The Art of Debugging: From STRDBG to RDi

Charles Guarino

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

In This Session …

For years we believed that STRDBG had been adequate for everyday debugging situations. With the introduction of WDSC/RDP/RDi we have been given the ability to extend our productivity in a feature-rich graphical environment.

In this session we will review every aspect of this new environment and explore how the days of green screen debugging have become a technology of the past.

What We'll Cover …

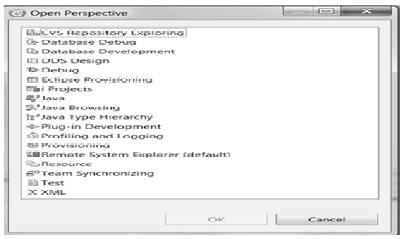
- Perspectives
- Review program we will debug
- The Debug Server
- Service Entry Points
- Calling a program from within RDi and debug configurations
- Debugging views
- Debugging another user's program
- Wrap-up

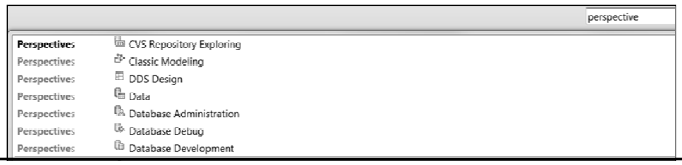
Perspectives

- There are many available in RDi
  - This session focuses on the debugging perspective
  - To see all available perspectives click on

    Window>Open Perspective>Other

    - Or – Typing "Perspective" in Quick Access

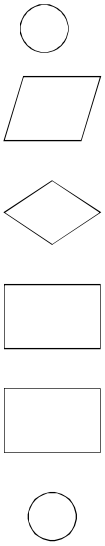




What We'll Cover …

- Perspectives
- Review program we will debug
- The Debug Server
- Service Entry Points
- Calling a program from within RDi and debug configurations
- Debugging views
- Debugging another user's program
- Wrap-up

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, return ciphertext

Update CUSTMAST with encrypted data

Read more records from file CUSTMAST

Program we will be debugging (cont.)

```
*ENCDEBUG.RPGLE
 Line 31      Column 72    Replace  1 change
      .....  Free-Form++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
000100        ctl-opt bnddir('UTILITIES' : 'QC2LE') dftactgrp(*no) actgrp('QILE')
000101
000200            option(*srcstmt : *nodebugio) debug(*input);
000300       *
000301        dcl-f  custmast disk(*ext) keyed usage(*update);
000302
000303        dcl-pr secretdata   char(24);
000304                 *n         char(24)  value;
000305                 *n         char(1)   value;
000308        end-pr;
000309
000311        dcl-s   cleardata      char(24);
000312        dcl-s   encrypteddata  char(24);
000313        dcl-s   direction      char(1);
000317
001400
001600        read custmast;
001700        dow not %eof (custmast);
001800
001900          direction = 'E';    // Value of 'E' tells procedure to ENCRYPT
002000          cleardata = cclrdata;
002100          encrypteddata = secretdata(cleardata:direction);
002200
002300         //  Update file CUSTMAST with encrypted data
002400          cencdata = encrypteddata;
002500          update custmstr;
002600
002700        read custmast;
002800        enddo;
002900
003000        *inlr = *on;
```

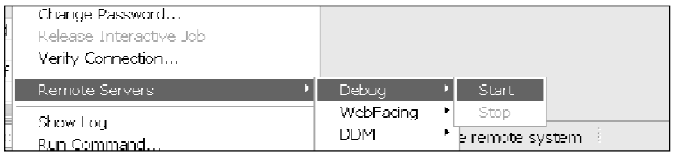
What We'll Cover …

- Perspectives
- Review program we will debug
- The Debug Server
- Service Entry Points
- Calling a program from within RDi and debug configurations
- Debugging views
- Debugging another user's program
- Wrap-up

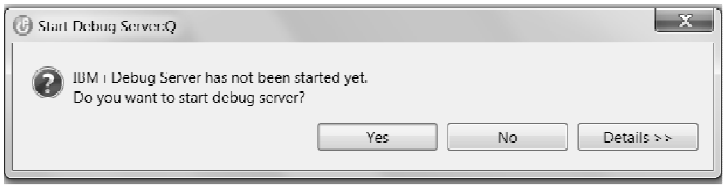
The Debug Server

- Listens on the IBM i for debugging instructions from RDi
- It needs to be active before any debugging can occur
  - You will receive a warning message if you try to debug a program and the server is not yet active.
    - Don't panic! You can start it immediately directly from RDi.
  - Once the debug server is started it will work for everyone
  - There is NOT one server PER USER – only one per system which will service every developer's RDP debugging requests
    - I recommend putting command STRDBGSVR in your startup program

