

About The Speaker

Charles Guarino has been an Information Technology consultant for the vast majority of his career. He is also the founder and President of Central Park Data Systems, Inc., a Long Island and New York City area based IBM midrange consulting and corporate training company. In addition, he is a professional speaker presenting at technical conferences and events across the United States and Europe. He is a frequent contributor of articles, webcasts and videos for the IBM i community, with topics covering a broad range of IT topics and strategies. Charles is a member of COMMON's Speaker Excellence Hall of Fame and The Long Island Software and Technology Network's Twenty Top Techies. More recently Charles became the proud recipient of the Al Barsa Memorial Scholarship Award. Additionally, he currently serves as a board member of COMMON and also participates on COMMON's Strategic Education Team (SET). Other professional endeavors have included being the president and monthly Q&A host of the Long Island System User's Group LISUG (www.lisug.org)

Charles can be reached at cguarino@centralparkdata.com.
LinkedIn - <http://www.linkedin.com/in/guarinocharles>
Twitter - [@charlieguarino](https://twitter.com/charlieguarino)

What We'll Cover ...

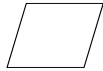
- **Let's Get Started**
- **Preparing to Submit**
- **The Debugger Comes to Life**
- **Setting a Service Entry Point**
- **RDl Debugging Hacks**
- **Wrap-up**

```
CREATE TABLE CUSTMAST (  
    CCMP DECIMAL(2, 0) NOT NULL DEFAULT 0 ,  
    CCUSNO DECIMAL(7, 0) NOT NULL DEFAULT 0 ,  
    CCSNAM CHAR(30) CCSID 37 NOT NULL DEFAULT " ,  
    CADDR1 CHAR(30) CCSID 37 NOT NULL DEFAULT " ,  
    CADDR2 CHAR(30) CCSID 37 NOT NULL DEFAULT " ,  
    CADDR3 CHAR(30) CCSID 37 NOT NULL DEFAULT " ,  
    CYTDSL DECIMAL(7, 0) NOT NULL DEFAULT 0 ,  
    CYTSLA DECIMAL(7, 0) NOT NULL DEFAULT 0 ,  
    CYTSLB DECIMAL(7, 0) NOT NULL DEFAULT 0 ,  
    CYTSLC DECIMAL(7, 0) NOT NULL DEFAULT 0 ,  
    CAVGSAL DECIMAL(7, 0) NOT NULL DEFAULT 0 ,  
    CDTLSSL DATE NOT NULL DEFAULT '0001-01-01' ,  
    CDTLSPM DATE NOT NULL DEFAULT '0001-01-01' ,  
    CSTRLEN DECIMAL(5, 0) NOT NULL DEFAULT 0 ,  
    CPGMRUN DECIMAL(5, 0) NOT NULL DEFAULT 0 ,  
    CYTDPRF DECIMAL(7, 0) NOT NULL DEFAULT 0 ,  
    CCLRDATA CHAR(24) CCSID 37 NOT NULL DEFAULT " ,  
    CENCDATA CHAR(24) CCSID 37 NOT NULL DEFAULT " )  
  
    RCD_FMT CUSTMSTR ;  
  
LABEL ON TABLE CUSTMAST  
    IS 'Customer Master File' ;
```

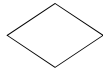
Program we will be debugging



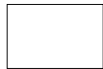
Start program



Read a record from file CUSTMAST



If %EOF, leave program loop and exit program



Call encryption service program, pass either cleartext or ciphertext depending on incoming parm



Update CUSTMAS



Read more records from file CUSTMAST

Program we will be debugging (cont.)

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

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

// Prototypes
dcl-pr encdebugm extpgm;
      *n      char(1);
end-pr;

dcl-pr secretdata char(24);
      *n      char(24) value;
      *n      char(1) value;
end-pr;

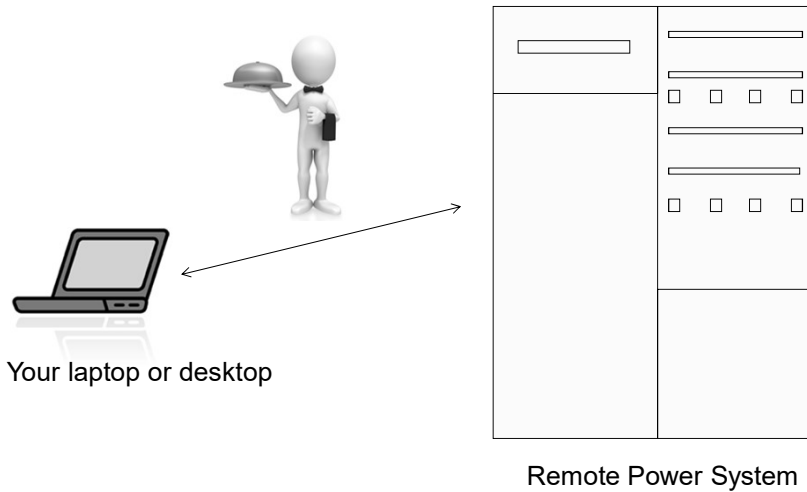
// Procedure interface
dcl-pi encdebugm;
      directionin char(1);
end-pi;

dcl-s cleardata char(24);
dcl-s encrypteddata char(24);
dcl-s direction char(1);

