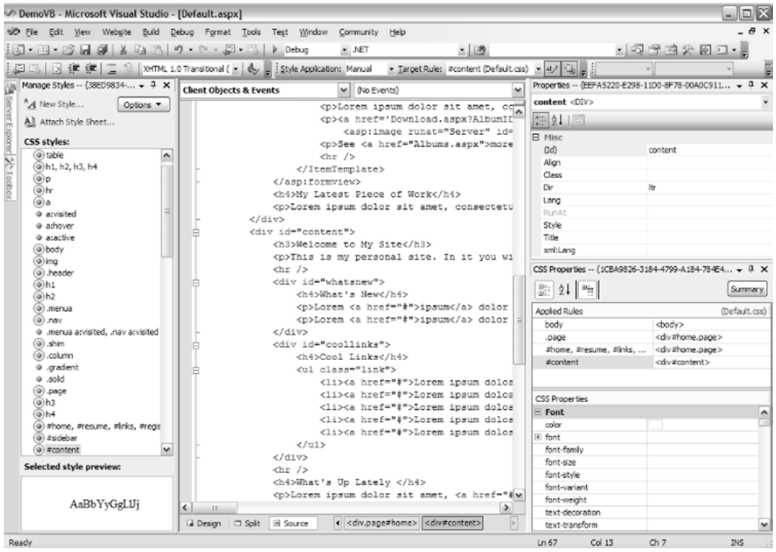
What are modern developers using today?

Komodo Edit - http://www.activestate.com/komodo-edit



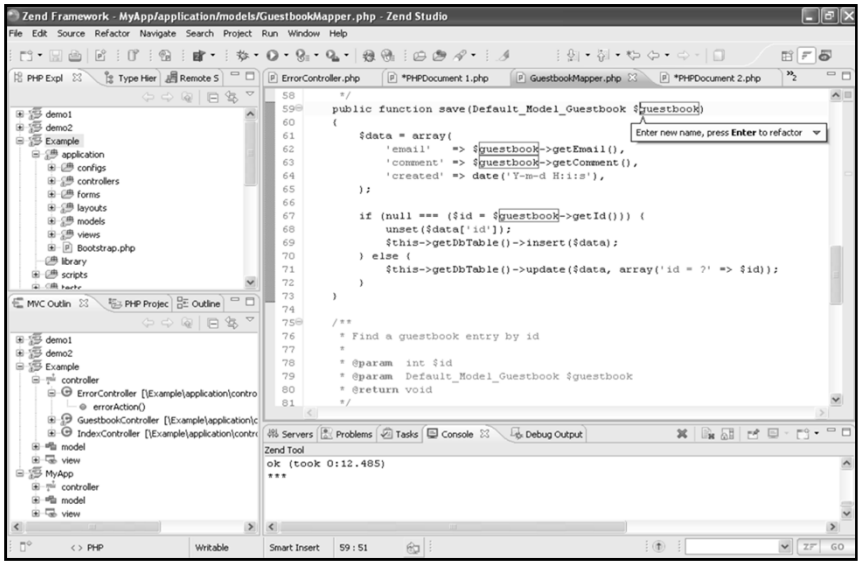
What are modern developers using today?

Visual Studio – www.visualstudio.com

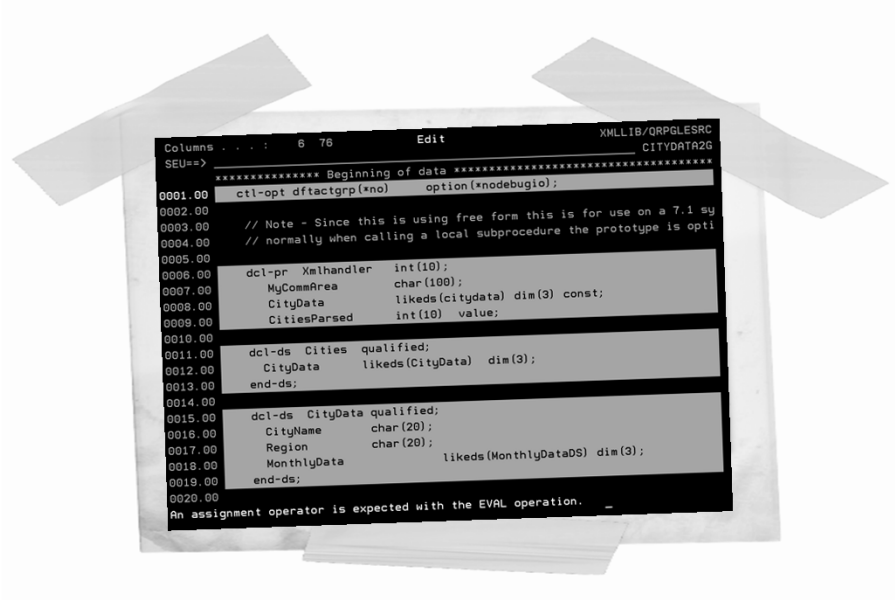


What are modern developers using today?