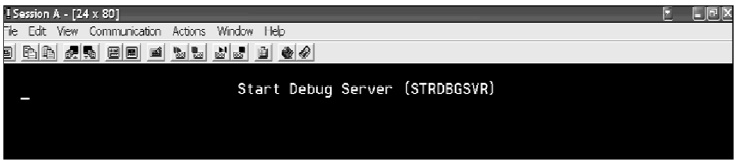
Starting the Debug Server (3 different ways!)



- OR -



- OR -



The Debug Server in action

- Runs in subsystem QUSRWRK as jobs and programs QB5BROUTER and QB5SERVER
- Job will use the user ID that started the server
- The debug server will remain active until it is explicitly ended
- There will be an additional job for each program being debugged, serviced by program QRSEEXEC

Opt	Subsystem/Job	User	Type	CPU %	Function	Status
__	QUSRWRK	QSYS	SBS	.0		DEQW
__	QB5ROUTER	CGUARINO	BCH	.0	PGM-QB5ROUTER	SELW
__	QB5SERVER	CGUARINO	BCI	.0	PGM-QB5SERVER	SELW

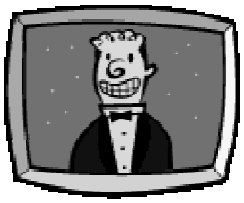
Opt	Subsystem/Job	User	Type	CPU %	Function	Status
__	QBATCH	QSYS	SBS	.0		DEQW
__	QDFTJOBD	CGUARINO	BCH	.0	PGM-QRSEEXEC	EVTW

What We'll Cover …

- Perspectives
- Review program we will debug
- The Debug Server
- Service Entry Points
- Calling a program from within RDi and debug configurations
- Debugging views
- Debugging another user's program
- Wrap-up

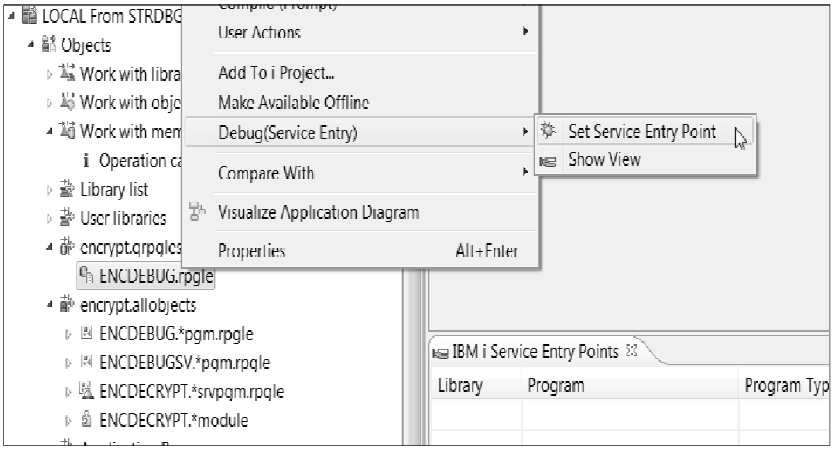
Starting the Debugger

- There are three methods to prepare and launch the debugger:
  - Method 1: A program can be launched directly from RDi
    - With or without parameter prompting
  - Method 2: Setting a Service Entry Point
    - When the program is run anywhere using the specified parameters the debugger will be launched
  - Method 3: Debugging an active job
    - Can intercept an active job to identify and resolve issues



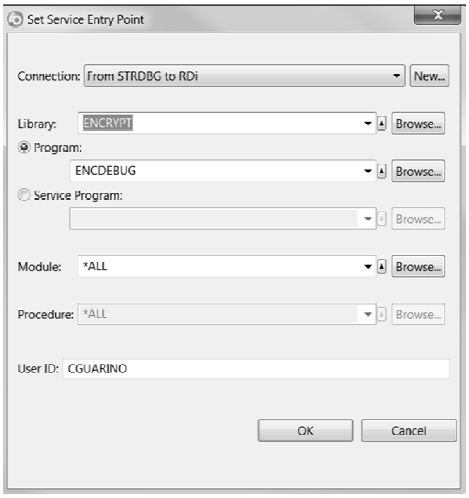
Setting a Service Entry Point from a source member

- **Right click on any source member**



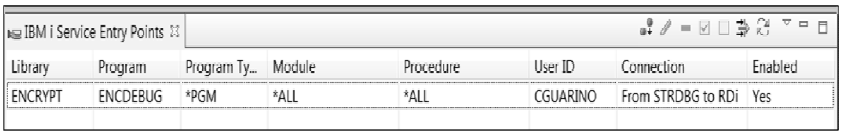
Setting a Service Entry Point from a source member (cont.)

