

# Application Development for z/OS

## *Not your Father's Green Screen*

Tim Hahn  
IBM

8 August 2011  
Session 9767

# Abstract



Ask most people how they write and maintain applications on z/OS and you hear "oh, you use this thing called a green screen" followed by a chuckle.

In reality, application development for zEnterprise applications has been transformed over the past several years to the point where application developers enjoy the same or better features from integrated development environments as programmers who work on other platforms.

Advances in remote system communication and interaction, syntax-highlighting, parsing, and code understanding for Assembler, PL/I, C/C++, and COBOL source code, as well as programming assists such as code snippets and templates are all available to application programmers. Interactive debug of applications, written in multiple programming languages and running in various runtime environments is also possible and can greatly boost programmer productivity.

Come and learn about how these features can enable application developers who are new to the mainframe to interact with, update, and efficiently enhance mainframe applications.

# Agenda

- Application Development is Hard
- Tools to the rescue!
- Using tools is Hard
- Continual Discovery
- Reprise: Application Development is Hard

# Agenda

- Application Development is Hard
- Tools to the rescue!
- Using tools is Hard
- Continual Discovery
- Reprise: Application Development is Hard

# Traditional Application Development for z/OS

- study compiler listings (green bar printout) or use ISPF



```

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT      DDS0001.TEST.COBO(LHOSPEDIT) - 27.26      Columns 00001 00072
***** ***** Top of Data *****
000001      *****
000002      IDENTIFICATION DIVISION.
000003      PROGRAM-ID.  HOSPEDIT.
000004      AUTHOR.  JON SAYLES.
000005      INSTALLATION. COBOL DEVELOPMENT CENTER.
000006      DATE-WRITTEN. 01/01/08.
000007      DATE-COMPILED. 01/01/08.
000008      SECURITY. NON-CONFIDENTIAL.
000009
000010      *****
000011      *  A new comment ...
000012      *  Jon's new comment
000013      ENVIRONMENT DIVISION.
000014      CONFIGURATION SECTION.
000015      SOURCE-COMPUTER. IBM-390.
000016      OBJECT-COMPUTER. IBM-390.
000017      INPUT-OUTPUT SECTION.
Command ==>
F1=Help    F2=Split    F3=Exit    F5=Rfind    F6=Rchange  F7=Up
F8=Down    F9=Swap     F10=Left   F11=Right   F12=Cancel
  
```

# Multiple Edit windows are possible - but limited

- ISPF split-screen mode allows this ... but it is far from sufficient for complex, multi-module application programming problems



```

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT      DDS0001.TEST.COBOL(HOSPEDIT) - 27.26          Columns 00001 00072
***** ***** Top of Data *****
000001 *****
000002 IDENTIFICATION DIVISION.
000003 PROGRAM-ID. HOSPEDIT.
000004 AUTHOR. JON SAYLES.
Command ==> Scroll ==> PAGE
F1=Help    F2=Split    F3=Exit    F5=Rfind    F6=Rchange  F7=Up
F8=Down    F9=Swap     F10=Left   F11=Right   F12=Cancel

File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT      DDS0001.TEST.COBOL(HOSPCALC) - 06.05          Columns 00001 00072
***** ***** Top of Data *****
000001 IDENTIFICATION DIVISION.
000002 PROGRAM-ID. HOSPCALC.
000003 AUTHOR. JON SAYLES.
000004 ENVIRONMENT DIVISION.
Command ==> Scroll ==> PAGE
F1=Help    F2=Split    F3=Exit    F5=Rfind    F6=Rchange  F7=Up
F8=Down    F9=Swap     F10=Left   F11=Right   F12=Cancel
  
```

# And it's not just the basic tools ...

- Existing systems have grown by evolution over many years
- Many documented (and un-documented) dependencies
- Sheer volume of applications
  - thousands of batch jobs
  - thousands of programs
  - billions of lines of COBOL code run daily, not to mention on other schedules (e.g. end of quarter, end of year).
- Online transaction processing also factors in
- Without application analysis tools, teams have difficulty understanding even where to start

# Agenda

- Application Development is Hard
- **Tools to the rescue!**
- Using tools is Hard
- Continual Discovery
- Reprise: Application Development is Hard



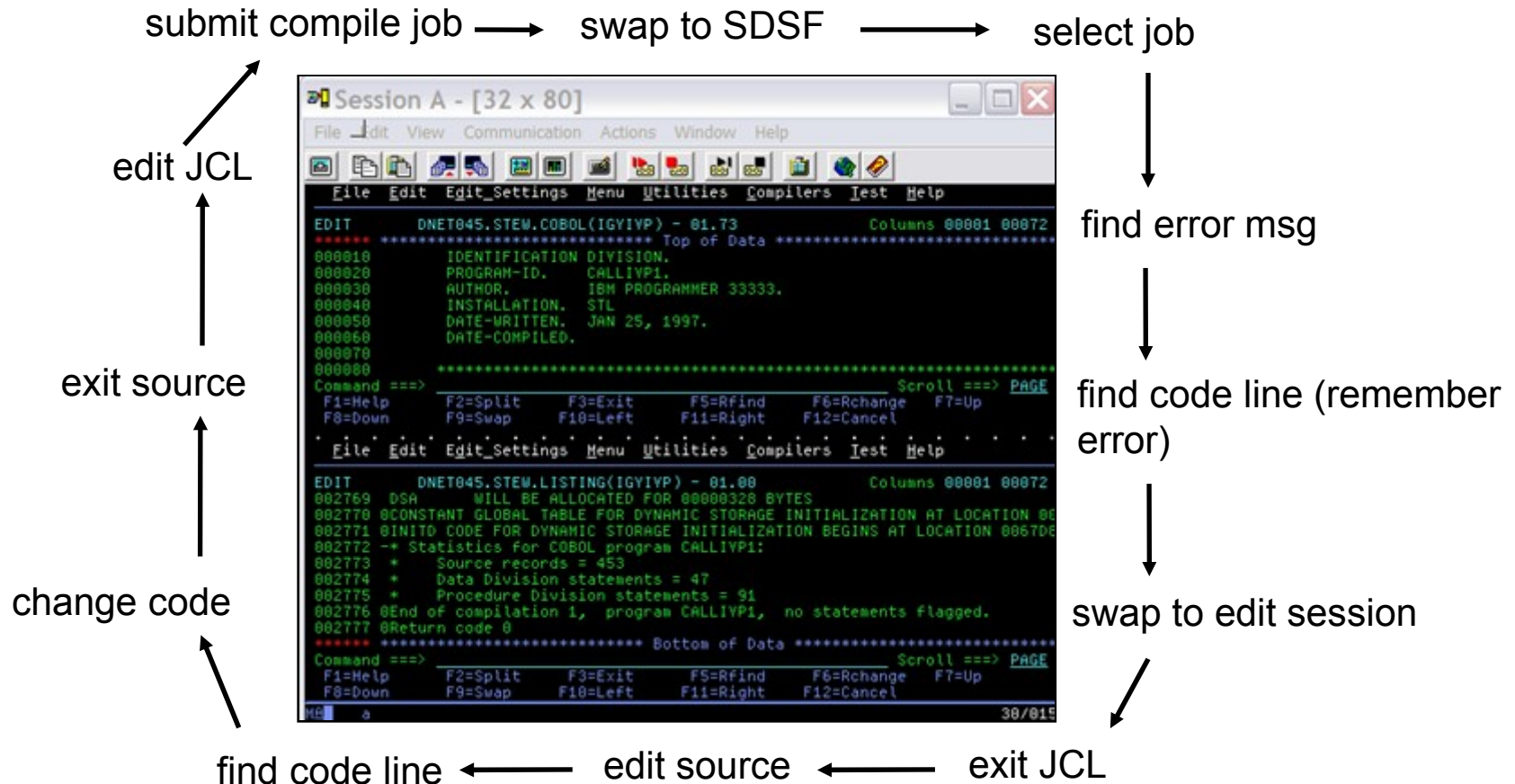
# Rational Developer for System z and zEnterprise