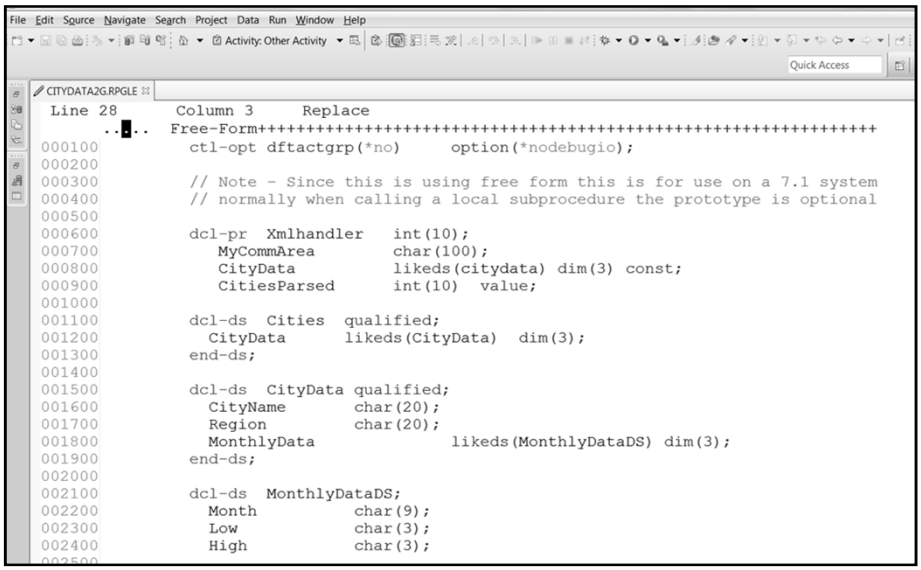
Zend Studio for PHP – www.zend.com



How do you get there from here?



Modern IBM i Developers use tooling that understands a modern language



What's the best way to learn free form RPG (and RDi)?

Best Ways to Learn a Language #1 - Immerse Yourself

4 Reasons Why Learning a New Language Requires Immersion

You learn aspects of language that cannot be replicated in a foreign classroom.

Do a little bit everyday to maintain fluency.

MOTIVATION IS STILL THE KEY

ADULTS CAN LEARN NEW LANGUAGES, HERE'S WHY

Putting it all together – a first side by side comparison

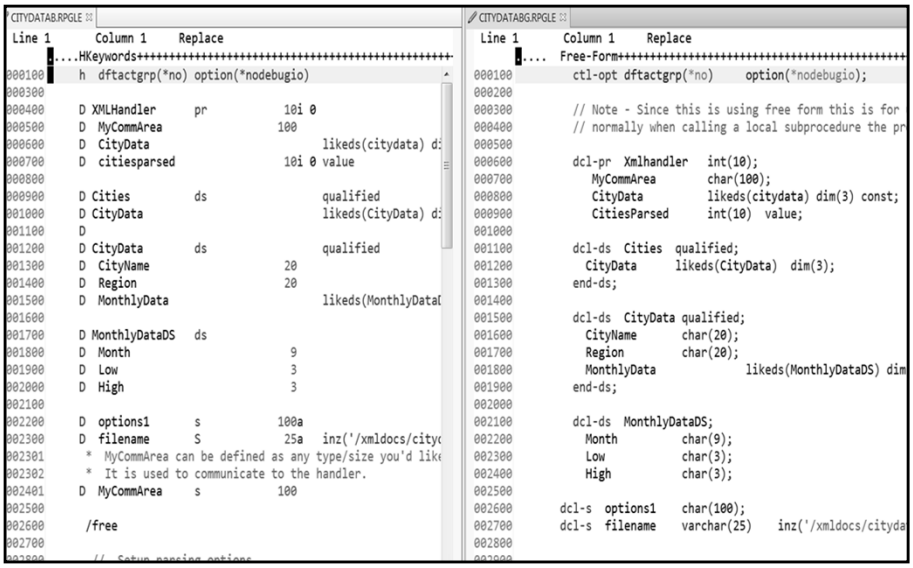
CITYDATA1.RPGLE

```
Line 32      Column 57    Replace
     .....  Free-Form++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
00100
00200     Fcitydata1 O  a e             disk    rename(citydata1:citydata
00300
00400     D Cities          ds                  qualified
00500     D CityData                            likeds(citydatads) dim(2)
00600     D
00700     D CityDatads      ds                  qualified
00800     D  CityName                     20
00900     D  Region                       20
01000     D  State                        20
01100     D  MonthlyData                        likeds(MonthlyData) dim(3
01200
01300     D MonthlyData     ds
01400     D  Month                         9
01500     D  Low                           3
01600     D  High                          3
01700
01800     D  options1       s             100a
01900     D  filename       S              25a   inz('/xmldocs/citydata1.x
02000     D  x              S               2  0
02100     D  y              S               2  0
02200
02300
02400      /free
02500
02600       //  Setup parsing options
02700       options1 = 'doc=file  +
02800         allowextra=yes allowmissing=yes case=any';
02900
03000       //  Initialize path to file in IFS
03100       filename = '/xmldocs/citydata1.xml';
03200
```