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



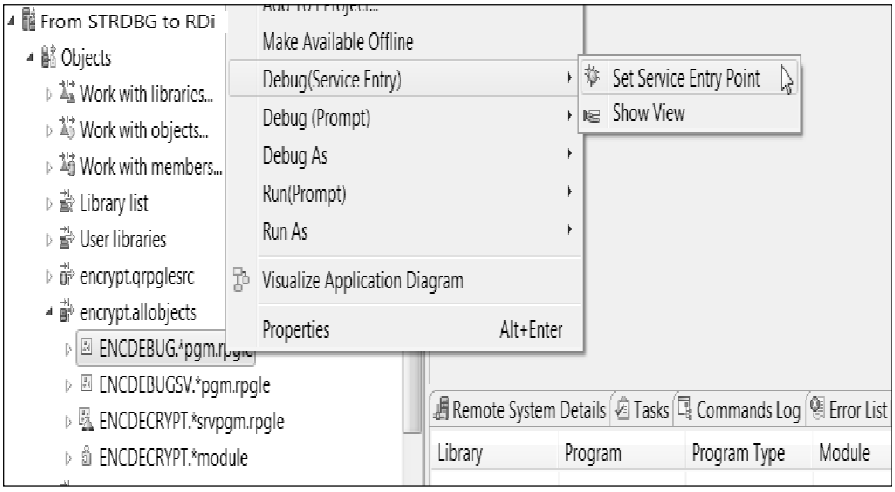
Setting a Service Entry Point from a source member (cont.)

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

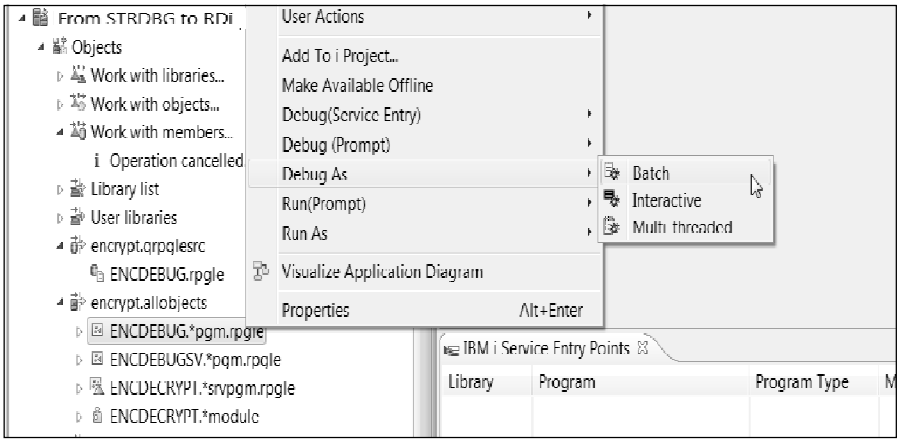


Library	Program	Program Ty...	Module	Procedure	User ID	Connection	Enabled
ENCRYPT	ENCDEBUG	*PGM	*ALL	*ALL	CGUARINO	From STRDBG to RDi	Yes

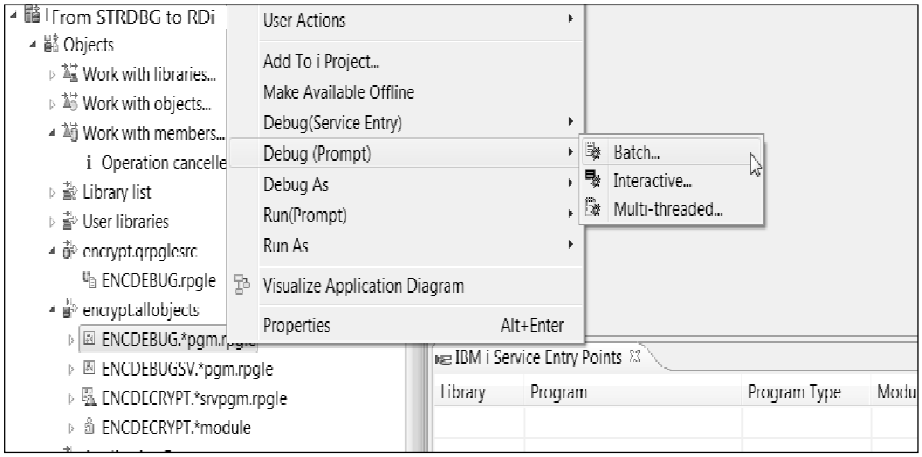
Service Entry Point being set from a program object