- Practitioner tools for application development and enhancement
  - ▶ Java
  - ▶ COBOL
  - ▶ PL/I
  - ▶ C/C++
  - ▶ Assembler
  - ▶ JCL
- Supporting tasks of
  - ▶ Remote access to files and jobs
  - ▶ Analyze, Understand, Edit, Build, and Unit Test of applications
  - ▶ Remote interactive debug of applications running in multiple environments
  - ▶ Integration with SCMs including Team Concert and Endevor
- Support for several source code location models
  - ▶ “remote” source code (source code held on development system)
  - ▶ “local” source code (source code held on system where IDE is running)

# ISPF-based development

- Multiple screens/sessions and multiple disparate tools
- 20 x 80 characters of content



# IDE-based development

- Common development environment for COBOL, PL/I, C/C++, and Java
- Simplified development with more information at your fingertips

Open and edit multiple source and JCL members simultaneously

Edit Source

Syntax Check

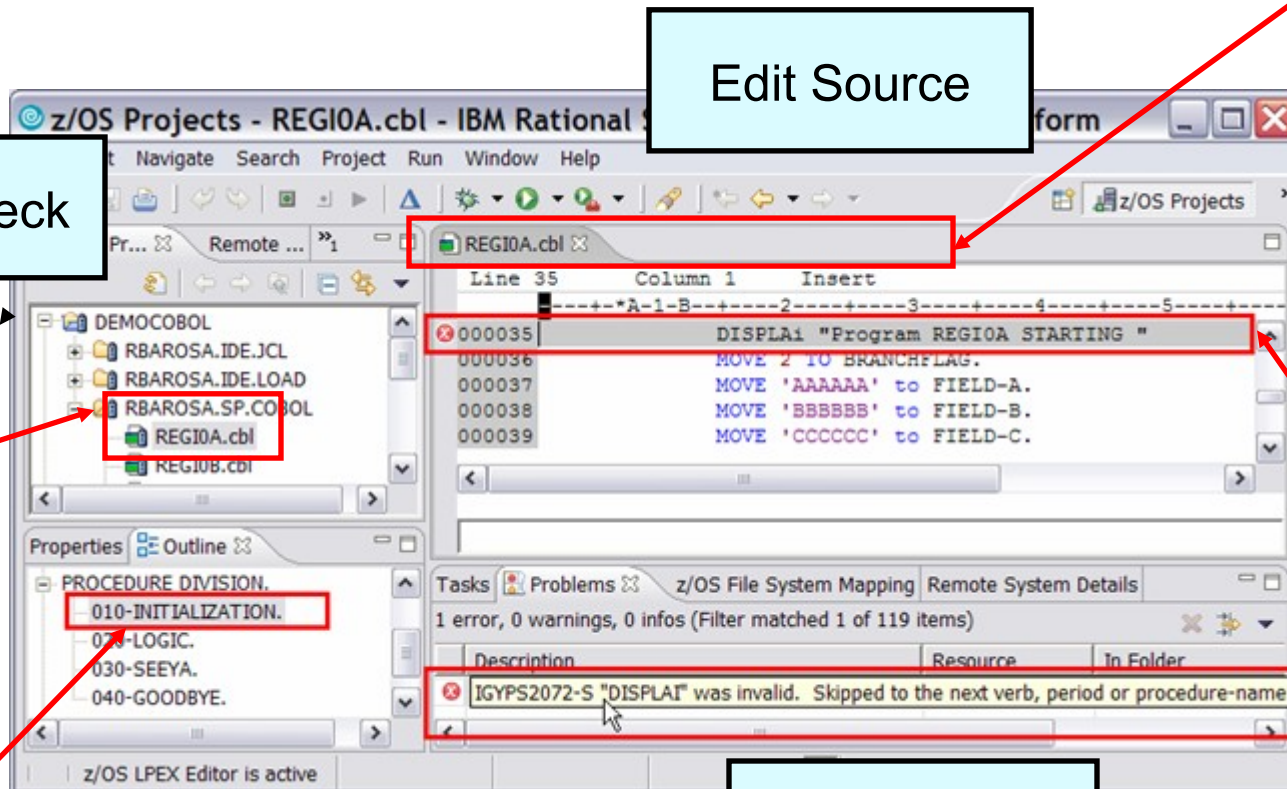
Submit jobs, access job output, or open source members with a single click

Statement in error indicated in source

Outline view presents COBOL structure

Double-Click on the Error

Error list in Problems view



# IDE-based Development



The screenshot displays the IBM Rational Developer for System z IDE interface. The main editor window shows a COBOL program named COBDATE.cbl. The code is as follows:

```
Line 16      Column 1      Insert
-----*A-1-B-----2-----3-----4-----5-----6-----
WORKING-STORAGE SECTION.
01  ChrDate.
   05  ChrDate-Length      pic s9(4) comp value 10.
   05  ChrDate-String      pic x(10).
01  PicStr.
   05  PicStr-Length       pic s9(4) comp.
   05  PicStr-String       pic x(80).
77  Lilian                 pic s9(9) comp.
77  Formatted-Date         pic x(80).

PROCEDURE DIVISION.
*
MAIN-LOGIC.
  DISPLAY "Starting COBDATE".
  DISPLAY "-----"
  ACCEPT ChrDate-String from DATE.
```

The left pane shows the project structure for 'z/OS Projects', including folders like 'BeerBottles', 'CompiledCodeSamples4B', and 'ParallelC-Sample'. The bottom-left pane shows the 'Properties' view for the 'PROGRAM: COBDATE', detailing sections like IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, CONFIGURATION SECTION, DATA DIVISION, and PROCEDURE DIVISION.