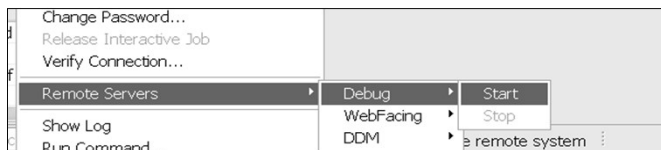
// If called without any parameters, default to (E)ncrypt
if %parms > *zero;
      direction = directionin;
endif;
```

Introducing the Debug Server

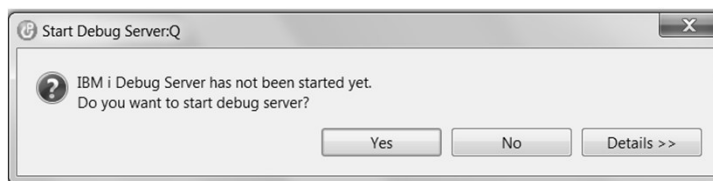
- You enter debugging commands in RDi
 - The commands execute on the Remote System



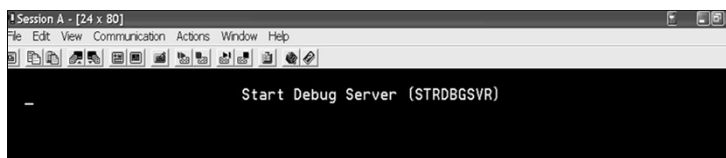
Starting the Debug Server (3 different ways!)



- OR -



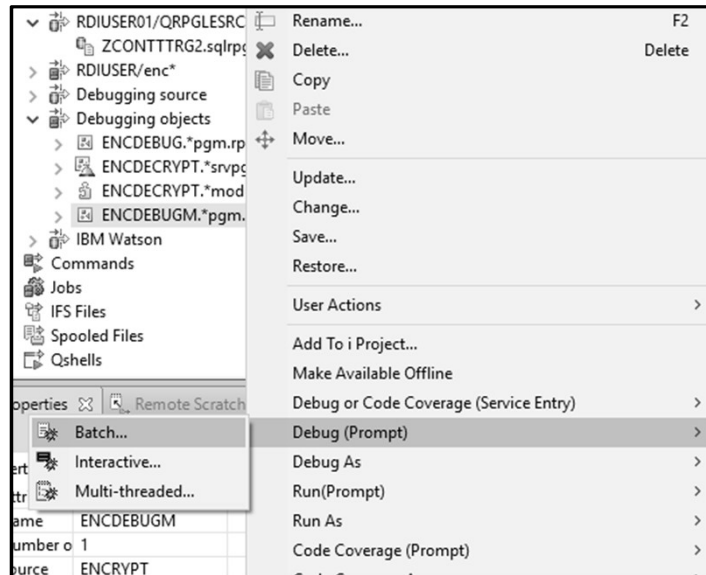
- OR -



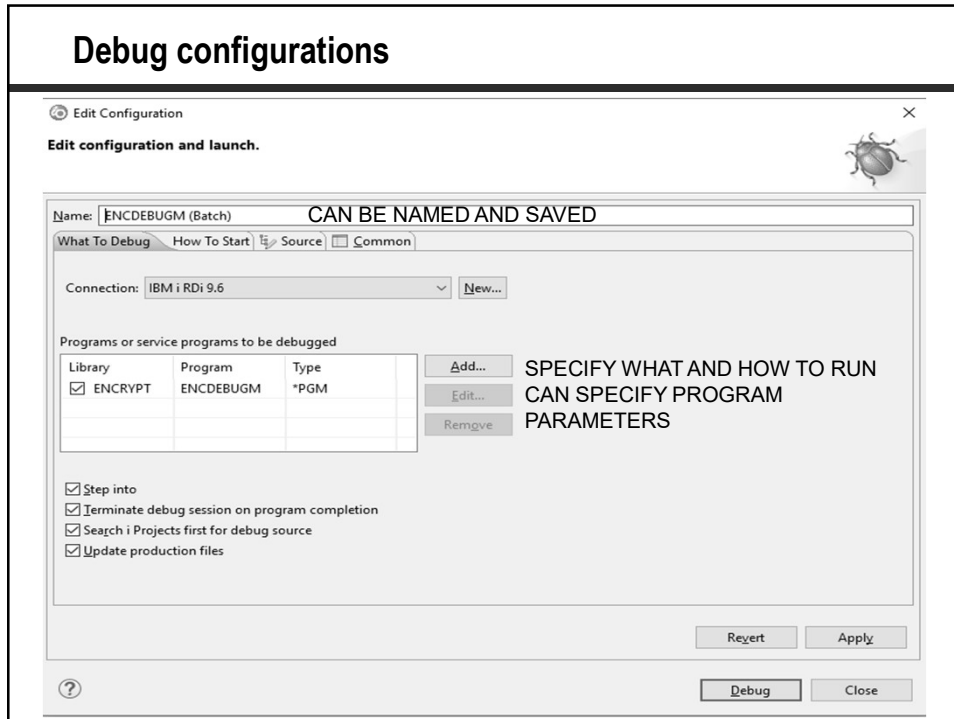
What We'll Cover ...

- Let's Get Started
- **Preparing to Submit**
- The Debugger Comes to Life
- Setting a Service Entry Point
- RDi Debugging Hacks
- Wrap-up

Calling and debugging with a prompt

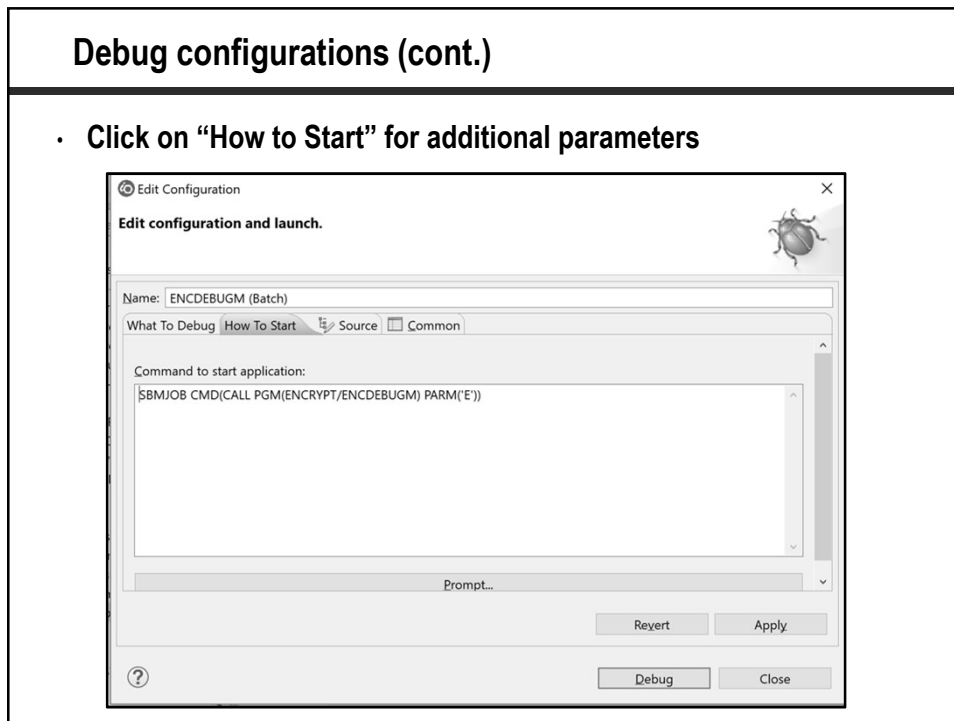


Debug configurations



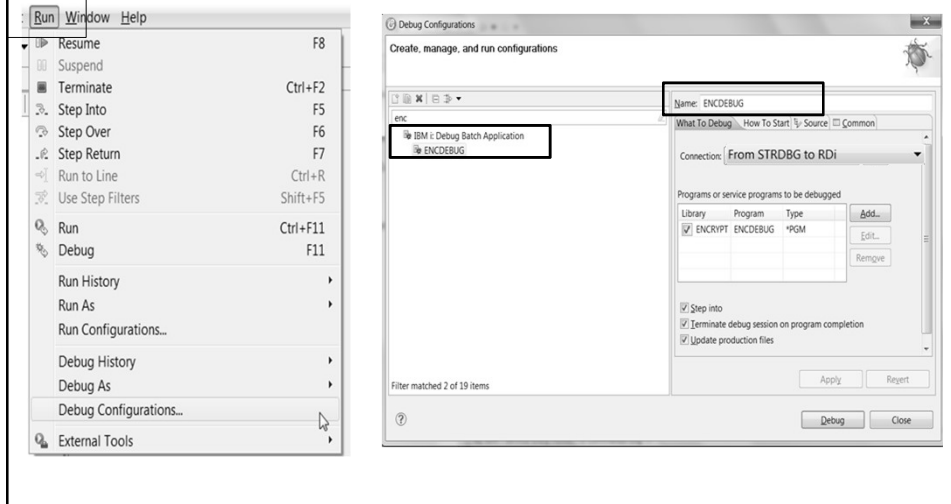
Debug configurations (cont.)

- Click on “How to Start” for additional parameters

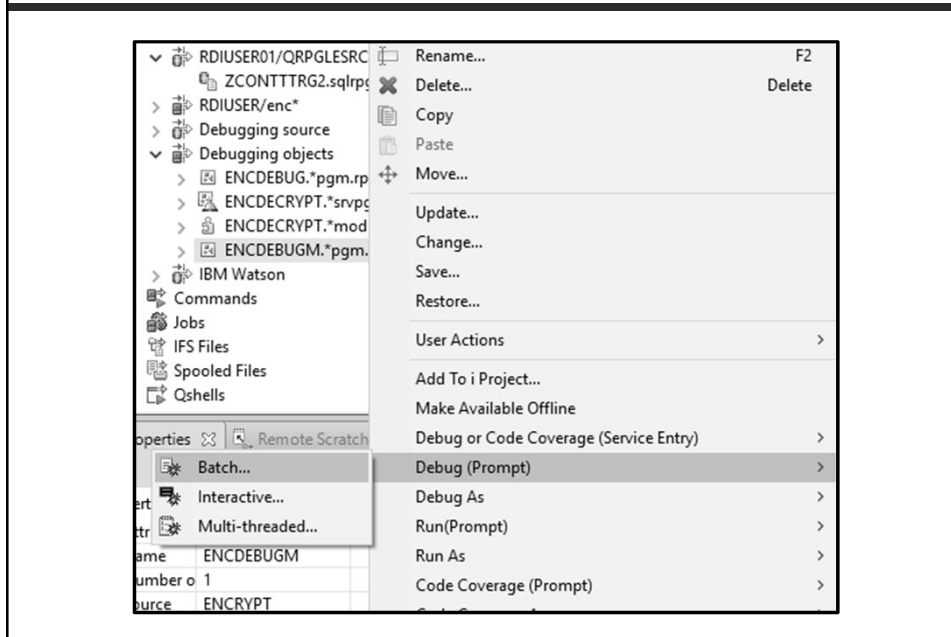


Launching an existing configuration

- Very useful if you will be debugging programs multiple times
