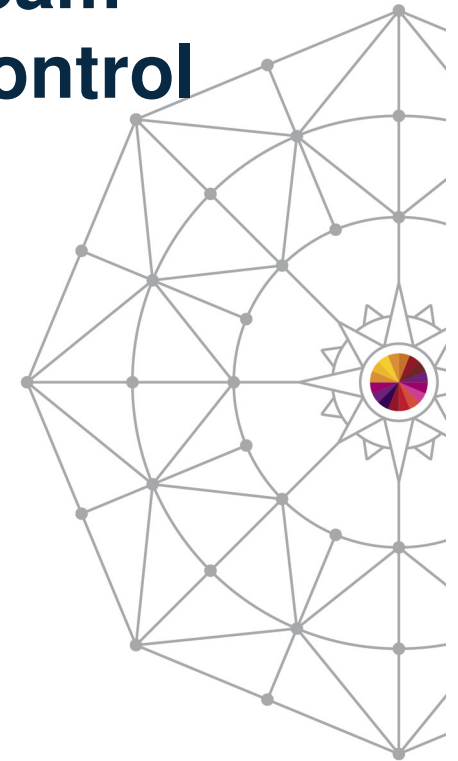


# Setting up and using Rational Team Concert's ISPF Client for source control

Liam Doherty  
IBM Corporation

Thursday August 7th, 2014  
Session 16110



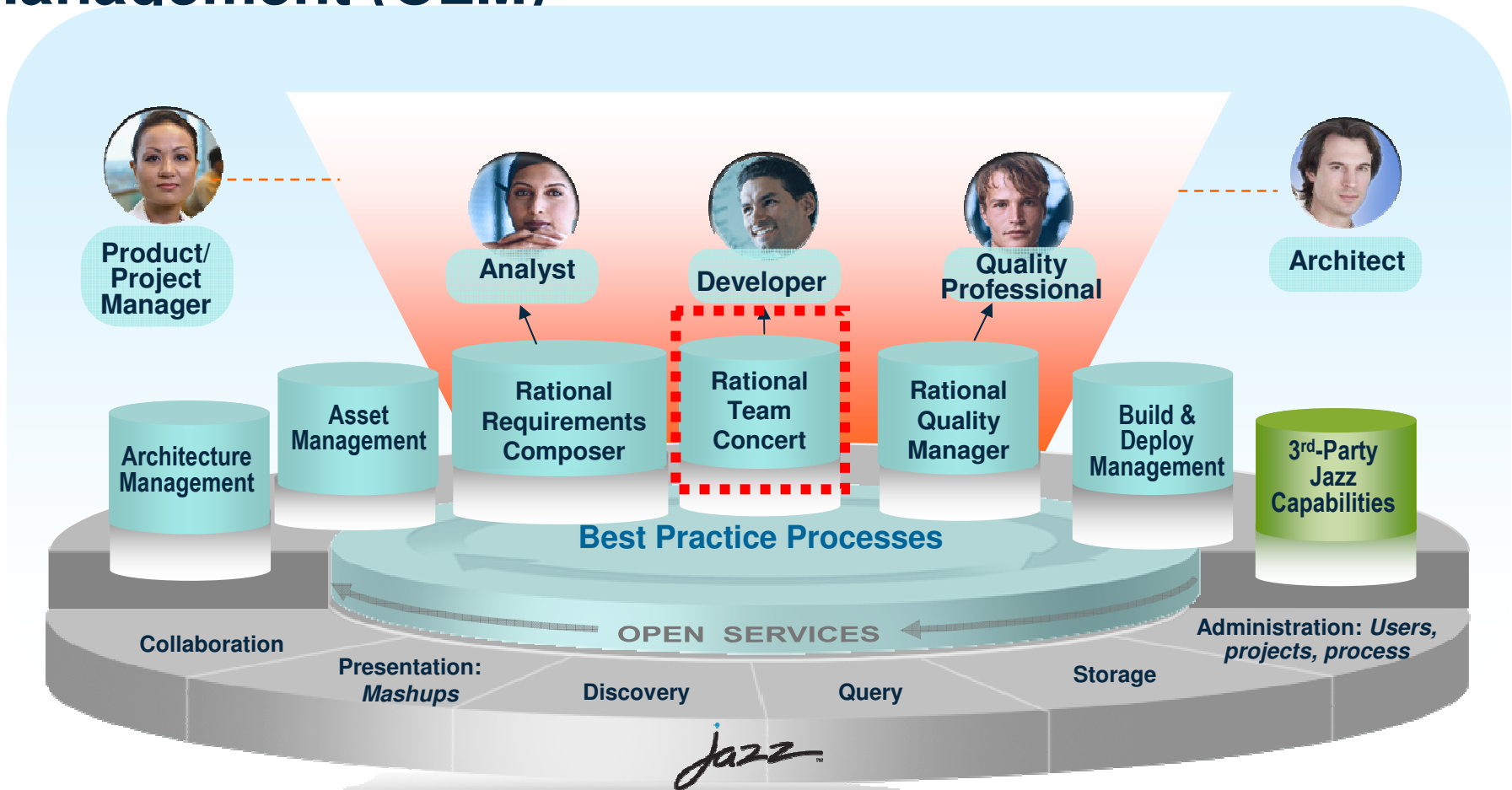
#SHAREorg



# Agenda

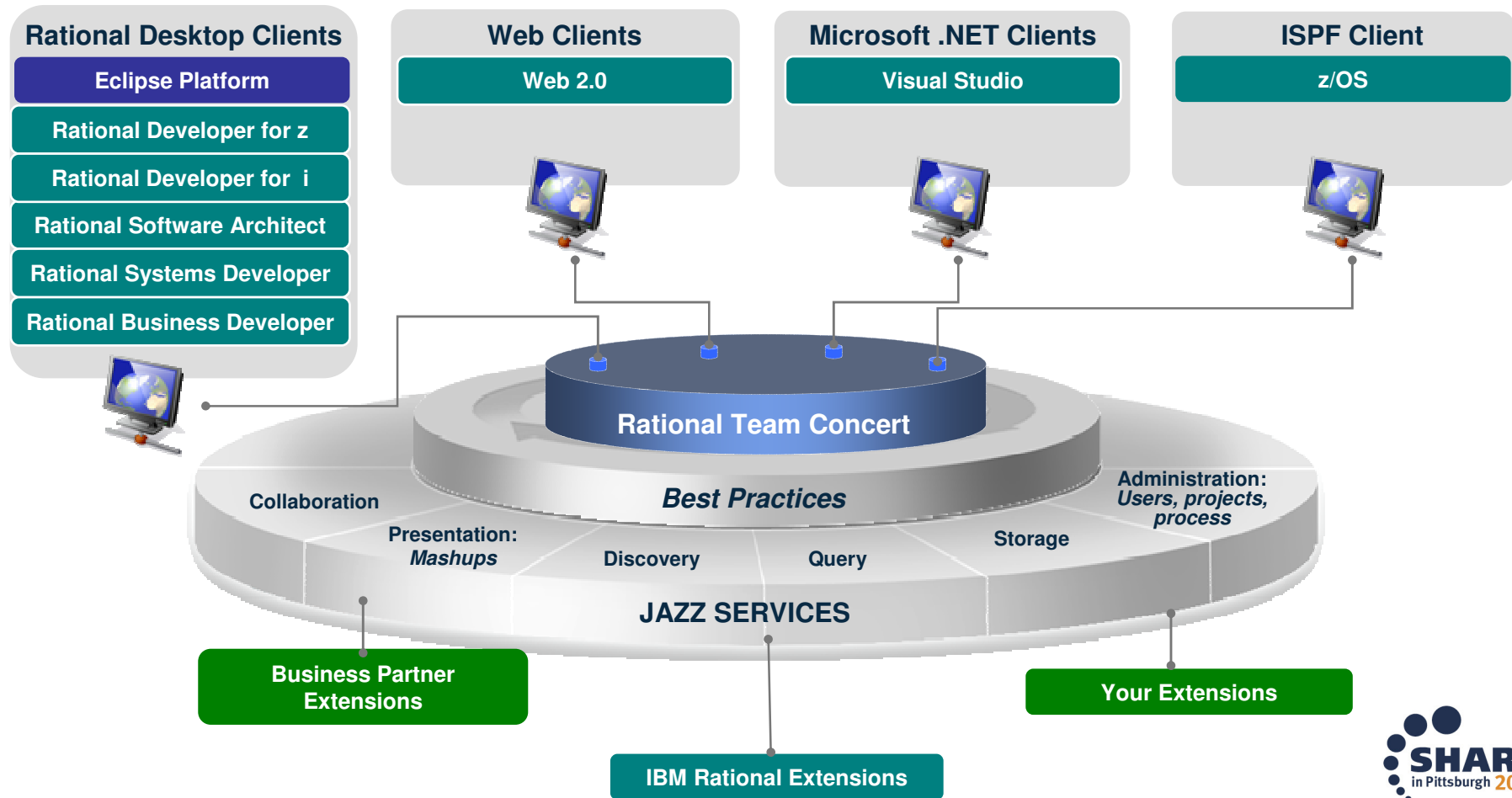
- What is Rational Team Concert?
- The Eclipse interface
  - The RTC repository
  - Streams, Components and projects
  - zComponent projects