CITYDATA1G.RPGLE

```
Line 33      Column 55    Replace
     .....  Free-Form++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
000100
000200
000300       dcl-f citydata1  disk(*ext)
000400                        usage(*output)
000500                        rename(citydata1:citydatar);
000600
000700       dcl-ds cities qualified;
000800         CityData    likeds(CityDataDS) dim(2);
000900       end-ds;
001000
001100       dcl-ds CityDataDS  qualified;
001200         CityName      char(20);
001300         Region        char(20);
001400         State         char(20);
001500         MonthlyData             likeds(MonthlyData)  dim(3);
001600       end-ds;
001700
001800       dcl-ds MonthlyData;
001900         Month         char(9);
002000         Low           char(3);
002100         High          char(3);
002200       end-ds;
002300
002400       dcl-s  options1   char(100);
002500       dcl-s  filename   varchar(25)  inz('/xmldocs/citydata1.xml');
002600       dcl-s  x          packed(2:0);
002700       dcl-s  y          packed(2:0);
002800
002900
003000       //  Setup parsing options
003100       options1 = 'doc=file  +
003200         allowextra=yes allowmissing=yes case=any';
```

Putting it all together – comparison #2



CITYDATAB.RPGLE

```
Line 1      Column 1     Replace
    ....+....HKeywords++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
000100     h  dftactgrp(*no) option(*nodebugio)
000300
000400     D XMLHandler      pr            10i 0
000500     D  MyCommArea                   100
000600     D  CityData                           likeds(citydata) d
000700     D  citiesparsed                10i 0 value
000800
000900     D Cities          ds                  qualified
001000     D CityData                            likeds(CityData) d
001100     D
001200     D CityData        ds                  qualified
001300     D  CityName                     20
001400     D  Region                       20
001500     D  MonthlyData                        likeds(MonthlyData
001600
001700     D MonthlyDataDS   ds
001800     D  Month                         9
001900     D  Low                           3
002000     D  High                          3
002100
002200     D  options1       s             100a
002300     D  filename       S              25a   inz('/xmldocs/city
002301      *  MyCommArea can be defined as any type/size you'd like
002302      *  It is used to communicate to the handler.
002401     D  MyCommArea     s             100
002500
002600      /free
002700
```

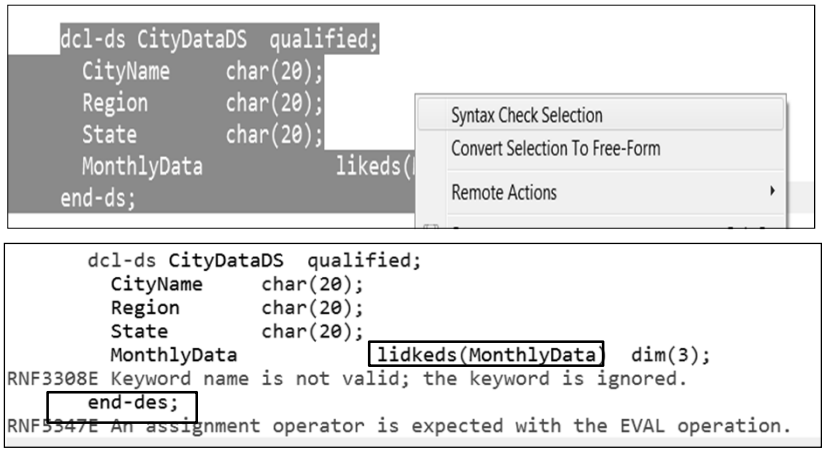
CITYDATABG.RPGLE

```
Line 1      Column 1     Replace
    ....+....  Free-Form++++++++++++++++++++++++++++++++++++++++++++++++++++++++
000100         ctl-opt dftactgrp(*no)     option(*nodebugio);
000200
000300         // Note - Since this is using free form this is for
000400         // normally when calling a local subprocedure the pr
000500
000600         dcl-pr  Xmlhandler   int(10);
000700            MyCommArea        char(100);
000800            CityData          likeds(citydata) dim(3) const;
000900            CitiesParsed      int(10)  value;
001000
001100         dcl-ds  Cities  qualified;
001200           CityData     likeds(CityData)  dim(3);
001300         end-ds;
001400
001500         dcl-ds  CityData qualified;
001600           CityName       char(20);
001700           Region         char(20);
001800           MonthlyData              likeds(MonthlyDataDS) dim
001900         end-ds;
002000
002100         dcl-ds  MonthlyDataDS;
002200           Month          char(9);
002300           Low            char(3);
002400           High           char(3);
002500
002600       dcl-s  options1    char(100);
002700       dcl-s  filename    varchar(25)    inz('/xmldocs/cityda
002800
```