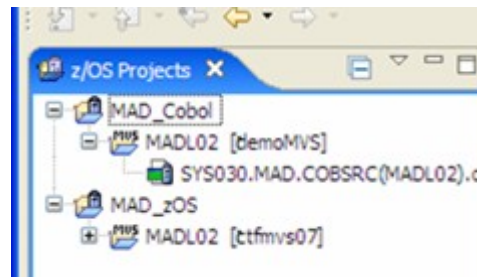
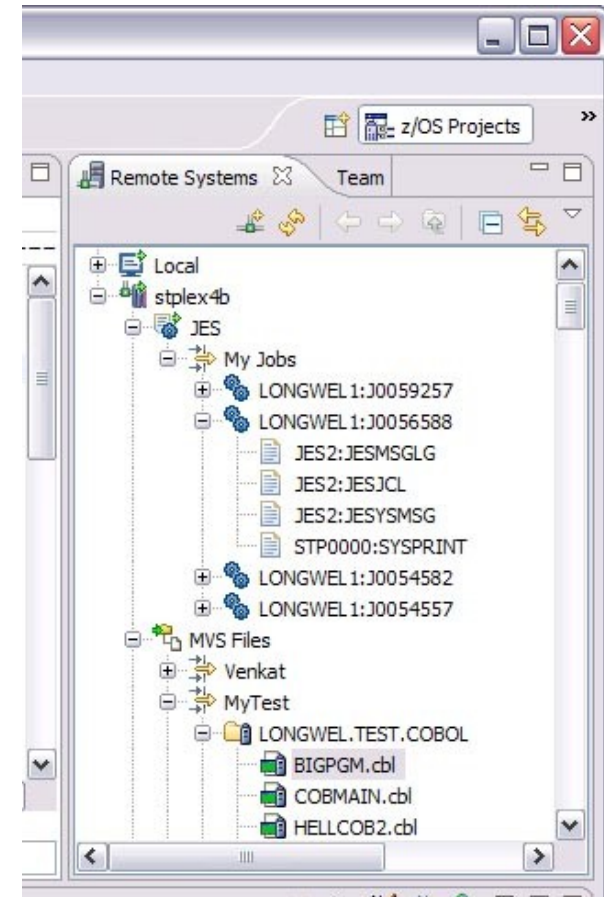
The bottom-right pane displays an error message. The search term is '\*OC4\*'. The results show:

Explanation	Results
OC4	Explanation: A program interruption occurred, but no routine had been specified to handle this type of interruption. Refer to the instruction description in Principles of Operation to find out how the instruction stops processing for the error condition.



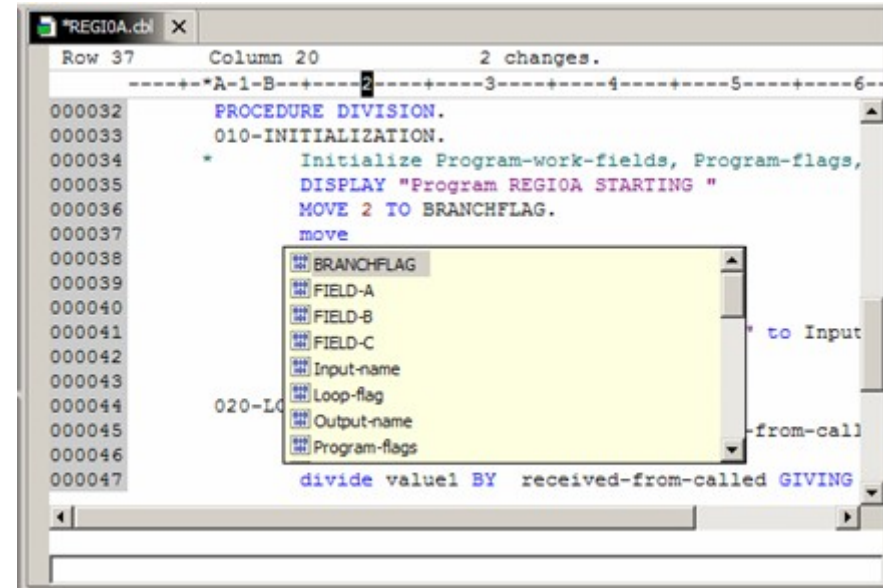
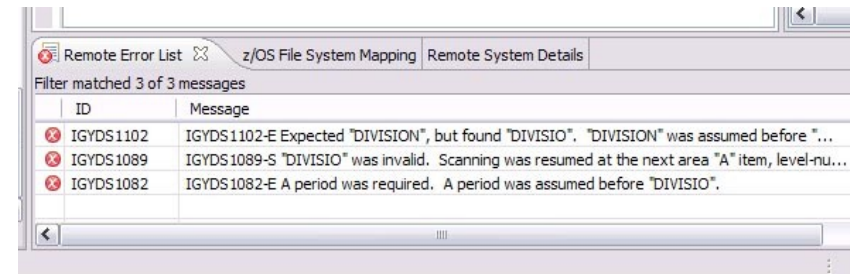
# Navigate datasets or jobs live on z/OS

- Connect to multiple hosts concurrently
- Respects existing security configurations and user IDs
- Search, filter, browse, edit, compare, migrate, and allocate new MVS datasets and USS files
- Copy source code, members, or datasets between systems with a few mouse clicks.
- Access JES queues submit jobs, view job state, and open output spools
- Submit TSO or USS commands
- Add datasets and members into projects to group applications and work items together logically
- Open an emulator in the IDE to configured hosts