- Setting up Enterprise Extensions System Definitions
- Setting up the Rational Team Concert ISPF Client
  - RTC Configuration utility
- Using the Rational Team Concert ISPF Client

# IBM Rational Collaborative Lifecycle Management (CLM)



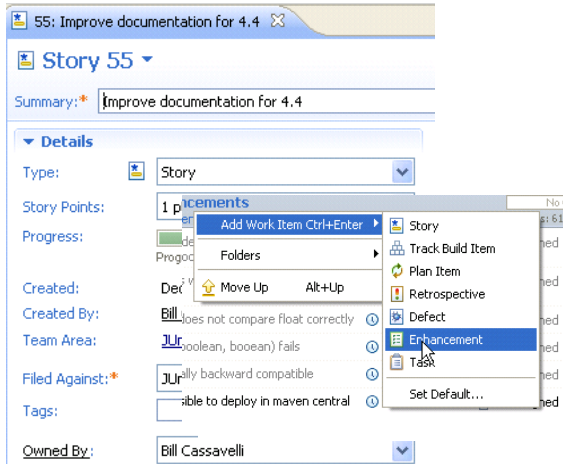
*Robust extensible solution for the entire extended development team*

# Rational Team Concert (RTC): An open, extensible architecture *Supporting a broad range of desktop clients, IDEs and languages*