- The configuration will remember all of your settings



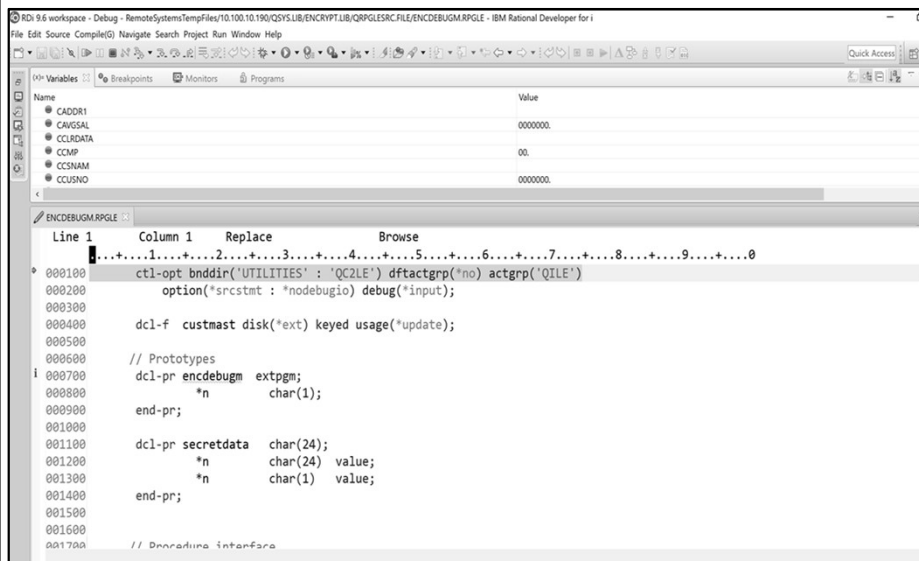
Calling and debugging with a prompt



What We'll Cover ...

- Let's Get Started
- Preparing to Submit
- **The Debugger Comes to Life**
- Setting a Service Entry Point
- RDi Debugging Hacks
- Wrap-up

Welcome to the DEBUG perspective



The screenshot shows the IBM Rational Developer for i interface. At the top, there is a menu bar with options: File, Edit, Source, Compile(G), Navigate, Search, Project, Run, Window, Help. Below the menu bar is a toolbar with various icons for debugging and development. The main window is divided into several panes. On the left, there is a 'Variables' pane showing a list of variables with their names and values:

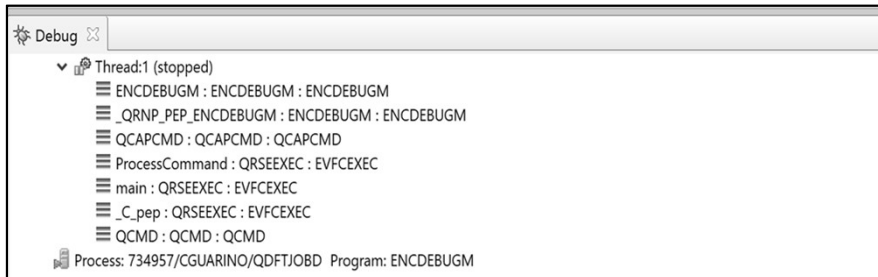
Name	Value
CADDR1	
CAVGSAL	00000000.
CCLRDATA	
CCMP	00.
CCSNAM	
CCUSNO	00000000.

Below the variables pane is a source code editor showing the contents of the file ENCODEBUG.MRPLE. The code is as follows:

```
Line 1      Column 1      Replace      Browse
...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8...+...9...+...0
000100    ct1-opt bnddir('UTILITIES' : 'QC2LE') dftactgrp(*no) actgrp('QILE')
000200    option(*srcstmt : *nodebugio) debug(*input);
000300
000400    dcl-f custmast disk(*ext) keyed usage(*update);
000500
000600    // Prototypes
i 000700    dcl-pr encdebugm extpgm;
000800        *n      char(1);
000900    end-pr;
001000
001100    dcl-pr secretdata char(24);
001200        *n      char(24) value;
001300        *n      char(1) value;
001400    end-pr;
001500
001600
001700    // Procedure interface
```

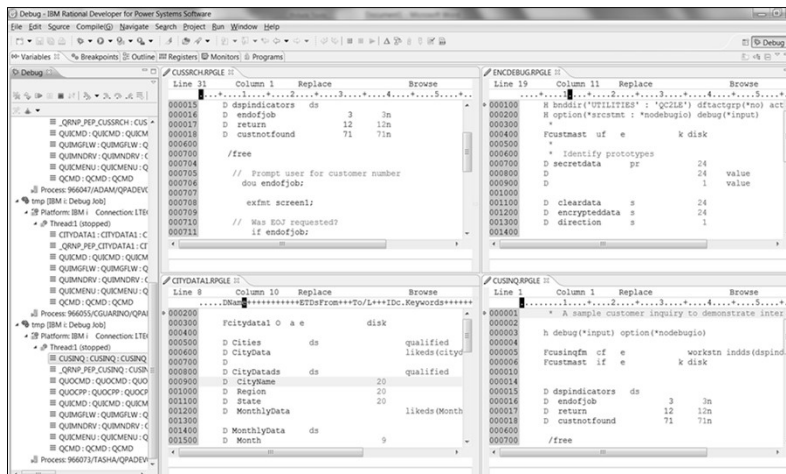

Introducing the DEBUG view

- This is the call stack and communication area between RDi and the IBM i
- Can be used to debug multiple jobs at the same time
 - Simply click on the job you want to debug



Debugging multiple jobs at once

- When debugging multiple jobs at once keep the debug view open
 - Makes it easier to keep track of current job being debugged



Introducing the VARIABLES view

- All program variables are displayed and updated in real time
 - Each variable will change color when its value changes
- This view is customizable using the drop-down menu
- Right click to change view and add to monitors view
- Values can be changed by simply over-typing

Name	Value
*IN	
CADDR1	
CADDR2	
CADDR3	
CAVGSAL	0000000.
CCLRDATA	
CCMP	00.
CCSNAM	
CCUSNO	0000000.
CDTLSPM	0001-01-01
CDTLSSL	0001-01-01
CENCDATA	
CLEARDATA	
CPGMRUN	00000.
CSTRLEN	00000.
CYTDPRF	0000000.
CYTDSL	0000000.
CYTDSLA	0000000.
CYTDSLB	0000000.
CYTDSL	0000000.
DIRECTION	
ENCRYPTEDDATA	

Introducing the VARIABLES view – drop down menu

- All program variables are displayed and updated in real time
 - Each variable will change color when its value changes
- This view is customizable using the drop-down menu
- Right click to change view and add to monitors view
- Values can be changed by simply over-typing

Value
123 Main Street
Minneapolis
0000000.
thisisCLEARdata
01.
Worlds Fair Main Processing
0037828.
0001-01-01

Green screen equivalent to variables view!

- Type the debug command `EVAL %LOCALVARS` to see all variables!