# Edit and syntax check source code

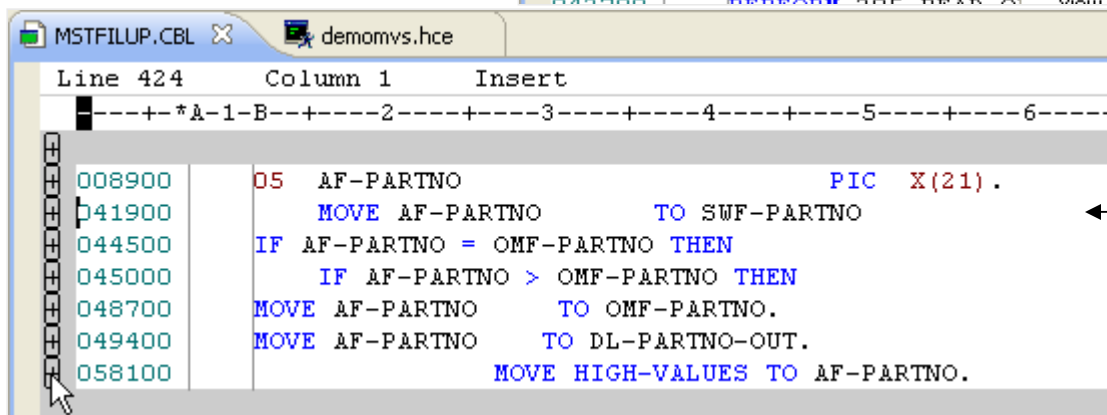
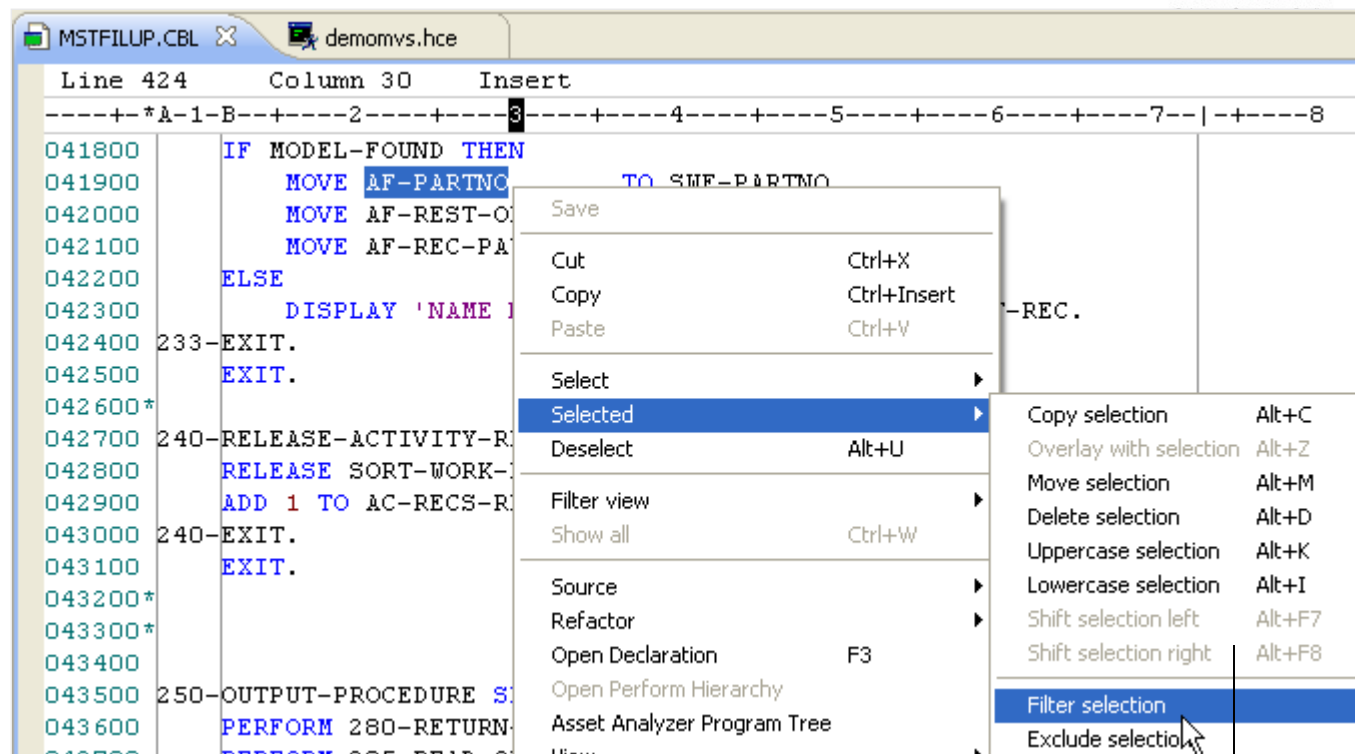
- *Use advanced editing technology to:*
  - ▶ Work with multiple source and JCL members concurrently from different systems
  - ▶ Perform ISPF-like commands in the workstation editor (e.g., FIND, CHANGE, INSERT LINE, etc)
  - ▶ Use syntax highlighting and code-completion to gain insight into available variables, verbs, and keywords
  - ▶ Quickly create programs from code templates, pattern definitions, or UML
  - ▶ Verify COBOL syntax with feedback as you type in real-time
- *Issue syntax check commands against project source code*
  - ▶ Syntax check remotely to ensure proper code structure before compilation
  - ▶ Syntax check locally ensure proper code structure and reduce server usage. RDz will download code and dependencies (e.g., copybooks) to the workstation as necessary
  - ▶ Syntax Errors are listed in the Remote error list. Double-click on the error to open the dataset and move to the line where the error was found

ID	Message
IGYDS1102	IGYDS1102-E Expected "DIVISION", but found "DIVISIO". "DIVISION" was assumed before "...
IGYDS1089	IGYDS1089-S "DIVISIO" was invalid. Scanning was resumed at the next area "A" item, level-nu...
IGYDS1082	IGYDS1082-E A period was required. A period was assumed before "DIVISIO".

# Isolating Code Elements

- Isolate and filter source code
- Multiple editor styles
  - ▶ “Eclipse”-style
  - ▶ ISPF-style
- Bookmark, and/or Expand & Collapse code details

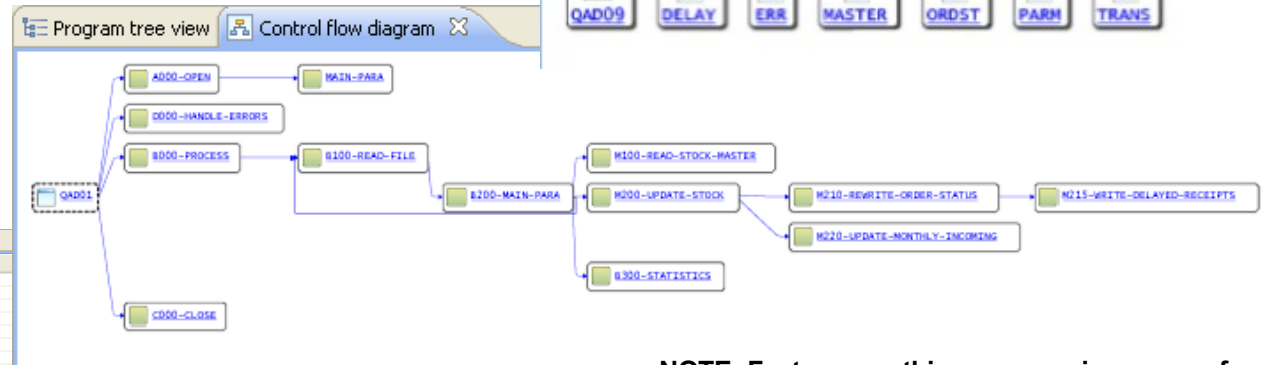


# Analyze applications using graphical diagrams

- **Bring application analysis information into the IDE to aid in program development and understanding**
  - ▶ Link code to data and runtime resources
  - ▶ Visualize code structure and flow
- **Understand the effect of changes made in the IDE when deployed into production**
  - ▶ Run impact analysis on code to determine affected modules
  - ▶ Size testing efforts and create workspaces for changes