Syntax Check Selection is your friend

Select any block of code and right click.

Any errors will appear immediately below the selected block.



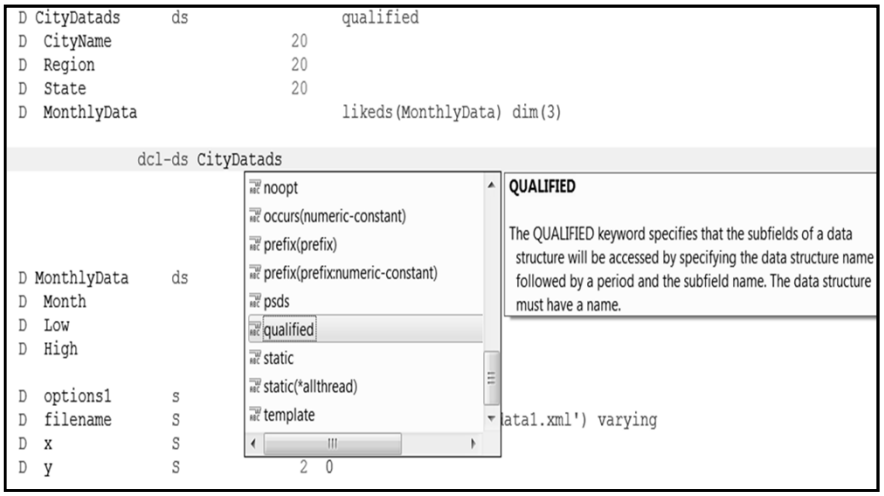
```
    dcl-ds CityDataDS  qualified;
      CityName     char(20);
      Region       char(20);
      State        char(20);
      MonthlyData            likeds(
    end-ds;
```

Syntax Check Selection
Convert Selection To Free-Form
Remote Actions

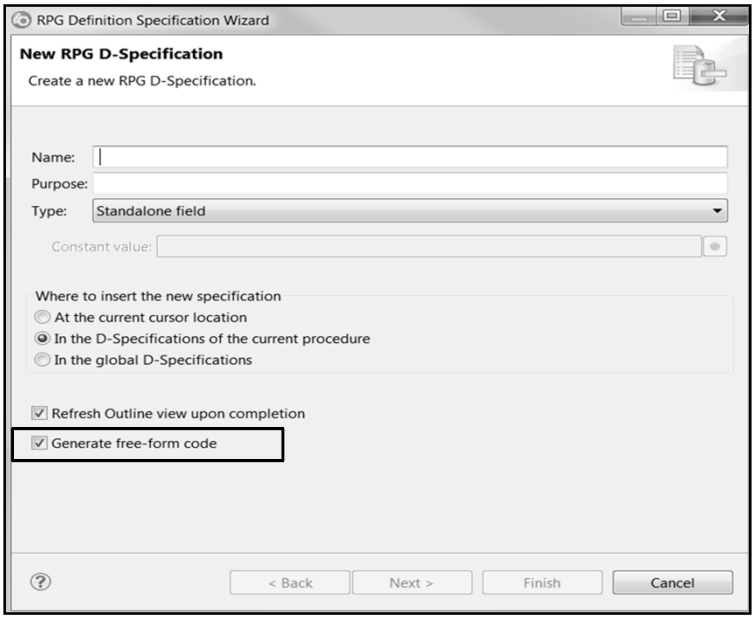
```
      dcl-ds CityDataDS  qualified;
        CityName     char(20);
        Region       char(20);
        State        char(20);
        MonthlyData            likeds(MonthlyData)  dim(3);
RNF3308E Keyword name is not valid; the keyword is ignored.
      end-des;
RNF5347E An assignment operator is expected with the EVAL operation.
```