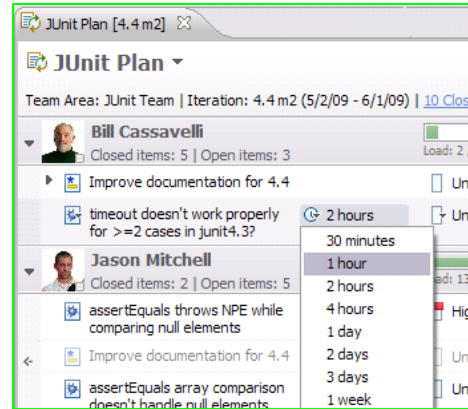


# Rational Team Concert – A single tool, many capabilities

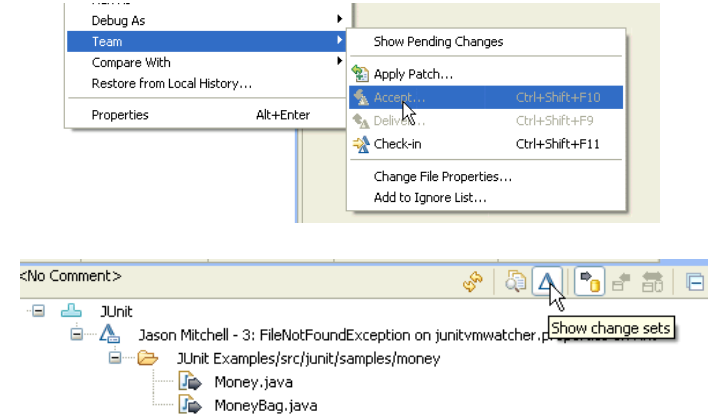
- Work Items



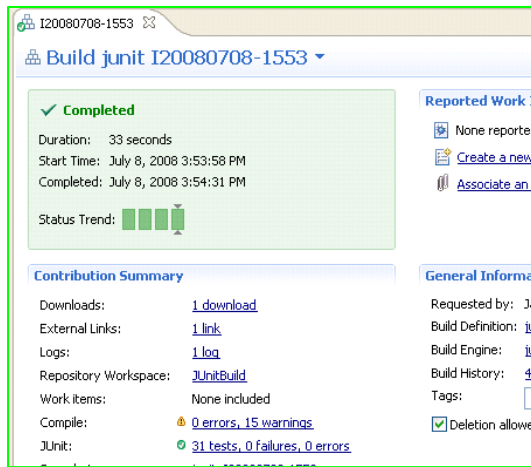
- Planning



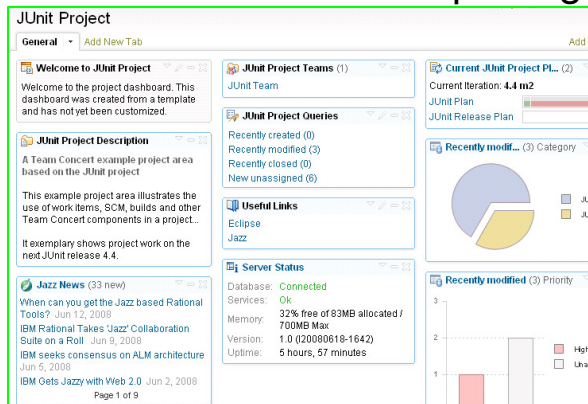
- Source Control



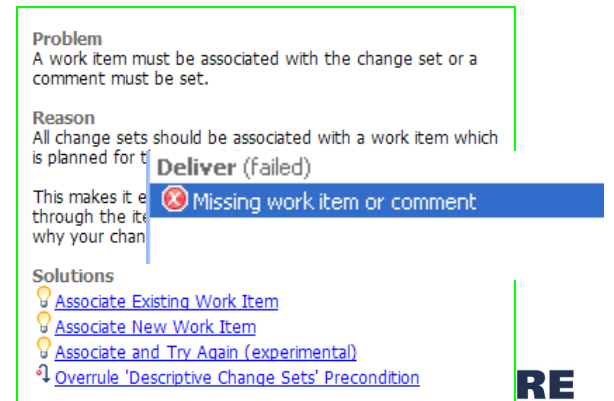
- Builds – Continuous



- Dashboards & Reporting



- Method Enforcement and Automation



# What is Rational team Concert?

- **So RTC is more than just an Software Configuration Management system**
  - Process, Planning and Work items coupled with an integrated SCM provide a complete solution
  - Ability to manage distributed and z/OS source in the same repository makes for a more integrated SCM solution

# Rational Team Concert terminology

<b>Stream</b>	Collection of components used to organize work, coordinate collaboration and integration, and capture the active configuration of each component. Related to a level in a hierarchy (e.g., promotion levels, releases, etc)	
<b>Component</b>	Collection of related artifacts (i.e., sourcefiles are logically organized into components) that have the same lifecycle Used to control access rights, facilitate sharing and reuse Theoretical limit: 50000 files <b>Recommended: 1000 – 2000 files / component</b>	
<b>Repository Workspace</b>	Workspace for 1 user synchronized with a Stream and the "Sandbox" Situated on the RTC server	
<b>Sandbox</b>	Workspace on the hard disk (e.g. local eclipse workspace). Note: Through the build or CLI you have jazz metadata but no eclipse metadata. For ISPF Client a Sandbox is a collection of data sets with the same HLQ.MLQ	
<b>Change Set</b>	Contains a collection of consistent changes made to a configuration of a component. Means for flowing file and folder changes between repository workspaces and streams.	
<b>Work Item</b>	Captures the tasks and issues to be addressed by the team members Associated with change sets created by the developer. Automatically and dynamically populate plans and reports	
<b>Baseline</b>	Non-editable version of a component capturing an interesting point in time The baseline is performed implicitly when a Snapshot is taken Can be done manually on a given component	
<b>Snapshot</b>	Collection taken of all component baselines for a stream or repository workspace capturing an interesting point in time	

# Rational Team Concert terminology (cont)

<b>Load</b>	Action that copies selected files and folders from the repository workspace to the sandbox (eclipse workspace or MVS data sets)	
<b>Accept</b>	Action that allows for synching the repository workspace reference with changes delivered to the stream by other developers Load of the accepted changes into the sandbox is automatically performed Note – you can also accept change sets from a WI	
<b>Check-in</b>	Action that allows to save local changes into the repository workspace, within a Change Set	
<b>Deliver</b>	Action to push the workspace changes from the workspace to the Stream	