Program tree view | Data element table

Name	Level	Type
DELAY-FILE	0	FD
DELAY-STAT	1	UNKN
ERROR-DA	5	NUMB
ERROR-DES	5	CHAR
ERROR-FIL	5	FD
ERROR-REC	5	GRP
ERROR-STA	5	UNKN
FILLER	5	CHAR
IMPACT-ANALYSIS	5	CHAR
INP-DELAY-RCPT-REC	1	GRP
INP-DRCP-EXPECTED-DT	5	CHAR
INP-DRCP-PT-NO	5	CHAR
INP-DRCP-PENDING-QTY	5	NUMB
INP-DRCP-RECEIVED-DT	5	CHAR
INPUT-FILE	0	FD
INPUT-PART-IN	5	NUMB
INPUT-PART-NO	5	NUMB
INPUT-REC	1	GRP
INPUT-STATUS-OK	88	UNKN
MASTER-FILE	0	FD
MASTER-INVALID-REC	88	UNKN



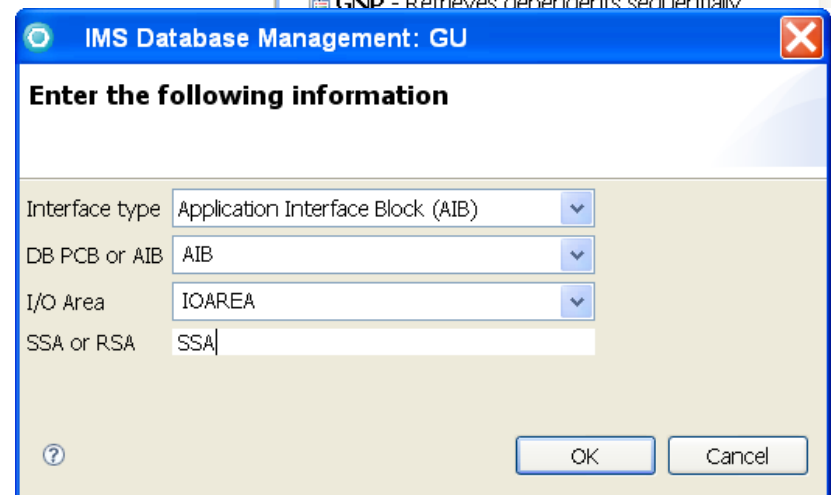
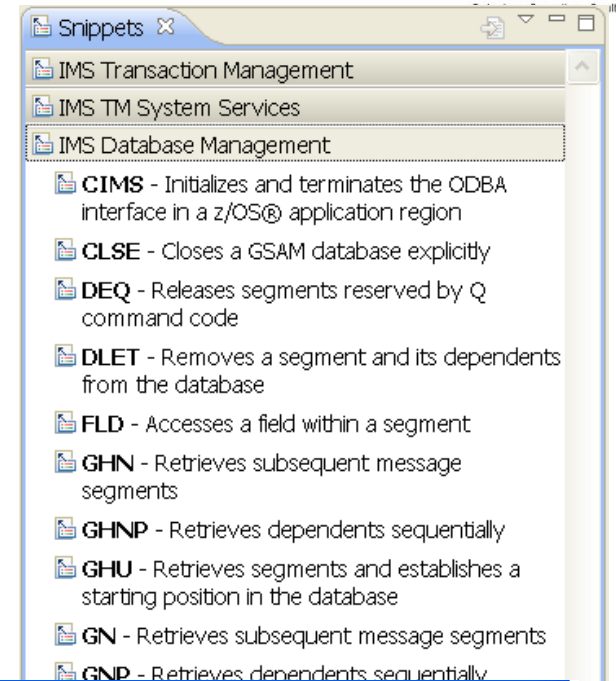
**NOTE:** Features on this page require usage of Rational Asset Analyzer in conjunction with Rational Developer for System z or zEnterprise



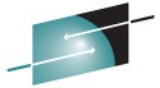
# Speed Development with Code generation



- Model Driven Development
  - use UML to generate COBOL code
- CICS
  - Create working CICS-DB2 CRUD transactions
- IMS statement insertion
  - 71 IMS code generation wizards aid to create IMS COBOL code inline
- DB2
  - Stored Procedure wizards
- Batch applications
  - VSAM / QSAM access program creation
  - Pattern-based code creation preview



# Interactive test of database queries



The screenshot displays a database application interface with three main panes:

- Left Pane (cursavg.cbl):** Contains COBOL code. A section is highlighted in blue:
 

```
EXEC SQL
  DECLARE C1 CURSOR FOR
  SELECT DEPT, MIN(PERF), MAX(PERF), AVG(PERF),
         MIN(HOURS), MAX(HOURS), AVG(HOURS)
  FROM EMPL, PAY
  WHERE EMPL.NBR = PAY.NBR
  GROUP BY DEPT
END-EXEC.
```
- Middle Pane (\*Script1.sql):** Contains the SQL query:
 

```
SELECT DEPT,
  MIN(PERF), MAX(PERF), AVG(PERF),
  MIN(HOURS), MAX(HOURS), AVG(HOURS)
FROM EMPL, PAY
WHERE EMPL.NBR = PAY.NBR
GROUP BY DEPT
```
- Right Pane (EMPL):** Shows a table with columns NBR, LNAME, and FNAME. A context menu is open over the table, with the 'Insert Row' option selected.