```
Evaluate Expression

Previous debug expressions

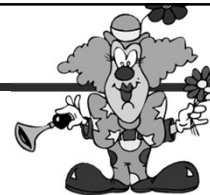
*IN(98) = '0'
*IN(99) = '0'
CADDR1 = '18 Ridgepage Road
CADDR2 = 'Ohio
CADDR3 = '
CAVGSAL = 0045678.
CCLRDATA = '1234567890
CCMP = 01.
CCSNAM = 'John's Hardware Store
CCUSNO = 0000145.
CDTLSPM = '2009-12-22'
CDTLSSL = '2009-12-01'
CENCDATA = 'S1 B 0Wg8}0P 90 5 *'
CLEARDATA = '1234567890

More...

Debug . . . eval %localvars

F3=Exit  F9=Retrieve  F12=Cancel  F16=Repeat find  F19=Left  F20=Right
F21=Command entry  F23=Display output
```

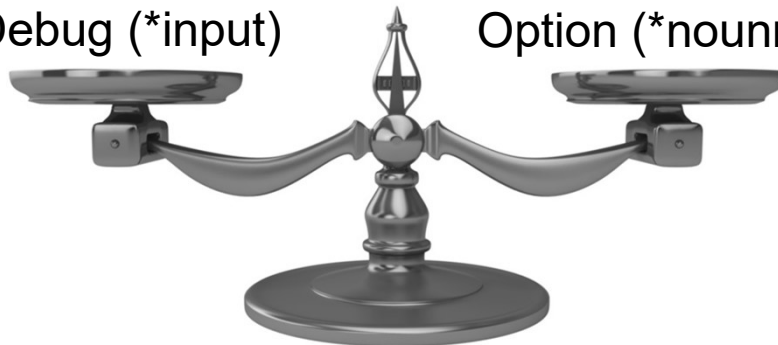
Unreferenced Field Fun Fact!



RNF7031

Debug (*input)

Option (*nounref)



Field hovering

- Position the cursor over a field and its value appears.
- Much easier than typing “ev cleardata” or pressing F11!



```
dcl-s cleardata char(24);
dcl-s encrypteddata char(24);
dcl-s direction char(1);

read custmast;
dow not %eof (custmast);

direction = 'E'; // Value of 'E' tells procedure to ENCRYPT
cleardata = cclrdata;
encrypteddata = secretdata(cleardata:direction);

re
en

*inlr = *on;
return;
```



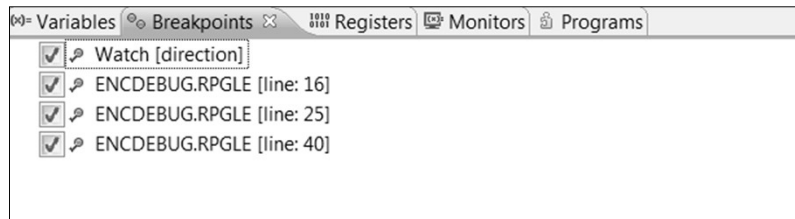
Introducing the MONITORS view

- You decide which variables will appear in this view
- Useful when watching a specific set of fields
- You can right click on a field to switch between character or hexadecimal view

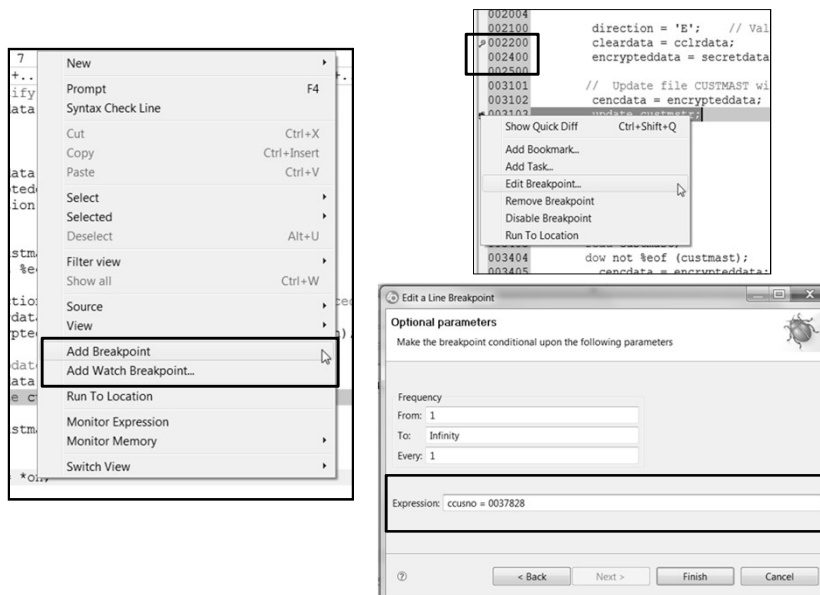


Introducing the BREAKPOINTS view

- Breakpoints can be set at the source level or at runtime
- Breakpoints can be conditional or unconditional
- Can also be disabled so you don't have to delete them
- Watch breakpoints are set at runtime – here we're watching the variable named "direction"



Adding a breakpoint



Adding a watch breakpoint

```

001801
002001 /free
002002 read custmast;
002003 dow not %eof (
002004
002100 direction =
002200 cleardata =
002400 encrypteddat
002500
003101 // Update fi
003102 cencdata = e
003103 update custm
003105
003106 read custmast;
003107 enddo;
003108
003300 *inlr = *on;
003400 return;
003401
003402 begsr *inzsr;
003403 read custmast
003404 dow not %eof
003405 cencdata =
003406 update cust
003407 read custmast
003408 enddo;
003409 setll *zero c
003410 endsr;
003411

```

A context menu is overlaid on the code, with the following items visible:

- New
- Prompt F4
- Syntax Check Line
- Cut Ctrl+X
- Copy Ctrl+Insert
- Paste Ctrl+V
- Select
- Selected
- Deselect Alt+U
- Filter view
- Show all Ctrl+W
- Source
- View
- Edit Breakpoint...
- Remove Breakpoint
- Disable Breakpoint
- Add Watch Breakpoint...
- Run To Location
- Monitor Expression
- Monitor Memory
- Switch View

Watching for changes in the field “direction”

```

IGRPGLE **
21 Column 91 Replace Browse
..... Free-Form.....
ctl-opt bnmdir('UTILITIES' : 'QC2LE') dftactgrp(*no) actgrp('QILE')
option(*srcstmt : *nodebugio) debug(*input);
*
dcl-f custmast disk(*ext) keyed usage(*update);
dcl-pr secretdata char(24);
      *n char(24) value;
      *n char(1) value;
end-pr;
dcl-s cleardata char(24);
dcl-s encrypteddata char(24);
dcl-s direction char(1);

read custmast;
dow not %eof (custmast);

direction = 'E'; // Value of 'E' tells proces
cleardata = cclrdata;
encrypteddata = secretdata(cleardata:direction)