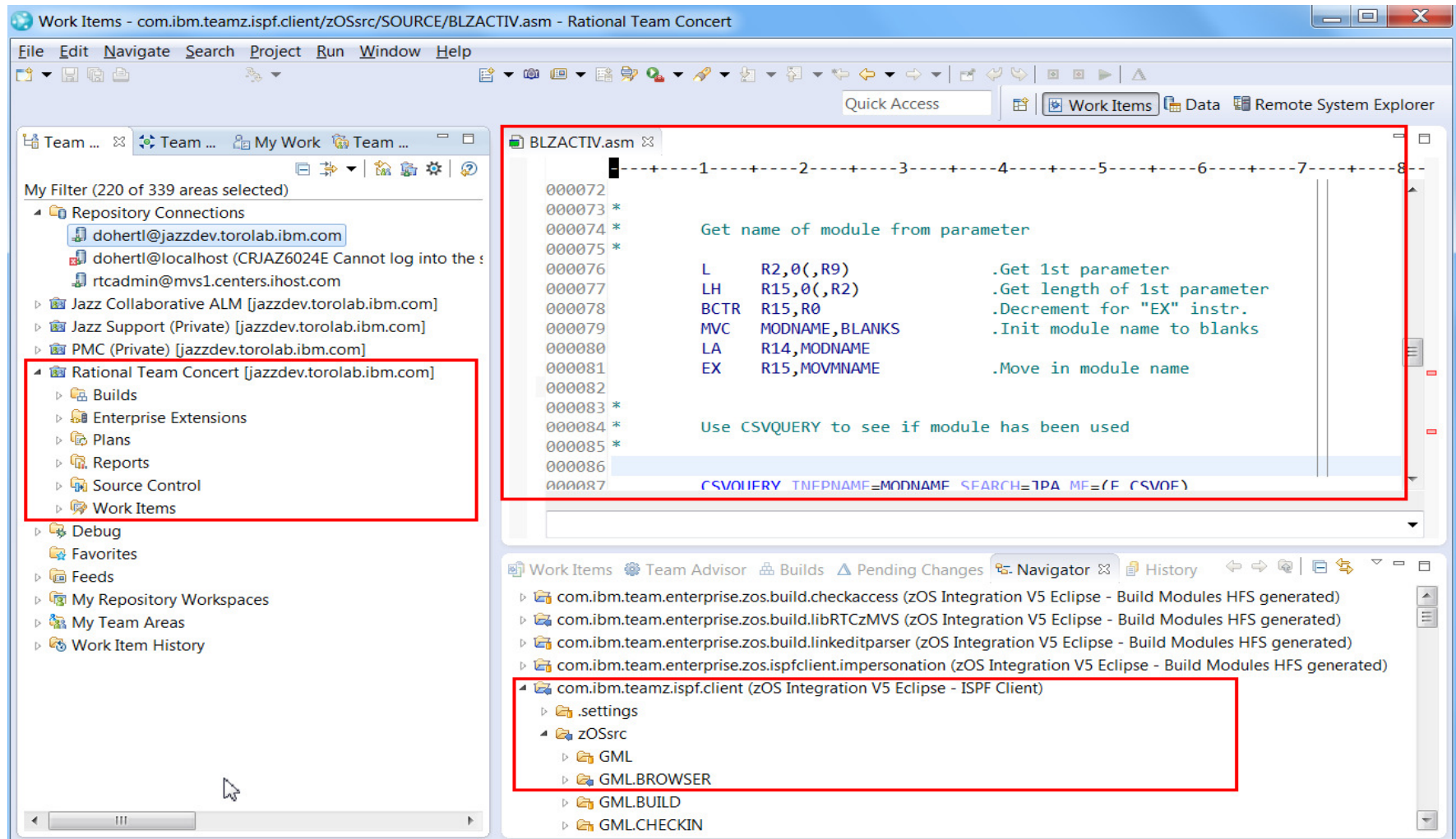
# The Eclipse Interface

- **Rational Team Concert originally offered as an Eclipse Client and a Web Client**
  - Other clients now available such as Visual Studio and of course, ISPF
- **Some functions are only offered through the Eclipse interface**
  - Enterprise Extensions definitions
  - So even though we are going to use the ISPF Client for source control, some administration functions will need to be carried out in Eclipse
- **Let's use the Eclipse interface to familiarize ourselves with some RTC repository terminology...**

# The RTC repository

- **The RTC repository consists of:**
  - Source Code
    - Streams
    - Components
    - Projects
    - Repository Workspaces
  - Work items
  - Plans that consist of related work items
  - Builds
  - Enterprise Extension definitions
  - Reports

# The RTC Repository in Eclipse



The screenshot shows the Eclipse IDE interface. On the left, the Team Explorer shows a tree view of repository connections. The 'Rational Team Concert [jazzdev.torolab.ibm.com]' connection is expanded, and the 'Work Items' folder is highlighted with a red box. The main editor window displays the source file 'BLZACTIV.asm' with the following code:

```

000072
000073 *
000074 *      Get name of module from parameter
000075 *
000076 L      R2,0(,R9)          .Get 1st parameter
000077 LH     R15,0(,R2)        .Get length of 1st parameter
000078 BCTR  R15,R0             .Decrement for "EX" instr.
000079 MVC   MODNAME,BLANKS    .Init module name to blanks
000080 LA   R14,MODNAME
000081 EX   R15,MOVNAME        .Move in module name
000082
000083 *
000084 *      Use CSVQUERY to see if module has been used
000085 *
000086
000087      CSVQUERY TNFPNAME=MODNAME SEARCH=TPA MF=(F CSVQF)