At the bottom, the **SQL Results** pane shows the execution status and a table of results:

Status	Operation	Date
Warning	SELECT DEPT, MIN(PERF), MAX(PERF), AVG(PERF), MIN(HOURS), MAX(HOURS), AVG(HOURS)	10/24/09

DEPT	2	3	4	5	6	7
1 ACC	2	3	2	15.99	26.75	21.3700
2 FIN	2	4	3	8.89	32.45	22.6966
3 MKT	1	3	1	6.11	32.41	17.2500
4 R&D	1	1	1	43.59	43.59	43.5900
5 NULL	NULL	NULL	NULL	67.82	67.82	67.8200

## Use Case – Database Applications:

1. Copy/Paste your SQL Declare into a SQL Script and run it – verify results
2. Without doing any other navigation open a test table, and edit row values – or modify the statement (or both)
3. Re-run the SQL Script – verify results
4. Return to step 2 – repeat until satisfied with functionality

# Edit a program and edit its copybooks (all at the same time)



The screenshot displays three windows in the SHARE environment:

- MSTFILUP.CBL**: A COBOL program with lines 100-222. A context menu is open over line 105, showing options like Save, Cut, Copy, Paste, Select, Deselect, Filter view, Show all, Source, Refactor, Open Declaration, Open Perform Hierarchy, Asset Analyzer Program Tree, View, Run As, Debug As, Profile As, Validate, Software Analyzer, Team, Compare With, Replace With, Start Flagging Changed Lines, Local Syntax Check, Browse Copy Member, Open Copy Member, and Content assist.
- PERSON.CPY**: A copybook with lines 1-10. It contains a COPYLIB for PERSONNEL DATABASE and a RECORD LENGTH IS 80 BYTES. It defines fields like PERSON-NUMBER, PERSON-NAME, PERSON-FIRST-NAME, PERSON-LAST-NAME, PERSON-STREET-ADDRESS, PERSON-CITY-ADDRESS, PERSON-STATE-ADDRESS, PERSON-SALARY, and FILLER.
- PERSNFIL.CPY**: A copybook with lines 1-10. It defines fields like PERSON\_LAST\_NAME, PERSON\_STREET\_ADDR, PERSON\_CITY\_ADDR, PERSON\_STATE\_ADDR, PERSON\_SALARY, and PERSONFL.

# Interactive Debug of applications running on z/OS

SHARE

**Debug - ENGLAND.COBLIST.LISTING(COBDATE) - IBM Rational Developer for System z**

File Edit Navigate Search Project Data Run Window Help

Upgrade to Optim Debug

**Debug** Servers

COBDATE [Remote Compiled Application]  
Platform: zOS 390X Connection: 9.30.5.17:17920  
Thread: 1 (Runnable)  
COBDATE : 01  
Process: 874578704 Program: COBDATE

**Variables**

Name	Value
PICSTR	
PICSTR-LENGTH	+00050
PICSTR-STRING	'WWWWWWWWWZ, MMMMMMMM DD, YYYY
LILIAN	+0000155928
FORMATTED-DATE	'SUNDAY, SEPTEMBER 13, 2009

**COBDATE.cbl** ENGLAND.COBLIST.LISTING(COBDATE)

Line	Column	Insert	Browse
28		Move 6 to PicStr-Length	
29		Call "CEEDAYS" Using ChrDate, PicStr, Lilian, Omitted.	
30	*		
31		Move 'WWWWWWWWZ, MMMMMMMM DD, YYYY ' to PicStr-String .	
32		Move 50 to PicStr-Length .	
33		Call "CEEDATE" USING Lilian, PicStr, Formatted-Date, Omitted.	
34		DISPLAY "*****".	
35		Display Formatted-Date .	
36		DISPLAY "*****".	

**Registers**

Name	Value
General Purpose	
%GPR0	342925A4
%GPR1	3420027E
%GPR2	000127FC
%GPR3	342003D0
%GPR4	34200038
%GPR5	342108E0
%GPR6	00000000

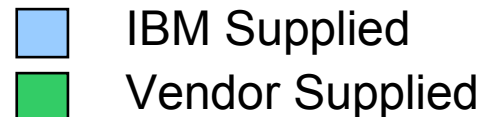
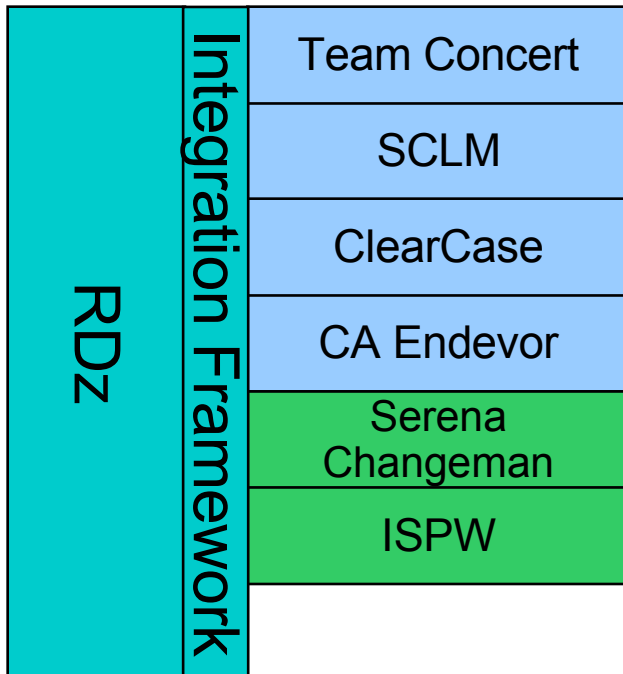
**Console** Tasks Problems JUnit Debug Console Memory Lookup

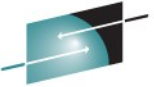
The subblocks in this program are nested as follows:  
1 COBDATE  
Program was stopped due to line/statement breakpoint at statement 35.

Debug Engine Command: Enter Commands...