// Update file CUSTMAST with encrypted data
cencdata = encrypteddata;
update custmstr;

```

An "Add a Watch Breakpoint" dialog box is open over the code. The dialog has the following fields:

- Address or expression:
- Number of bytes to watch:
- User label (optional):

Buttons at the bottom: < Back, Next >, Finish, Cancel.

Watching for changes in the field "direction" (cont.)

The screenshot shows a debugger window with a watch breakpoint set on the variable 'direction'. The breakpoint is configured with the following parameters:

- Thread: Every
- Frequency: 1
- From: 1
- To: Infinity
- Every: 1

The code being debugged is as follows:

```

disk(*ext) keyed usage
data char(24);
char(24) value;
char(1) value;

data char(24);
cleardata char(24);
direction char(1);

custmast);
'E'; // Value of 'E'
ccldata;
cleardata = secretdata(cleardata:direction);
// Update file CUSTMAST with encrypted data
encrypteddata;
update custmstr;

read custmast;

```

A "Debug Engine Message" dialog box is displayed, indicating that the variable 'direction' has changed in the program ENCDEBUD, module ENCDEBUD, which is running in job QDFT108D /CGUARINO /051498.

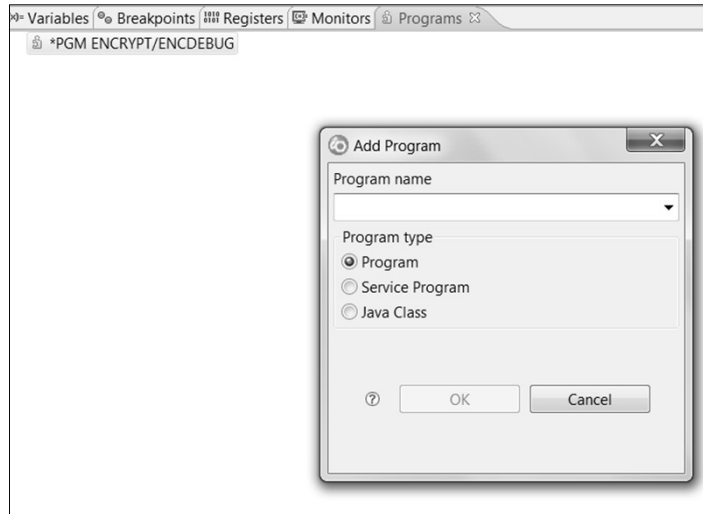
Introducing the OUTLINE view

The screenshot shows the Outline view of a program, displaying a hierarchical tree structure of its components:

- Global Definitions
 - Files
 - custmast : DISK (Externally Described)
 - custmstr
 - 17
 - 18
 - 32
 - Fields
 - CADDR1 : Character (30)
 - CADDR2 : Character (30)
 - CADDR3 : Character (30)
 - CAVGSAL : Packed Decimal (7,0)
 - ccldata : Character (24)
 - CCMP : Packed Decimal (2,0)
 - CCSNAM : Character (30)
 - CCUSNO : Packed Decimal (7,0)
 - CDTLSPM : Date (10)
 - CDTLSSL : Date (10)
 - cenclata : Character (24)
 - cleardata : Character (24)
 - CPGMRUN : Packed Decimal (5,0)
 - CSTRLN : Packed Decimal (5,0)
 - CYTDPFR : Packed Decimal (7,0)
 - CYTDSL : Packed Decimal (7,0)
 - CYTDSL A : Packed Decimal (7,0)
 - CYTDSL B : Packed Decimal (7,0)
 - CYTDSL C : Packed Decimal (7,0)
 - direction : Character (1)
 - encrypteddata : Character (24)
 - Indicators
 - *INLR
 - Prototypes
 - secretdata : Character (24) EXTPROC ('SECRETDATA')
- Main Procedure

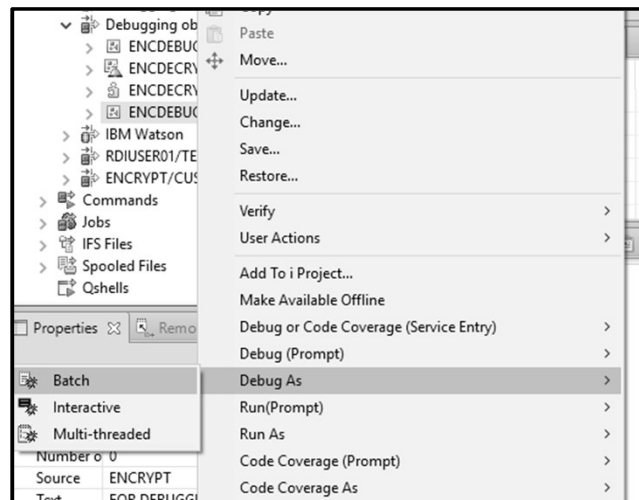
Introducing the PROGRAMS view

- Functionally similar to pressing F14 from DSPMODSRC screen



Submitting without any parameters

- RDi will use the job description associated with the log in profile

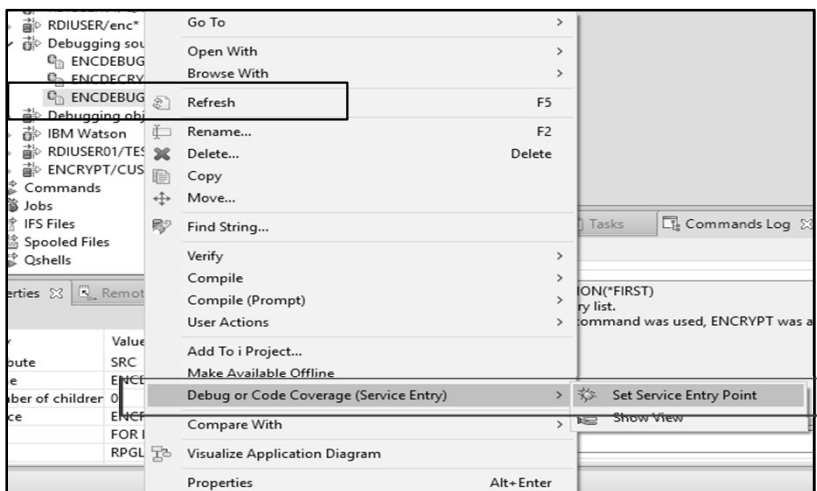


What We'll Cover ...

- Let's Get Started
- Preparing to Submit
- The Debugger Comes to Life