Method 2: Calling and debugging a program directly from RDi



Calling and debugging with a prompt



Debug configurations

Edit Configuration

Edit configuration and launch.

Name: My program (Batch) CAN BE NAMED AND SAVED

What To Debug | How To Start | Source | Common

Connection: From STRDBG to RDi | New...

Programs or service programs to be debugged

Library	Program	Type
ENCRYPT	ENCDEBUG	*PGM

Add... Edit... Remove

SPECIFY WHAT AND HOW TO RUN
CAN SPECIFY PROGRAM PARAMETERS

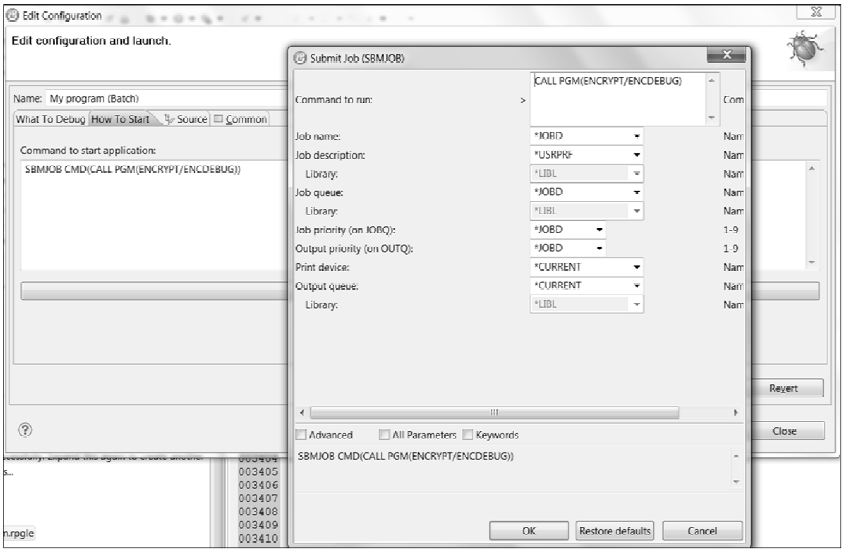
Step into
Terminate debug session on program completion
Update production files

Apply Revert

Debug Close

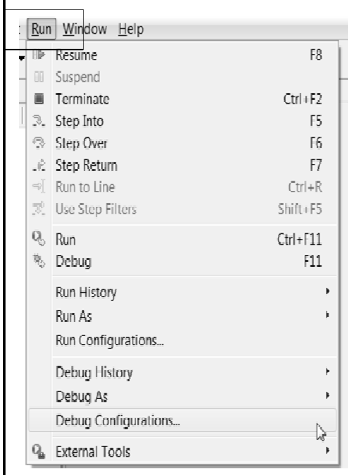
Debug configurations (cont.)

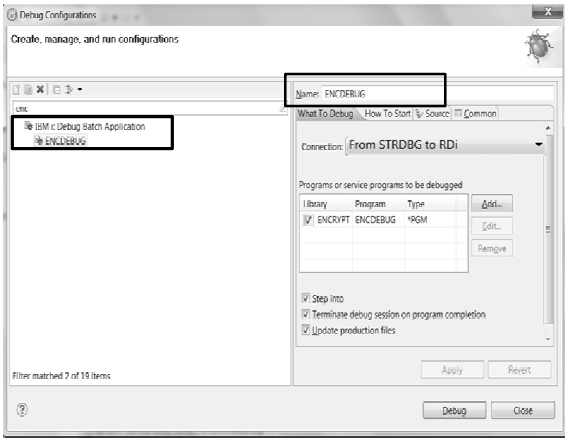
- **Click on "How to Start" for additional parameters**