Writing free form code using RDi (must have version 9.0.1 min)

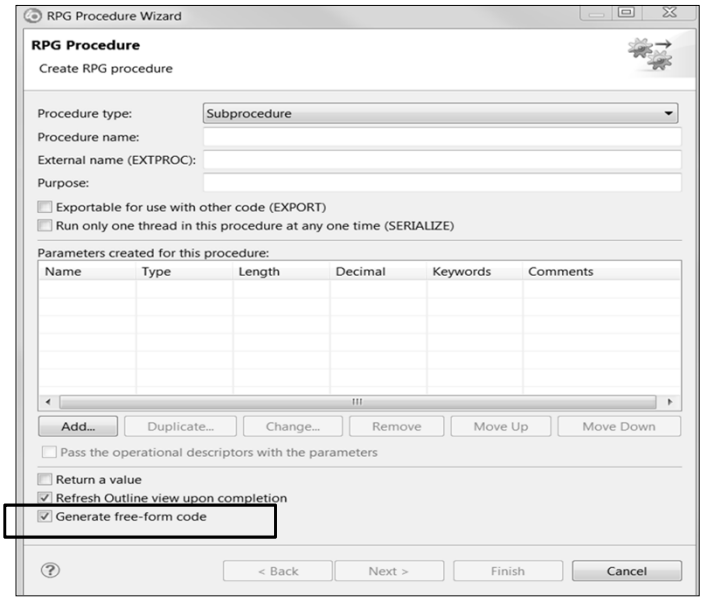
Content assist (CTRL-SPACE) helps you write your code



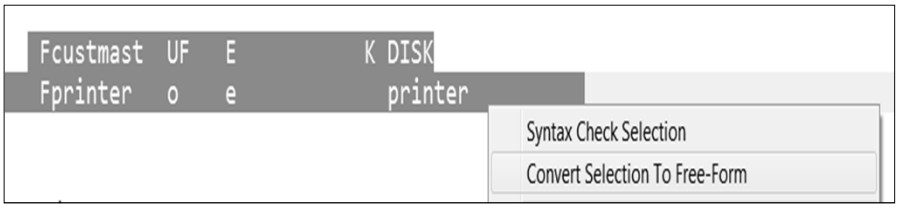
RDi Wizards have been updated as well! D specs or Free Form?



New Procedure Wizard in RDi



Convert Selection to Free-form still only for calcs

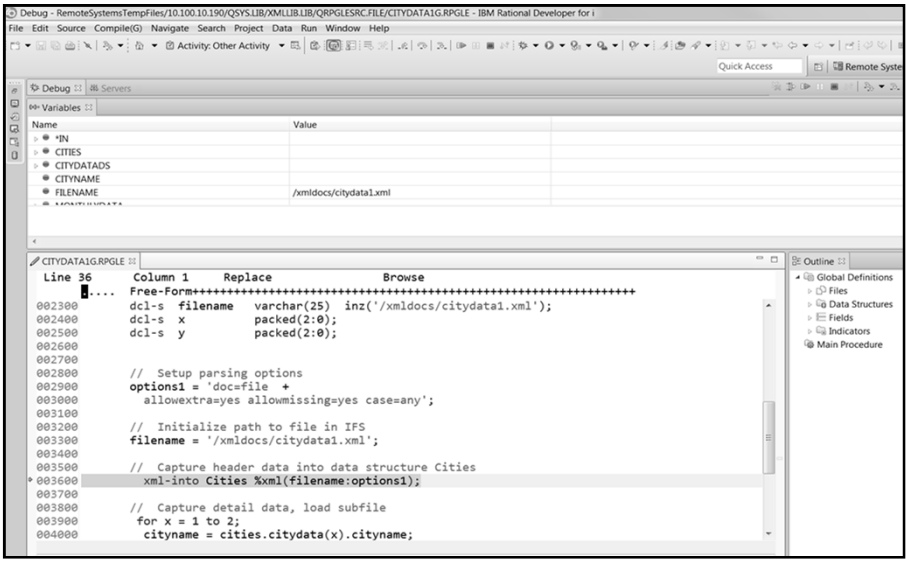



```
Fcustmast  UF   E           K DISK
Fprinter   o    e             printer

Conversion complete.
```