```

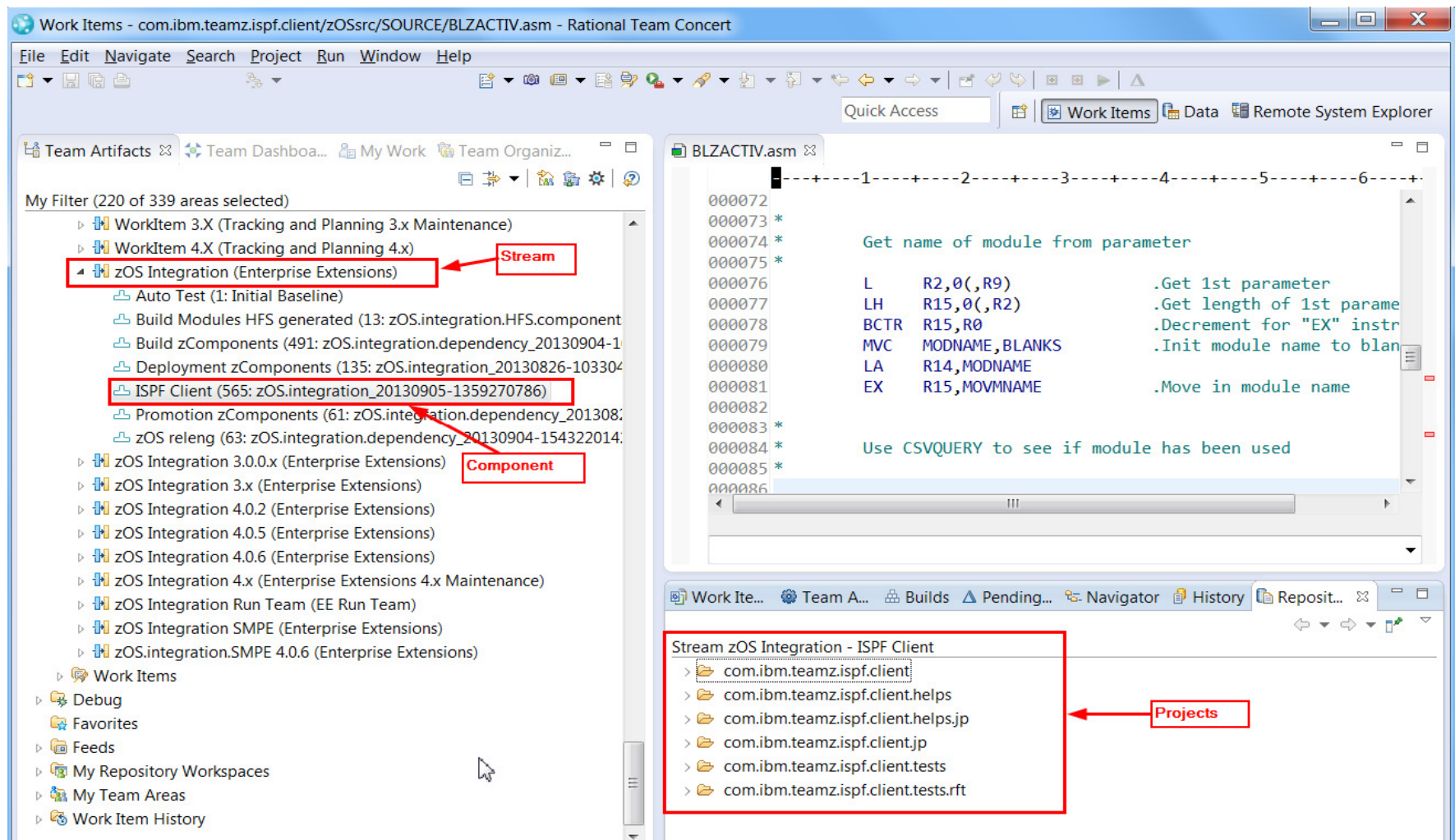
At the bottom, the Navigator shows a project tree for 'com.ibm.teamz.ispf.client (zOS Integration V5 Eclipse - ISPF Client)'. The 'zOSsrc' folder is expanded, and the 'GML' folder is highlighted with a red box. The contents of 'GML' are:

- GML
- GML.BROWSER
- GML.BUILD
- GML.CHECKIN

# Streams, Components and projects

- **Source code is stored in Streams**
  - *Think of a stream as a particular point-in-time version of a project, or part of a project.*
  - *E.g. Current Development, v4.0.6 maintenance, etc*
- **Streams are composed of components**
  - *Components are ways to break down projects or parts of projects.*
  - *The same component will exist in different streams, just at different versions*
- **Components are composed of projects**
  - *These are the physical containers that will hold the code*