Launching an existing configuration

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



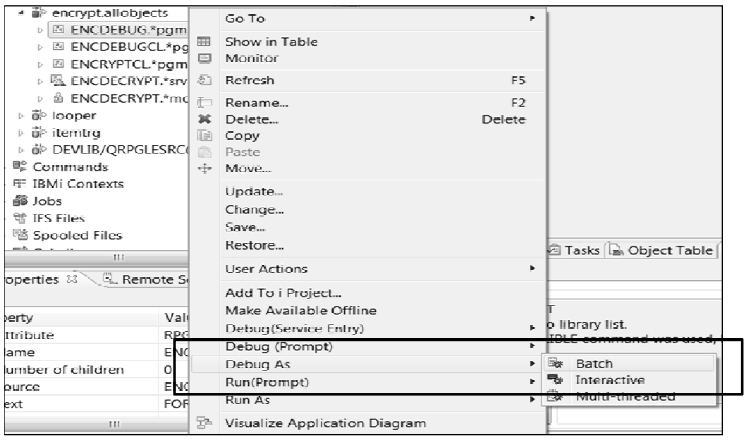


What We'll Cover …

- Perspectives
- Review program we will debug
- The Debug Server
- Service Entry Points
- Calling a program from within RDi and debug configurations
- Debugging views
- Debugging another user's program
- Wrap-up

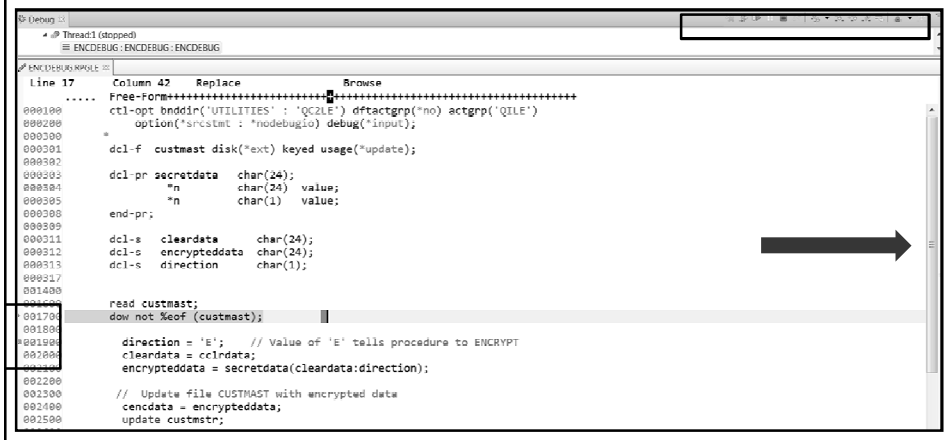
Submitting and debugging a job directly from RDi

- Debug as submits with your current session's settings
  - This includes library list, updprod settings, etc.
- Debug (prompt) brings up a debug configuration



Introducing the DEBUG perspective

- "Wakes up" automatically when a program launched in debug mode or an active service entry point is encountered
- Green line is the current line of execution
- Boxes shows shortcuts, breakpoints and current line of execution pointer

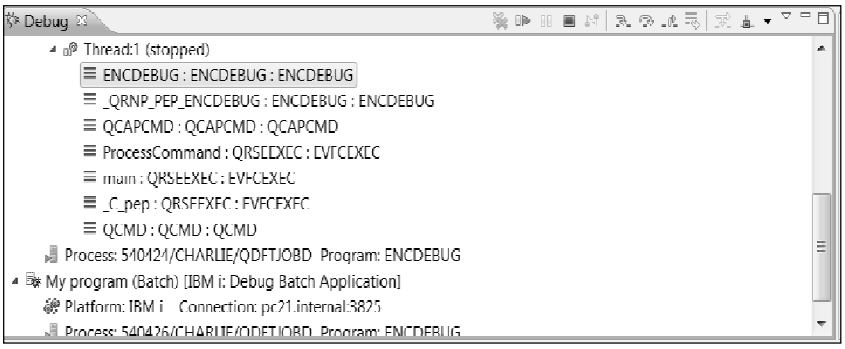


What We'll Cover …

- Perspectives
- Review program we will debug
- The Debug Server
- Service Entry Points
- Calling a program from within RDi and debug configurations
- Debugging views
- Debugging another user's program
- Wrap-up