Debugging free form using green screen debugger

```
                          Display Module Source

Program:   CITYDATA1G     Library:   XMLLIB         Module:   CITYDATA1G
    16          dcl-ds MonthlyData;
    17            Month         char(9);
    18            Low           char(3);
    19            High          char(3);
    20          end-ds;
    21
    22          dcl-s  options1   char(100);
    23          dcl-s  filename   varchar(25)  inz('/xmldocs/citydata1.xml');
    24          dcl-s  x          packed(2:0);
    25          dcl-s  y          packed(2:0);
    26
    27
    28          //  Setup parsing options
    29          options1 = 'doc=file  +
    30            allowextra=yes allowmissing=yes case=any';
                                                                      More...
Debug . . .

F3=End program   F6=Add/Clear breakpoint   F10=Step   F11=Display variable
F12=Resume       F17=Watch variable        F18=Work with watch   F24=More keys
Breakpoint at line 29
```

Debugging free form using RDi

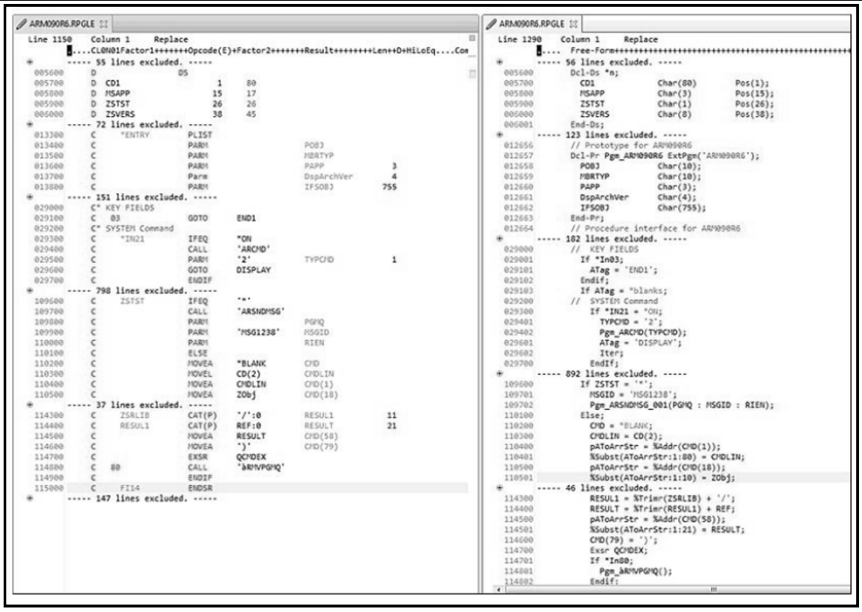


The Power of Automation

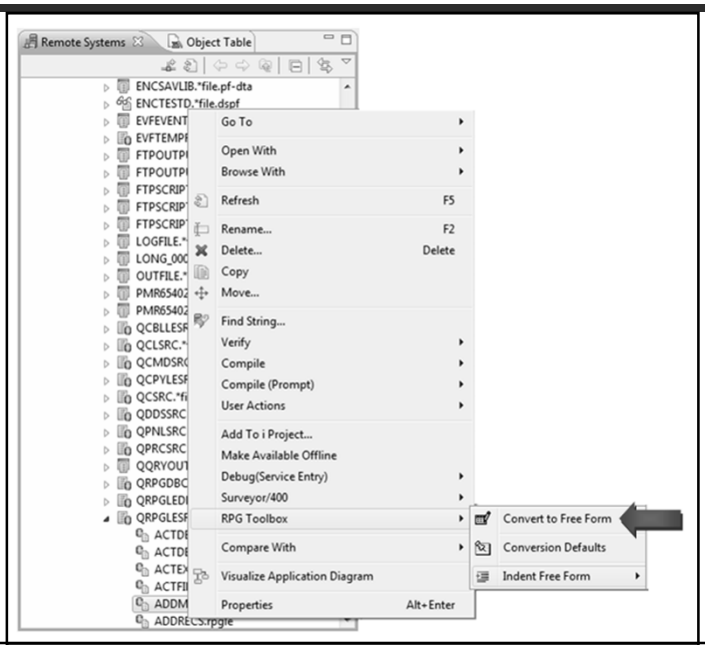
```
.DName++++++++++ETDsFrom+++To/L+++IDc.Keywords++++++++++++++++++++++
h  dftactgrp(*no) option(*nodebugio)

D XMLHandler      pr            10i 0
D  MyCommArea                  100
D  CityData                          likeds(citydata) dim(3) const
D  citiesparsed                10i 0 value
```

```
Free-Form++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
  ctl-opt dftactgrp(*no)      option(*nodebugio);

  dcl-pr  Xmlhandler    int(10);
     MyCommArea         char(100);
     CityData           likeds(citydata) dim(3) const;
     CitiesParsed       int(10)  value;
```