# Access source code...

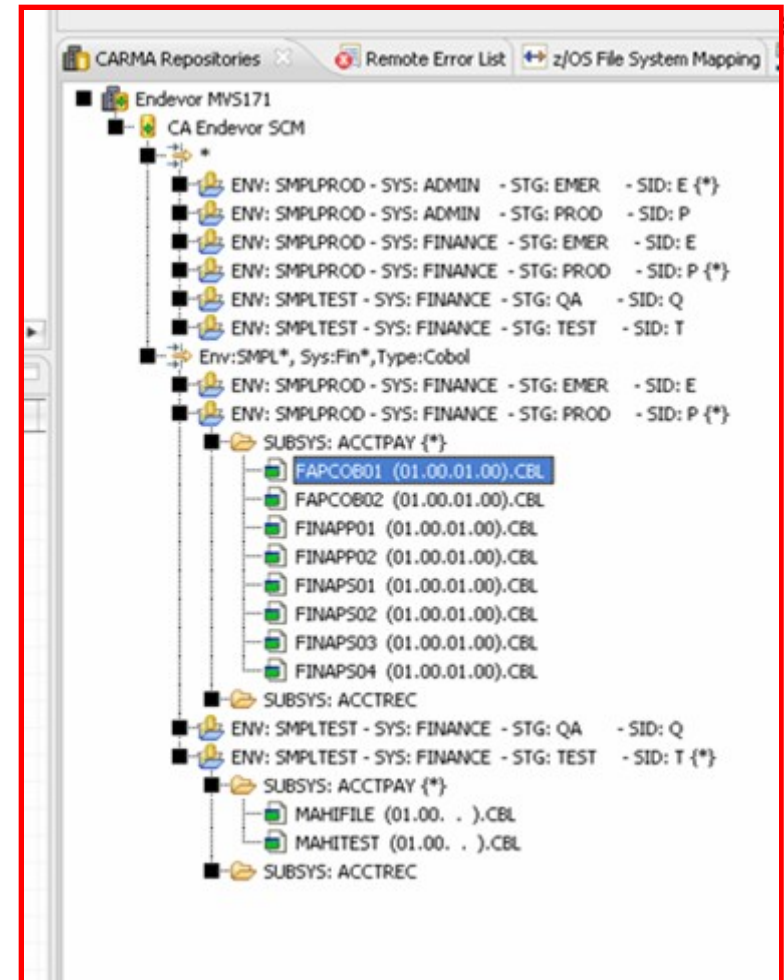
- RDz offers integration into a variety of Source Code Management (SCM) tools as well as a framework for creating SCM integration on your own
- Several vendors supply plug-ins to RDz to provide easy access to processes and source code controlled by their products





# Endevor Integration

- Filter and search through environments, systems, subsystems, members, and stages based on queries (equivalent to DISPLAY)
  - ▶ Filters saved across z/OS sessions
  - ▶ Easy access to common searches and members
  - ▶ Drill down into subsystems
- RETRIEVE members to z/OS projects
  - ▶ Access to typical RDz functionality like syntax check, content assist, debug, etc
- ADD/UPDATE members with single click
  - ▶ RDz remembers Endevor location for retrieve and adds back
- QuickEdit (browse) members from CARMA interface
- Integration with existing GENERATE configuration

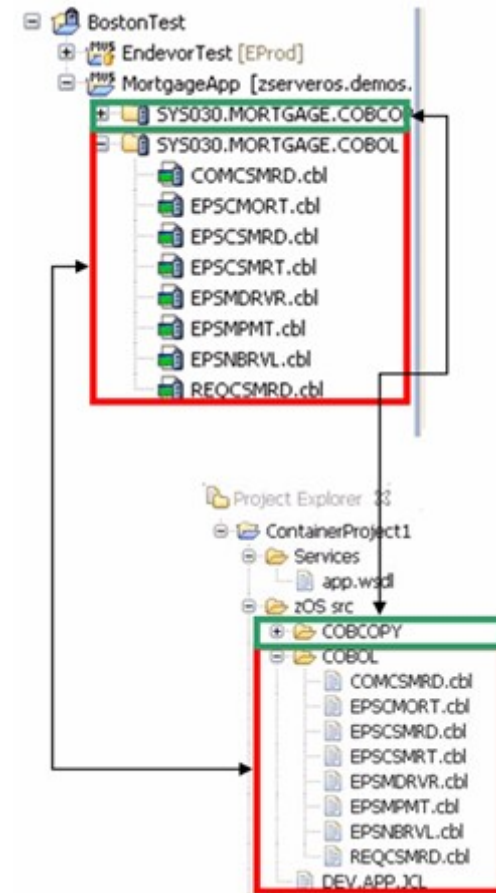




# RTC integration with RDz - all tasks within a single IDE

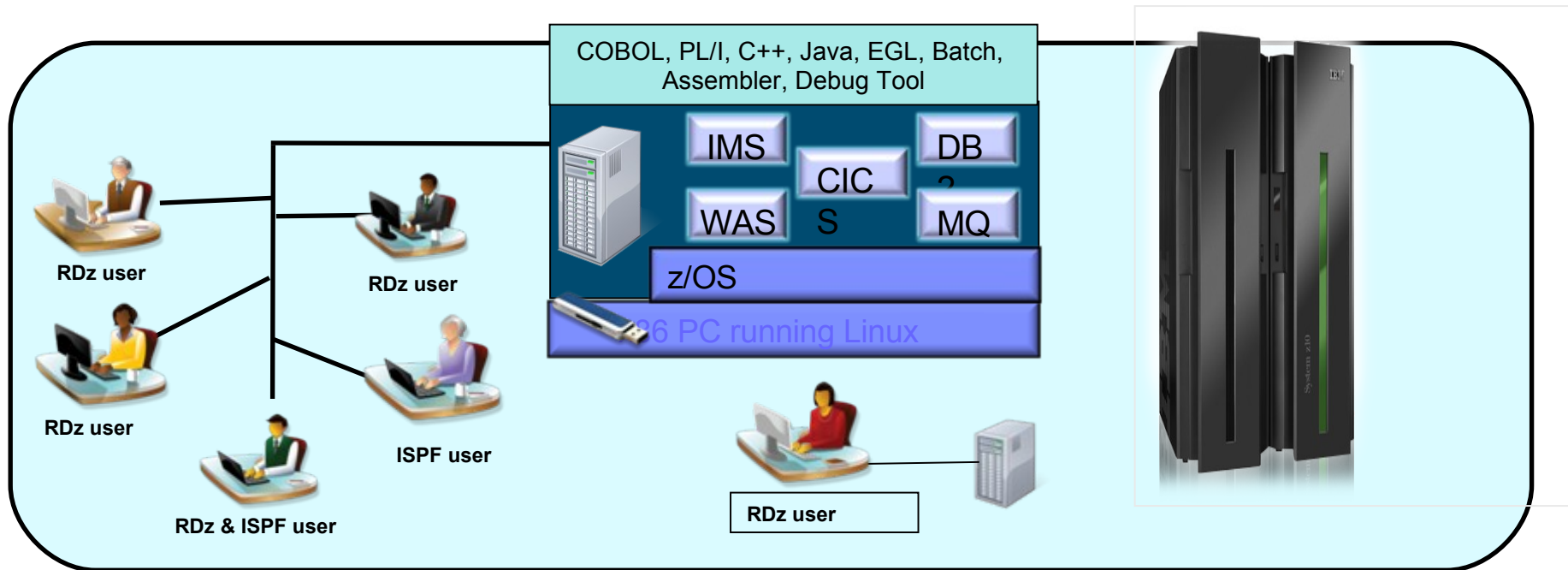


- RTC provides
  - ▶ agility, collaboration and process
  - ▶ Work item planning and coordination
  - ▶ SCM and Build functions for z/OS (and other platforms)
- RDz augments the development productivity & experience
  - ▶ files act as if on the host
  - ▶ Appropriate editors (COBOL, maps, etc.) and functions (content assist, syntax check, etc.)
  - ▶ High value functions (XML enablement, SFM, code generation from models, from UML, etc)
- RDz projects in RTC
  - ▶ RDz projects are a view into the RTC project
  - ▶ RDz projects provide a working set for the developer



# RDz Unit Test Feature

*Assisting application development for System z*



- Liberate developers to rapidly prototype new applications
- Develop and test System z applications anywhere, anytime!
- Free up mainframe development systems for production capacity
- Eliminate costly delays by reducing dependencies on operations staff

Note: This Program is licensed only for development and test of applications that run on IBM z/OS. The Program may not be used to run production workloads of any kind, nor more robust development workloads including without limitation production module builds, pre-production testing, stress testing, or performance testing.