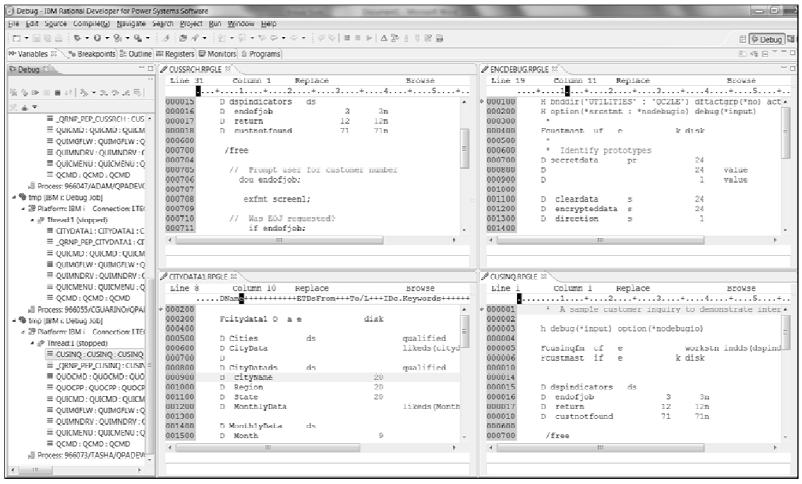
Introducing the DEBUG view

- 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



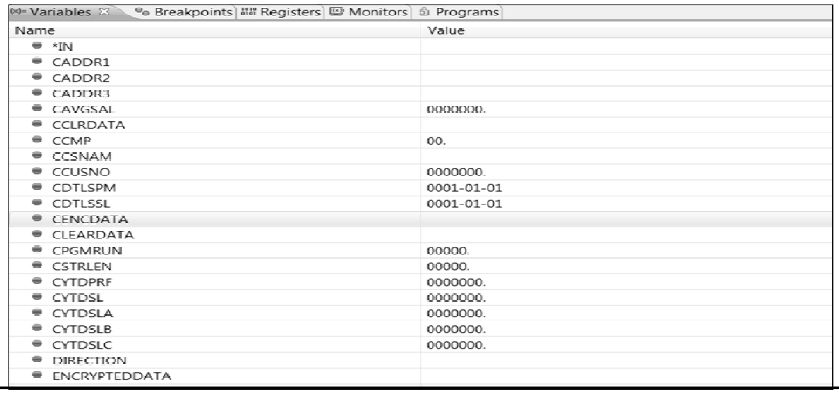
The DEBUG view

- 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



Variables | Breakpoints | Registers | Monitors | Programs

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.
CYTDSLC	0000000.
DIRECTION	
ENCRYPTEDDATA	

Green screen equivalent to variables view!

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

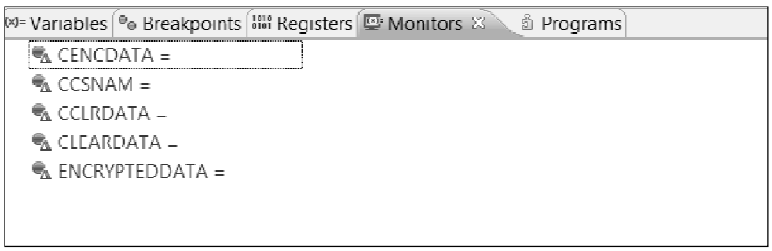
```
                              Evaluate Expression

Previous debug expressions

  *IN(98) = '0'
  *IN(99) = '0'
  CADDR1 = '10 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 = 'SÎ▒▒B▒ÛWg8¦ûP ▒▒gÔ▒ä▒▒+'
  CLEARDATA = '1234567890              '
                                                                    More...
Debug . . .   eval %localvars_
F3=Exit   F9=Retrieve   F12=Cancel   F18=Repeat find   F19=Left   F20=Right
F21=Command entry       F23=Display output
```