- **Setting a Service Entry Point**
- RDi Debugging Hacks
- Wrap-up

Setting a Service Entry Point

- Right click on any source member



Setting a Service Entry Point from a source member (cont.)

- You will have an opportunity to change any of these values
- This is a HUGE improvement over service jobs and STRSRVJOB

Set Service Entry Point

Connection: IBM i RD1 9.6 [New...]

Library: ENCRYPT [Browse...]

Program: ENCDEBUGM [Browse...]

Service Program: [Browse...]

Module: *ALL [Browse...]

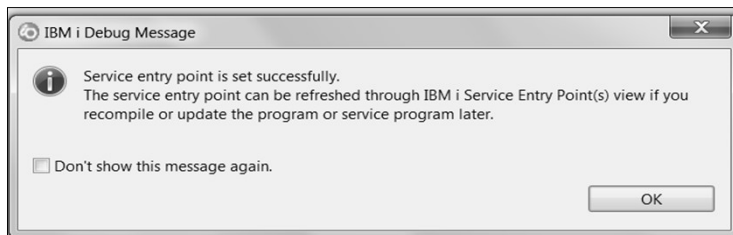
Procedure: *ALL [Browse...]

User ID: CGUARINO

[OK] [Cancel]

Setting a Service Entry Point from a source member (cont.)

- Once the SEP has been set you will receive this confirmation
- You will see your parameters in the SEP view in the RSE



Library	Program	Program Type	Module	Procedure	User ID	Connection	Enabled
ENCRYPT	ENCDEBUGM	*PGM	*ALL	*ALL	CGUARINO	IBM i RD1 9.6	Yes

What We'll Cover ...

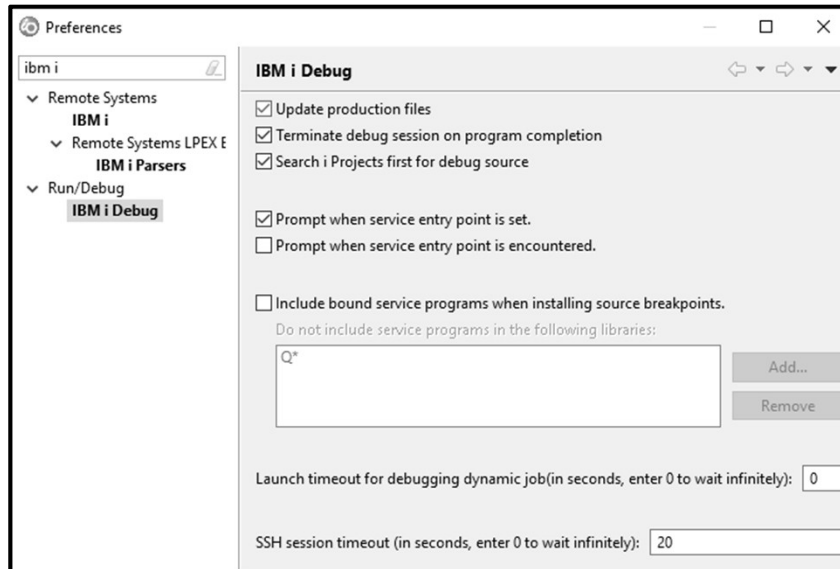
- Let's Get Started
- Preparing to Submit
- The Debugger Comes to Life
- Setting a Service Entry Point
- **RDi Debugging Hacks**
- Wrap-up

Problem: Update production files set to *NO* when using a Service Entry Point

- Received error CPF4203 when starting to debug program

The screenshot displays the IBM Rational Developer for System z interface. At the top, there is a 'Variables' window showing a list of variables: *IN, CADDR1, CADDR2, CADDR3, CAVSSAL (value: 0000000), and CCMP (value: 00). Below this is the main editor window showing the source code for 'ENCDEBGM.RPGLE'. The code includes declarations for 'extpgm' and 'secretdata', and a procedure interface for 'encdebugm'. A 'Debug Engine Message' dialog box is overlaid on the code, displaying the error: 'DBGP0003E Program received unmonitored exception RNK1216: Error message CPF4203 appeared during OPEN for file CUSTMAST.' The dialog has an 'OK' button and a 'Hide this message' checkbox.

Solution: Select Update production files set in IBM i Debug



Problem: Debug view keeps “popping up” every time you step through the code

- Reason: Debug view must be open to retain communication with the debug servicer

The screenshot shows the 'Debug' view with a tree view of threads. The top thread is 'Thread:1 (stopped)' with sub-threads: 'ENCDEBUGM: ENCDEBUGM: ENCDEBUGM', '_QRNP_PEP_ENCDEBUGM: ENCDEBUGM: ENCDEBUGM', 'QCAPCMD: QCAPCMD: QCAPCMD', 'ProcessCommand: QRSEEXEC: EVFCEXEC', 'main: QRSEEXEC: EVFCEXEC', '_C_pep: QRSEEXEC: EVFCEXEC', and 'QCMD: QCMD: QCMD'. Below the tree is a code editor window titled 'ENCDEBUGM.RPGLE' showing the following code:

```
Line 27      Column 1      Replace      Browse
002000      end-pi;
002100
002200      dcl-s      cleardata      char(24);
002300      dcl-s      encrypteddata  char(24);
002400      dcl-s      direction      char(1);
002500
002600      // If called without any parameters, default to (E)ncrypt
002700      if %parms > *zero;
002800          direction = directionin;
002900      else;
003000          direction = 'E';
003100      endif;
003200
003300
003400      read custmast;
003500      dow not %eof (custmast);
003600
```

Solution: Move the Debug view to an out of sight area

- Better yet, move it to a second monitor!

The screenshot shows the IBM Debug view for the program ENCDEBUGM.RPGLE. The view is positioned in the top-left corner of the screen, partially overlapping the main application window. The source code is displayed in a table format with columns for Line, Column, and the code text.

Line	Column	Code
33	671.....2.....3.....4.....5.....6.....7..
002000		end-pi;
002100		
002200		dcl-s cleardata char(24);
002300		dcl-s encrypteddata char(24);
002400		dcl-s direction char(1);

Problem: Variables view doesn't show very large values

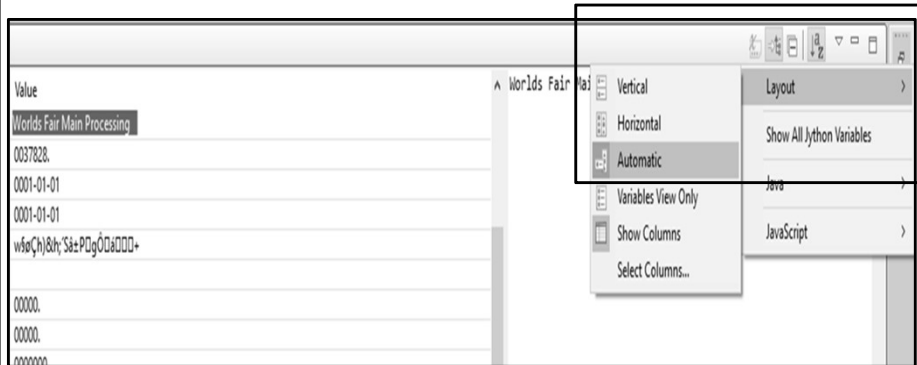
- Debug engine has a limit of 1024 characters

The screenshot shows the IBM Variables view for the program ENCDEBUGM.RPGLE. The view is positioned in the top-left corner of the screen, partially overlapping the main application window. The variables and their values are listed in a table format.

Name	Value
CCSNAM	Worlds Fair Main Processing
CCUSNO	0037828.
CDTLSPM	0001-01-01
CDTLSSL	0001-01-01
CENCDATA	wfC)h;S±P#gÔs+++
CLEARDATA	
CPGMRUN	00000.
CSTRLEN	00000.
CYDPRF	0000000.
CYTDSL	0000000.
CYTDSL	0000000.
CYTDSL	0000000.
CYTDSL	0000000.
CYTDSL	0000000.
CYTDSL	0000000.
DIRECTION	E
DIRECTIONIN	*
ENCRYPTEDDATA	
_QRNL_PRMCOPY_DIRECTIONIN	SPP:*NULL

Solution 1: Change the layout of the debug view

- The upside down triangle reveals the drop down menu



Solution 2: Official IBM response

IBM Support

Search support or find a product

SE62940: In RDi 9.1.1.1, Variables or structures larger than 1024 bytes cannot be displayed when debugging programs on the IBM i

APAR status

Closed as documentation error.

Error description

When using Rational Developer for i (RDi) 9.1.1.1 to debug an application on the IBM i it is not possible to view variables or structures larger than 1024 bytes in size. The debug engine on the IBM i currently has a limitation and cannot send more than 1024 bytes to the debugger interface to display in the Variables or Monitors view.

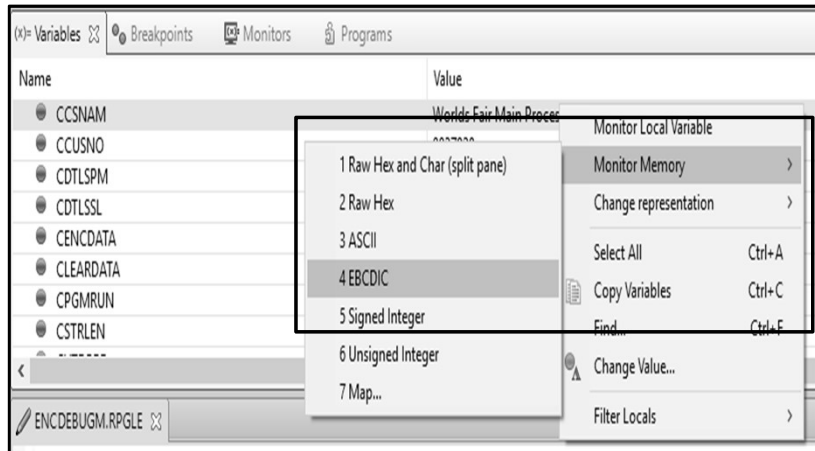
Local fix

You can view the string from the memory view when you monitor the expression in memory using the Raw Hex and Char (split pane) rendering.

<http://www-01.ibm.com/support/docview.wss?uid=swg1SE62940>

Solution 2: Monitor Memory of the variable

- Choose the EBCDIC option



Solution 2: Monitor Memory of the variable

- Can then display up to first 4096 bytes of the variable

The screenshot shows the 'Monitors' window with a memory dump for the 'CCSNAM' variable. The dump is displayed in a table with columns for address and character groups (0-3, 4-7, 8-B, C-F). The text 'World ds F air Main' is visible in the first row.

Address	0 - 3	4 - 7	8 - B	C - F
E9990351310026A0	World	ds F	air	Main
E9990351310026B0	Pro	cess	ing	..
E9990351310026C0	wšdÇ	h)&h	; 'S	âtP
E9990351310026D0	gÖá	.+
E9990351310026E0				
E9990351310026F0		
E999035131002700				
E999035131002710		

Solution 3: Use STRDBG (Wait, what?)

```
Display Module Source
Program: ENCDEBUGM Library: ENCRYPT Module: ENCDEBUGM
31     endif;
32
33
34     read custmast;
35     dow not %eof (custmast);
36
37
38     if direction = 'E';
39         encrypteddata = secretdata(cclrdata:direction);
40         cencdata = encrypteddata;
41     else;
42         cleardata = secretdata(cencdata:direction);
43         cclrdata = cleardata;
44     endif;
45
Debug . . . eval ccsnam :c 1000_ More...
```

What We'll Cover ...

- Let's Get Started
- Preparing to Submit
- The Debugger Comes to Life
- Setting a Service Entry Point
- RDi Debugging Hacks
- Wrap-up



RDi Debugging
All Lanes
Open Ahead

From the On Ramp to the Cruising Lane
Charles Guarino
THANK YOU !!!!