# Agenda

- Application Development is Hard
- Tools to the rescue!
- **Using tools is Hard**
- Continual Discovery
- Reprise: Application Development is Hard

# Using Tools - There is a Learning Curve

- It's true.
- Any new tool is, at first, overwhelming.
- ISPF and “green screen” users
  - Compatibility mode for editor environment
  - Remote system access
  - 3270 emulator included
- IDE-comfortable users
  - Multiple edit window support
  - multiple editor selections available based on preference
  - extensive preferences and customization
  - context menus for most common tasks

# Agenda

- Application Development is Hard
- Tools to the rescue!
- Using tools is Hard
- **Continual Discovery**
- Reprise: Application Development is Hard

# Continual Discovery

- Challenge Yourself
  - Resolve to learn something new every day
- Find a Buddy
  - learn from what each of you have found
- Impress your Friends
  - find out something cool? Share you knowledge!
- Be Social!
  - Join a user group, discussion group, or online community
  - ask questions, or just lurk and learn
  - (See links at end of presentation)

# Agenda

- Application Development is Hard
- Tools to the rescue!
- Using tools is Hard
- Continual Discovery
- Reprise: Application Development is **NOT SO** Hard

# The hardest part of Application Development is ...



- Getting started

# The hardest part of Application Development is ...



- Getting started ... and
- Staying engaged and on task

# Application development for z/OS used to be like this ...



```
File Edit Edit_Settings Menu Utilities Compilers Test Help

EDIT      DDS0001.TEST.COBO(LHOSPEDIT) - 27.26      Columns 00001 00072
***** ***** Top of Data *****
000001      *****
000002      IDENTIFICATION DIVISION.
000003      PROGRAM-ID.  HOSPEDIT.
000004      AUTHOR.  JON SAYLES.
000005      INSTALLATION. COBOL DEVELOPMENT CENTER.
000006      DATE-WRITTEN. 01/01/08.
000007      DATE-COMPILED. 01/01/08.
000008      SECURITY. NON-CONFIDENTIAL.
000009
000010      *****
000011      *  A new comment ...
000012      *  Jon's new comment
000013      ENVIRONMENT DIVISION.
000014      CONFIGURATION SECTION.
000015      SOURCE-COMPUTER.  IBM-390.
000016      OBJECT-COMPUTER. IBM-390.
000017      INPUT-OUTPUT SECTION.
Command ==>
F1=Help      F2=Split      F3=Exit      F5=Rfind      F6=Rchange      F7=Up
F8=Down      F9=Swap      F10=Left     F11=Right     F12=Cancel
```

And it still can be ...



# ... but why do that when you can use all these features?



The screenshot displays the IBM Rational Developer for System z interface. The main editor window shows a COBOL program named COBDATE.cbl. The code includes a WORKING-STORAGE SECTION with variables for dates and strings, and a PROCEDURE DIVISION with a MAIN-LOGIC section. The program is running, and an error has occurred, indicated by the 'Remote Error List' pane at the bottom. The error is a 0C4 (Program interruption) with the explanation: 'A program interruption occurred, but no routine had been specified to handle this type of interruption. Refer to the instruction description in Principles of Operation to find out how the instruction stops processing for the error condition.'

**COBDATE.cbl**

```
Line 16      Column 1      Insert
-----*A-1-B-----2-----3-----4-----5-----6-----
WORKING-STORAGE SECTION.
01  ChrDate.
   05  ChrDate-Length      pic s9(4) comp value 10.
   05  ChrDate-String      pic x(10).
01  PicStr.
   05  PicStr-Length       pic s9(4) comp.
   05  PicStr-String       pic x(80).
77  Lilian                 pic s9(9) comp.
77  Formatted-Date         pic x(80).

PROCEDURE DIVISION.
*
MAIN-LOGIC.
  DISPLAY "Starting COBDATE".
  DISPLAY "-----"
  ACCEPT ChrDate-String from DATE.
```

**Remote Error List**

Search	Explanation	Results
*0C4*	0C4	Explanation: A program interruption occurred, but no routine had been specified to handle this type of interruption. Refer to the instruction description in Principles of Operation to find out how the instruction stops processing for the error condition.

# Rational Developer for System z and zEnterprise



- Eases developers into working with System z systems
  - **Easier to get started**
- Offers many features within the Integrated Development Environment
  - **Keeps developers on task and engaged**



[www.ibm.com/software/rational](http://www.ibm.com/software/rational)

© Copyright IBM Corporation 2010,2011. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

# Useful Links

- Rational Developer for System z Information:
  - <http://www.ibm.com/software/rational/products/developer/systemz/>
- Rational Developer for System z Infocenter:
  - <http://publib.boulder.ibm.com/infocenter/ieduasst/rtnv1r0/index.jsp?topic=/com.ibm.iea.rdz/rdz/wdz76.html>
- Additional Video Demonstrations:
  - [http://websphere.dfw.ibm.com/atdemo/atdemo\\_rdz.html](http://websphere.dfw.ibm.com/atdemo/atdemo_rdz.html)
  - [http://websphere.dfw.ibm.com/atdemo/atdemo\\_rdz\\_zosad\\_recorded.html](http://websphere.dfw.ibm.com/atdemo/atdemo_rdz_zosad_recorded.html)
- Rational Developer for System z - Distance Learning:
  - <https://www.ibm.com/developerworks/mydeveloperworks/wikis/home?lang=en#/wiki/Learn%20RDz-Learn%20COBOL>
- Rational “COBOL Cafe” online discussion group:
  - <https://www.ibm.com/developerworks/rational/community/cafe/cobol.html>
- Rational Developer for System z - RDz hub:
  - <https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=df67969e-ba40-44c7-a1ca-ef4a2aa99e01>
- My information
  - email: [hahnt@us.ibm.com](mailto:hahnt@us.ibm.com)
  - Blog: <https://www.ibm.com/developerworks/mydeveloperworks/blogs/applicationmodernization>