# Streams, Components and projects

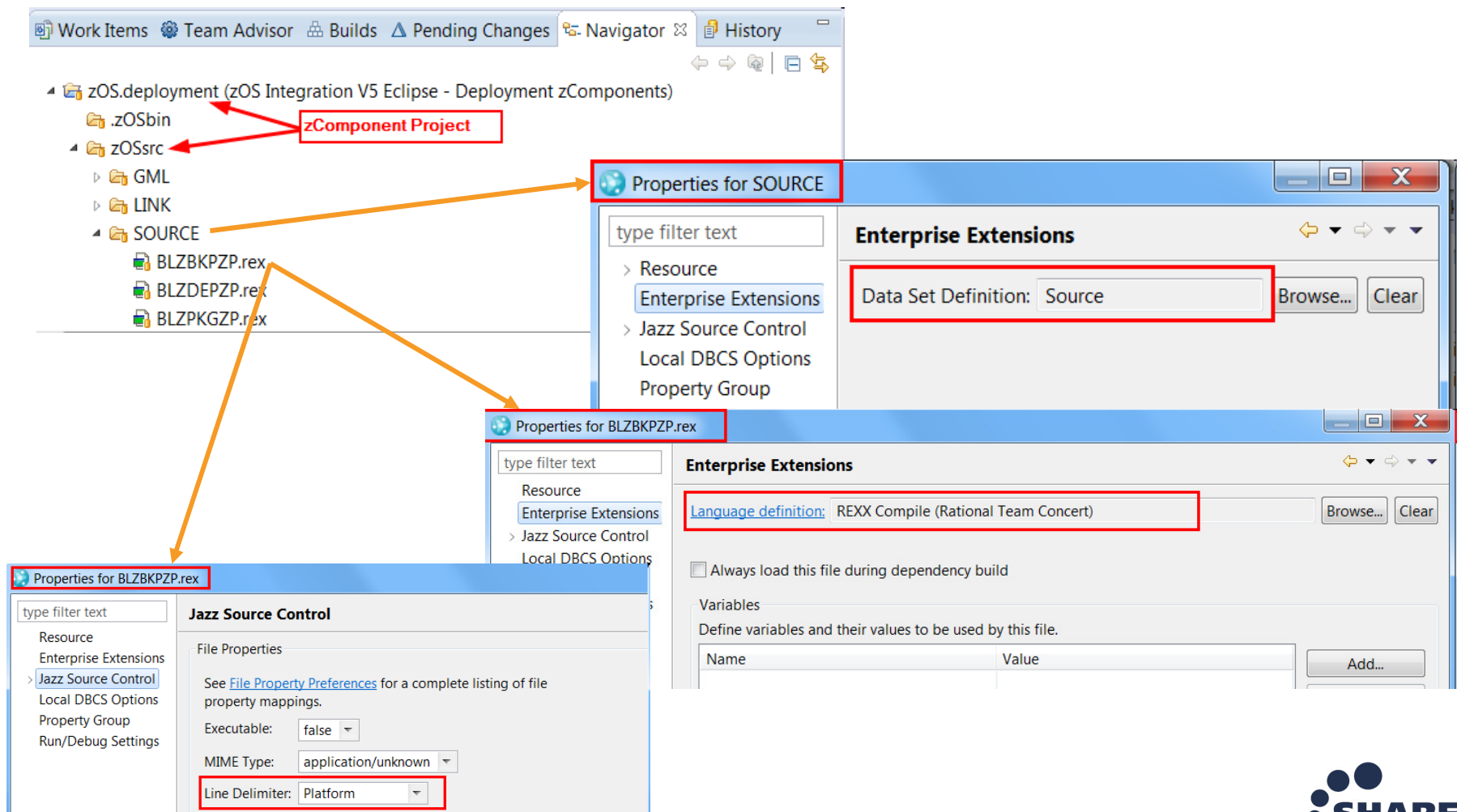


The screenshot displays the Rational Team Concert (RTC) interface. The left-hand pane shows a project tree under 'My Filter (220 of 339 areas selected)'. The tree includes several folders, with 'zOS Integration (Enterprise Extensions)' highlighted by a red box and labeled 'Stream'. Under this folder, 'ISPF Client (565: zOS.integration\_20130905-1359270786)' is also highlighted by a red box and labeled 'Component'. The main editor window shows the source code for 'BLZACTIV.asm', which contains assembly instructions and comments. The bottom-right pane shows a 'Stream zOS Integration - ISPF Client' view, listing a hierarchy of project folders: 'com.ibm.teamz.ispf.client', 'com.ibm.teamz.ispf.client.helps', 'com.ibm.teamz.ispf.client.helps.jp', 'com.ibm.teamz.ispf.client.jp', 'com.ibm.teamz.ispf.client.tests', and 'com.ibm.teamz.ispf.client.tests.rft'. This list is highlighted by a red box and labeled 'Projects'.

# zComponent projects

- **zComponent projects are projects that have a z/OS “nature”, so some specific processing is performed against them**
  - Allows for a data set definition to be assigned to a “**zfolder**”
    - This maps the folder to a data set on z/OS
  - Allows for a language to be assigned to a “**zfile**”
    - This tells RTC how a particular file is going to be built
  - Allows for encoding options to be set such that default EBCDIC code pages or language specific EBCDIC code pages (e.g. IBM-939) will be used on z/OS
    - **Note:** Generally everything is stored in UTF-8 in the repository

# zComponent projects



The screenshot displays the Eclipse IDE interface for configuring zComponent projects. The Navigator on the left shows a project structure under 'zOS.deployment (zOS Integration V5 Eclipse - Deployment zComponents)'. The 'zOSsrc' folder contains subfolders 'GML', 'LINK', and 'SOURCE'. The 'SOURCE' folder contains source files 'BLZBKZP.rex', 'BLZDEZP.rex', and 'BLZPKZP.rex'. Three property windows are open, each with a red box highlighting a specific configuration:

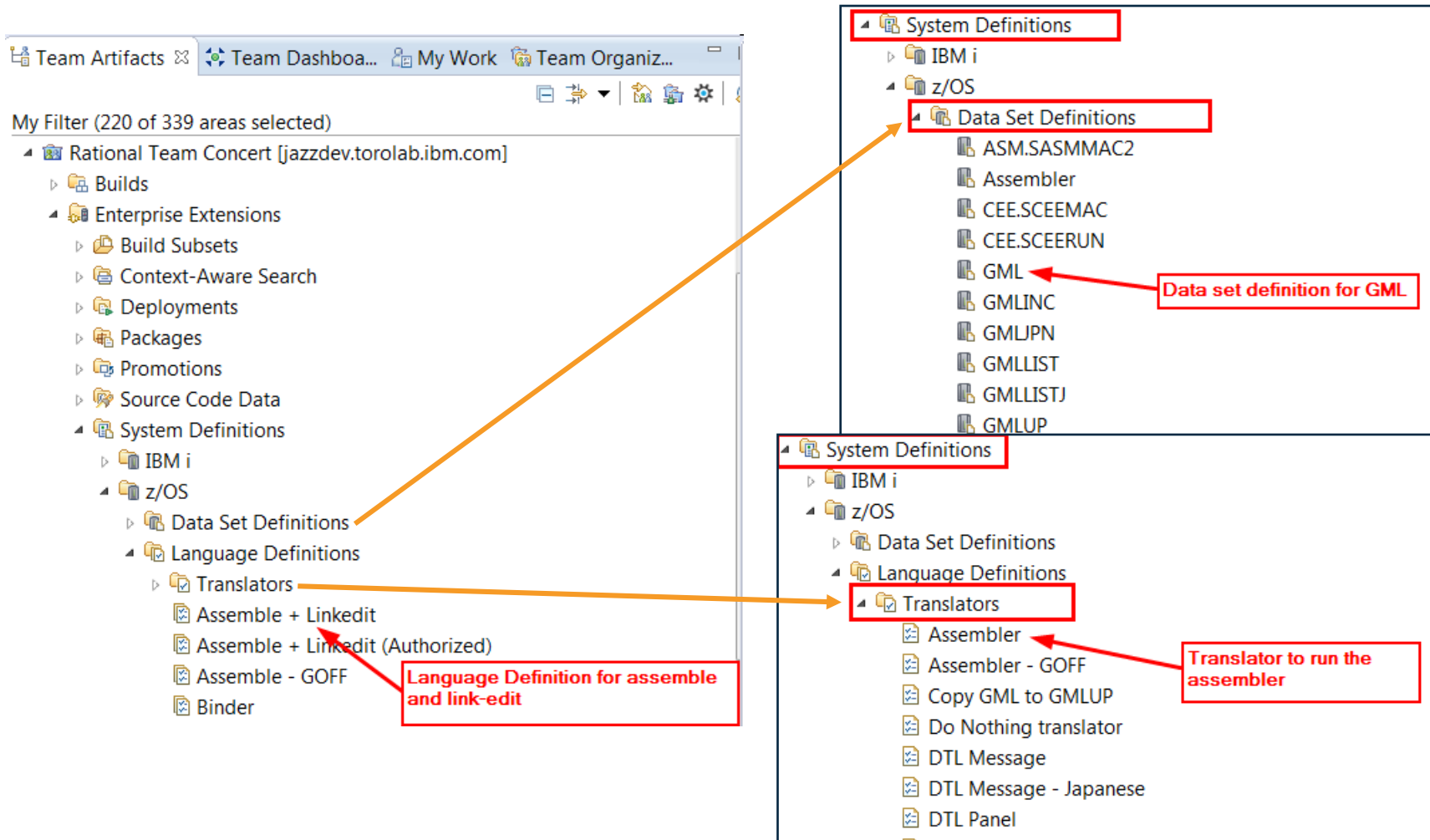
- Properties for SOURCE:** The 'Enterprise Extensions' section has 'Data Set Definition' set to 'Source'.
- Properties for BLZBKZP.rex:** The 'Enterprise Extensions' section has 'Language definition' set to 'REXX Compile (Rational Team Concert)'.
- Properties for BLZBKZP.rex:** The 'Jazz Source Control' section has 'Line Delimiter' set to 'Platform'.

# Setting up Enterprise Extensions System Definitions

- **Regardless of whether you are planning on using the ISPF Client or the Eclipse Client you will need to set up system definitions**
  - Data set definitions for each source type
    - Once set up a data set definition can be used for many zFolders
  - Language definitions for each different type of module
    - You can create a single generic language definition that does nothing, for use for things like JCL, Samples, Interpretive REXX execs, etc
  - Translators define a single step of a build process
    - A language definition is made up of one or more translators



# Setting up Enterprise Extensions System Definitions



The screenshot displays the IBM Team Dashboard interface for configuring system definitions. The left pane shows a tree view under 'My Filter (220 of 339 areas selected)' with 'System Definitions' expanded to 'z/OS' and 'Data Set Definitions'. The right pane shows a detailed view of 'Data Set Definitions' and 'Language Definitions' with red callouts pointing to specific items.

**System Definitions**

- IBM i
  - z/OS
    - Data Set Definitions**
      - ASM.SASMMAC2
      - Assembler
      - CEE.SCEEMAC
      - CEE.SCEERUN
      - GML ← **Data set definition for GML**
      - GMLINC
      - GMLJPN
      - GMLLIST
      - GMLLISTJ
      - GMLUP
    - Language Definitions**
      - Translators**
        - Assembler ← **Translator to run the assembler**
        - Assembler - GOFF
        - Copy GML to GMLUP
        - Do Nothing translator
        - DTL Message
        - DTL Message - Japanese
        - DTL Panel

**Language Definition for assemble and link-edit**

- System Definitions
  - IBM i
    - z/OS
      - Data Set Definitions
      - Language Definitions**
        - Translators
          - Assemble + Linkedit
          - Assemble + Linkedit (Authorized)
          - Assemble - GOFF
          - Binder

# Data set definitions

- Data set definitions to store source
  - Defines the attributes (DCB) of the data set such that when the ISPF Client “loads” a member it knows how to create the data set
  - Similarly Build will need to load any data sets required in build
  - Defines the Low Level Qualifier of the data set. The high level qualifier is specified in the ISPF Client when you load, or is specified in the build definition
    - This allows data set definitions to be generic across versions, with version specific middle level qualifiers specified by the user, or by the build

# Data set definitions

BLZACTIV.asm Source

## Data Set Definition

Name:  Project Area:  Save

### General

Description:

Usage

- Destination data set for a zFolder
- New data set used for build
- Existing data set used for build
- Temporary data set used for build

Data set name:  Member:

Add data set prefix from build definition to data set name

Ignore changes to this system definition during a dependency build

### Data Set Characteristics

Data class:

Storage class:

Management class:

Volume serial:

Generic unit:

Space units:

Primary quantity:

Secondary quantity:

Directory blocks:

Record format:

Record length:

Block size:

Data set type:

# Language Definitions and Translators

- Language definitions are required for source to load
  - Define a dummy language definition if you are not going to build
  - Define Language definitions for source types that are going to be built
- Translators define the actual build process, for example a PL/I compile or a link-edit

# Language Definitions and Translators

BLZACTIV.asm Assemble + Linkedit

### Language Definition

Name: **Assemble + Linkedit** Project Area: Rational Team Concert

#### General

Specify a language for this language definition. Optionally, you can specify file extensions to automatically associate files with this language definition.