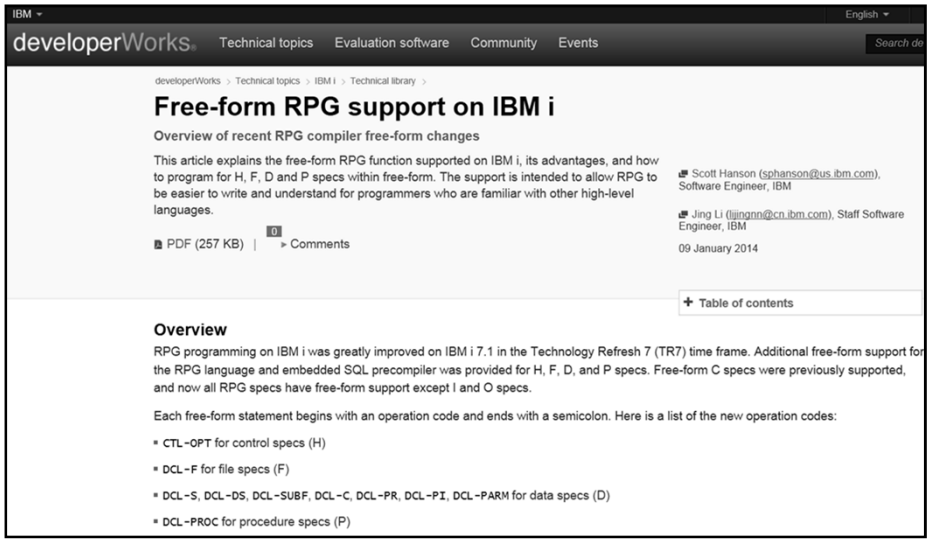
ARCAD – Transformer RPG www.arcadsoftware.com



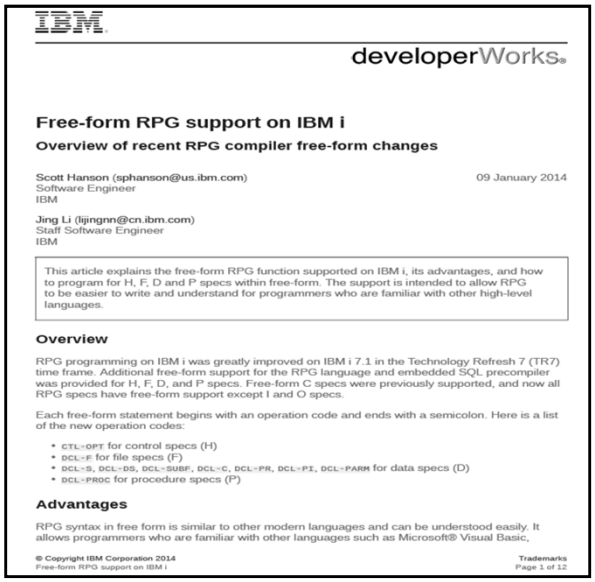
Linoma RPG Toolbox – www.linoma.com



Google “DeveloperWorks Free Form RPG”



Google “DeveloperWorks Free Form RPG”



IBM

developerWorks

Free-form RPG support on IBM i

Overview of recent RPG compiler free-form changes

Scott Hanson (sphanson@us.ibm.com)
Software Engineer
IBM

Jing Li (lijingnn@cn.ibm.com)
Staff Software Engineer
IBM

09 January 2014

This article explains the free-form RPG function supported on IBM i, its advantages, and how to program for H, F, D and P specs within free-form. The support is intended to allow RPG to be easier to write and understand for programmers who are familiar with other high-level languages.

Overview

RPG programming on IBM i was greatly improved on IBM i 7.1 in the Technology Refresh 7 (TR7) time frame. Additional free-form support for the RPG language and embedded SQL precompiler was provided for H, F, D, and P specs. Free-form C specs were previously supported, and now all RPG specs have free-form support except I and O specs.