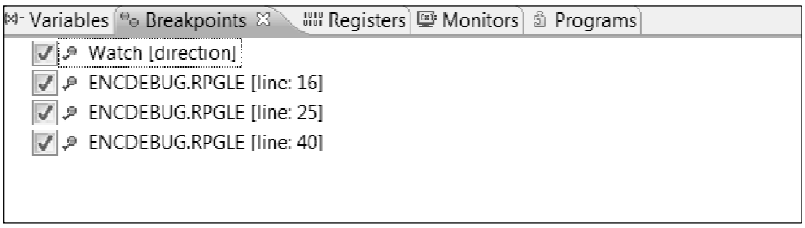
Introducing the MONITORS view

- **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**

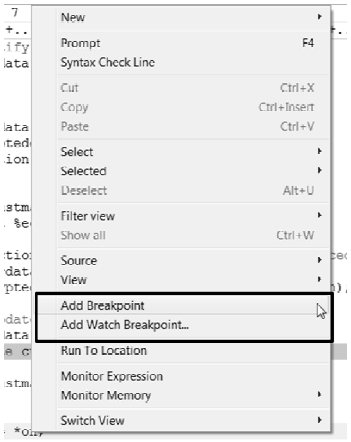


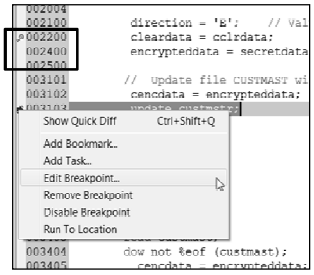
Introducing the BREAKPOINTS view

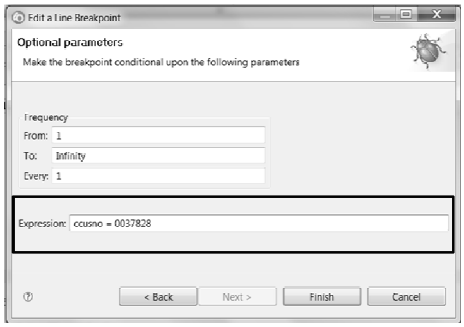
- Breakpoints can be set at the source level or at runtime
- Breakpoints can be conditional or unconditional
- Can also be disabled so you don't have to delete them
- Watch breakpoints are set at runtime – here we're watching the variable named "direction"



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 - c
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 custm
003407        read custmast
003408        enddo;
003409        setll *zero c
003410        endsr;
003411
```

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"

```
G.RPGLE
21      Column 91    Replace              Browse
.....  Free-Form++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
       ctl-opt bnddir('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 proce
         cleardata = cclrdata;
         encrypteddata = secretdata(cleardata:direction)

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

Add a Watch Breakpoint

Required information

Sets a breakpoint to stop execution when data changes at a specific address

Address or expression: direction

Number of bytes to watch:

0

User label (optional):

< Back | Next > | Finish | Cancel

Watching for changes in the field “direction” (cont.)

```
t disk(*ext) keyed usage

ata    char(24);
       char(24)  value;
       char(1)   value;

ata       char(24);
teddata   char(24);
ion       char(1);

custmast);

'E';    // Value of 'E'
cclrdata;
= secretdata(cleardata

le CUSTMAST with encrypt
ncrypteddata;
str;
```

Add a Watch Breakpoint

Optional parameters

Make the breakpoint conditional upon the following parameters

Thread: Every

Frequency

From: 1

To: Infinity

Every: 1

< Back | Next > | Finish | Cancel

```
dow not %eof (custmast);

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

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

read custmast;
```

Debug Engine Message

DBGB0009I Variable direction has changed in program ENCDEBUG, module ENCDEBUG, which is running in job QDFTJOBD /CGUARINO /051498.