Each free-form statement begins with an operation code and ends with a semicolon. Here is a list of the new operation codes:

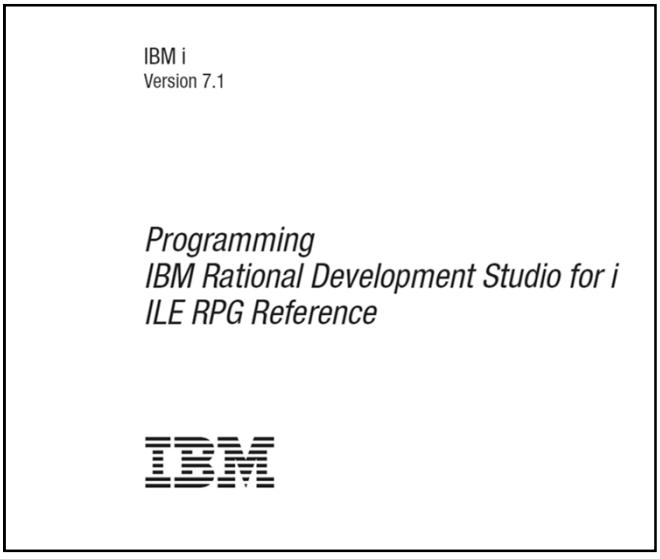
- `CTL-OPT` for control specs (H)
- `DCL-F` for file specs (F)
- `DCL-S`, `DCL-DS`, `DCL-SUBF`, `DCL-C`, `DCL-PR`, `DCL-PI`, `DCL-PARM` for data specs (D)
- `DCL-PROC` for procedure specs (P)

Advantages

RPG syntax in free form is similar to other modern languages and can be understood easily. It allows programmers who are familiar with other languages such as Microsoft® Visual Basic,

Where to learn more? – IBM i Information Center 7.1

http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/topic/books/sc092508a.pdf



IBM i
Version 7.1

Programming
IBM Rational Development Studio for i
ILE RPG Reference

IBM

Where to learn more? (cont.)

http://pic.dhe.ibm.com/infocenter/iseries/v7r1m0/topic/books/sc092508a.pdf

What's New since 7.1?

This section describes the enhancements made to ILE RPG after 7.1.

Free-form Control, File, Definition, and Procedure statements

- Free-form Control statements begin with CTL-OPT and end with a semicolon. See "Free-Form Control Statement" on page 5-14.

```
CTL-OPT OPTION(*SRCSTMT : *NODEBUGIO)
        ALWNULL(*USRCTL);
```

- Free-form File definition statements begin with DCL-F and end with a semicolon. See "Free-Form File Definition Statement" on page 5-38.

  The following statements define three files

  1. An externally-described DISK file opened for input and update.
  2. An externally-described WORKSTN file opened for input and output.
  3. A program-described PRINTER file with record-length 132.

```
DCL-F custFile usage(*update) extfile(custFilename);
DCL-F screen workstn;
DCL-F qprint printer(132) oflind(qprintOflow);
```

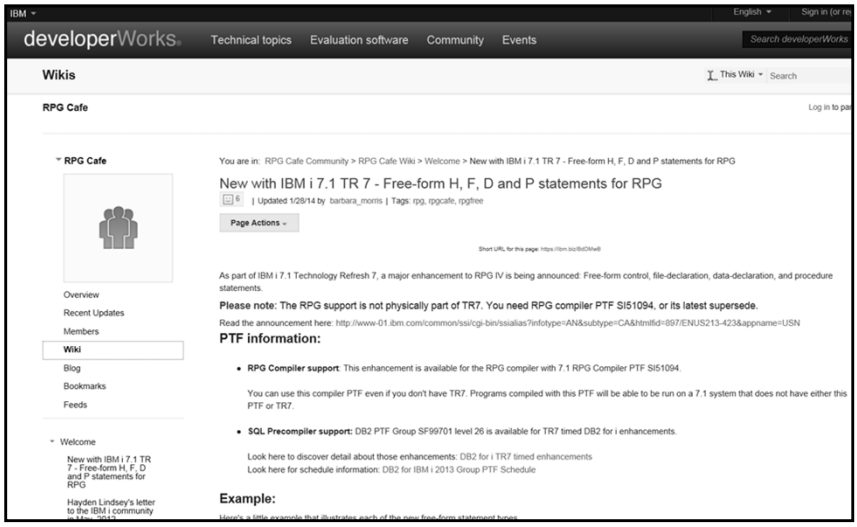
- Free-form data definition statements begin with DCL-C, DCL-DS, DCL-PI, DCL-PR, or DCL-S, and end with a semicolon. See "Free-Form Definition Statement" on page 5-74.

  The following statements define several items

  1. A named constant *MAX_ELEMS*.
  2. A standalone varying length character field *fullName*.

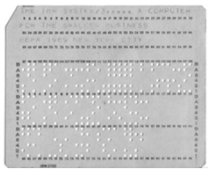
Where to learn more? - IBM's RPG Cafe

Search for "IBM i DeveloperWorks RPG 7.1"



Get With The Program!

It's Not Your Grandma's RPG Anymore



Charles Guarin

THANK YOU!!!