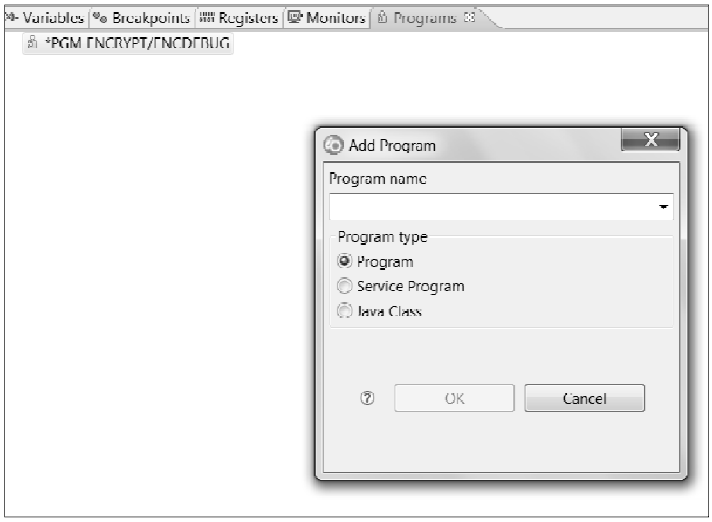
OK

Introducing the OUTLINE view

- Outline
  - 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)
      - cclrdata : Character (24)
      - CCMP : Packed Decimal (2,0)
      - CCSNAM : Character (30)
      - CCUSNO : Packed Decimal (7,0)
      - CDTLSPM : Date (10)
      - CDTLSSL : Date (10)
      - cencdata : Character (24)
      - cleardata : Character (24)
      - CPGMRUN : Packed Decimal (5,0)
      - CSTRLEN : Packed Decimal (5,0)
      - CYTDPRF : Packed Decimal (7,0)
      - CYTDSL : Packed Decimal (7,0)
      - CYTDSLA : Packed Decimal (7,0)
      - CYTDSLB : Packed Decimal (7,0)
      - CYTDSLC : Packed Decimal (7,0)
      - direction : Character (1)
      - encrypteddata : Character (24)
    - Indicators
      - *INLR
    - Prototypes
      - secretdata : Character (24) EXTPROC ('SECRETDATA')
  - Main Procedure

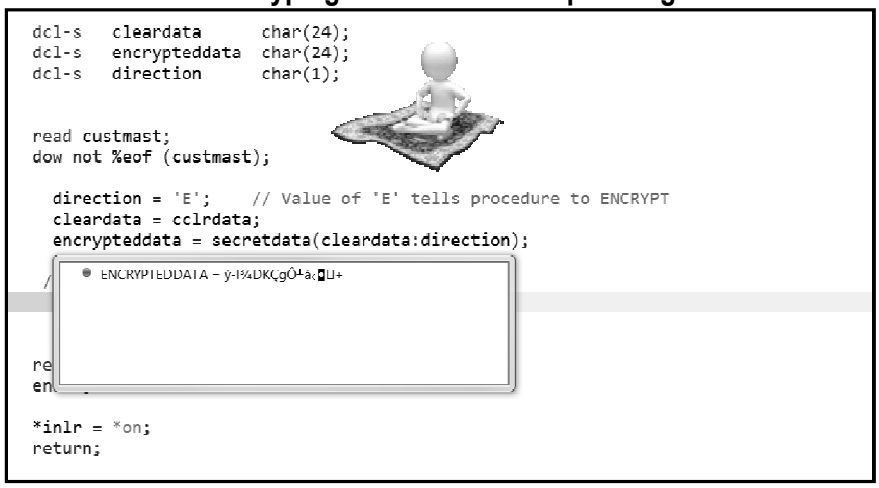
Introducing the PROGRAMS view

- **Functionally similar to pressing F14 from DSPMODSRC screen**



Field hovering

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

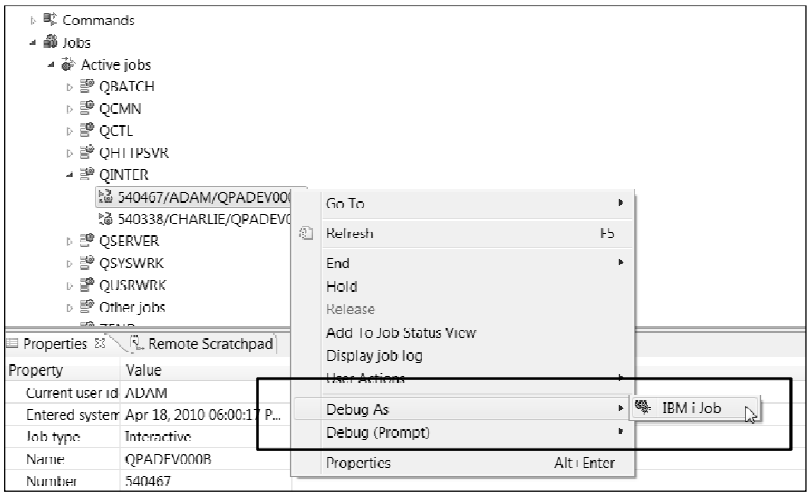


What We'll Cover …

- Perspectives
- Review program we will debug
- The Debug Server
- Service Entry Points
- Calling a program from within RDi and debug configurations
- Debugging views
- Debugging another user's program
- Wrap-up

Debugging Another User's Job

- Locate the active job, right click on it and select "Debug As > IBM i job"



What We'll Cover …

- Perspectives
- Review program we will debug
- The Debug Server
- Service Entry Points
- Calling a program from within RDi and debug configurations
- Debugging views
- Debugging another user's program
- Wrap-up

The Art of Debugging: From STRDBG to RDi

Charles Guarino

THANK YOU !!!