Language: **Assembler**

File extensions: **asm**  
Separate extensions with a comma; for example, "cbl,cob".

Description:

#### Translators

Specify and order translators for this language definition during the build.

**Assembler**  
IEWBLINK

---

### Translator

Name: **Assembler** Project Area: Rational Team Concert

Ignore

**Call Method**

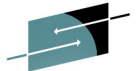
**Data Set Properties**

DD concatenations:

DD Name	Data Set Definitions
SYSLIB	CEE.SCEEMAC,SYS1.MACLIB,ASM.SASMMAC2,SYS1.MODGEN

DD allocations:

DD Name	Data Set Defi...	Member	Output Member Name	Keep	Mod	Output	Publish
SYSIN	<INPUT>	no		no	no	no	no
SYSLIN	OBJ	yes	Same as input	no	no	yes	no
SYSPRINT	LIST	yes		no	no	no	yes
SYSUT1	TEMPFILE	no		no	no	no	no



# Translator comparison to JCL using data set definitions

JCL Line	Corresponding data set definition name
COBOL EXEC PGM=IGYCRCTL,REGION=2048K,	COBOL Compiler
XX PARM=('EXIT(ADEXIT(ELAXMGUX)'), XX 'ADATA', XX 'LIB', XX 'TEST(NONE,SYM,SEP)', XX 'LIST', XX 'FLAG(I,I)')&CICS&DB2&COMP)	No DSD
XXSTEPLIB DD DISP=SHR, ... ...JCL - DISP=SHR,DSN=COBOL.V4R2.SIGYCOMP ...JCL - DISP=SHR,DSN=RDZ.V8R0M3.SFEKLOAD ...JCL - DISP=SHR,DSN=CICSTS.V4R1.CICS.SDFHLOAD ...JCL - DISP=SHR,DSN=DB2.DB40.SDSNLOAD	COBOL.SIGYCOMP WDZ.SFEKLOAD CICS.SDFHLOAD DB2.SDSNLOAD
COBOL.SYSLIB DD DISP=SHR, DSN=F057699.TEST.RTC.COPY	Copybooks
COBOL.SYSIN DD DISP=SHR, // DSN=F057699.TEST.RTC.COBOL(EPSCMORT)	<INPUT> represents the source file associated with the language definition being built
//COBOL.SYSLIN DD DSN= &&OBJ,SPACE=(TRK,(3,3)), // UNIT=SYSDA, DISP=(NEW,PASS) // DCB=(RECFM=FB,LRECL=256,BLKSIZE=2560)	Temporary file (object deck)
SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT2 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT4 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT5 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT6 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT7 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))	Temporary file

**Translator**  
Name: JKE COBOL compilation (CICS&DB2)

**General**  
Description:

**Call Method**  
 Called program  
Data set definition: JKE COBOL compiler Browse...  
Default options: EXIT(ADEXIT(ELAXMGUX)),ADATA,LIB,TEST(NONE,SYM,SEP),LIST,FLAG(I,I)

DD names list:

Maximum return code: 4

**Data Set Properties**  
DD concatenations:

DD Name	Data Set Definitions
SYSLIB	JKE Copybooks,JKE CICS.SDFHCOB
TASKLIB	JKE COBOL.SIGYCOMP,JKE CICS.SDFHLO...

DD allocations:

DD Name	Data Set Definition	Member	Keep	Output	Publish
SYSIN	<INPUT>	no	no	no	no
SYSLIN	JKE Temporary file (obje...	no	yes	no	no
DBRMLIB	JKE DBRM library	yes	no	yes	no
SYSPRINT	JKE Temporary file	no	no	no	yes
SYSADATA	JKE Temporary file	no	no	no	no
SYSXMLSD	JKE Temporary file	no	no	no	no
SYSUT1	JKE Temporary file	no	no	no	no
SYSUT2	JKE Temporary file	no	no	no	no
SYSUT3	JKE Temporary file	no	no	no	no
SYSUT4	JKE Temporary file	no	no	no	no

- Data Set Definitions

  - JKE Assembler
  - JKE BIND files
  - JKE BMS maps
  - JKE CEE.SCEELKED
  - JKE CICS.SDFHCOB
  - JKE CICS.SDFHLOAD
  - JKE CICS.SDFHMAC
  - JKE COBOL.SIGYCOMP
  - JKE COBOL compiler
  - JKE COBOL source codes
  - JKE Copybooks
  - JKE DB2.SDSNLOAD

# Setting up the RTC ISPF Client

- The Rational Team Concert ISPF Client is installed as part of the Build System Toolkit FMID (HRBT500 for v5.0)
- Consists of normal ISPF components : execs, panels, messages, skeletons and load modules
- Also has a Java daemon that handles communication to the RTC server
- A number of system programmer and RACF administrator activities need to be performed before the ISPF Client will work
- Running the SMP/E install will lay down the PDSEs and HFS components required

# SMP/E Installed components

```

Menu Options View Utilities Compilers Help
-----
MVS269 - Data Sets Matching JAZZ00.V5R0M1F0.GA.SBLZ*          Row 1 of 13
Command ==> _____ Scroll ==> CSR

Total Tracks:          311 non-x:          311 Data Sets:          13 non-x:          13
-----
Command - Enter "/" to select action                               Tracks %Used   XT
-----
JAZZ00.V5R0M1F0.GA.SBLZAUTH                                     10   60    1
JAZZ00.V5R0M1F0.GA.SBLZDBRM                                    2    50    1
JAZZ00.V5R0M1F0.GA.SBLZEXEC
JAZZ00.V5R0M1F0.GA.SBLZJCL
JAZZ00.V5R0M1F0.GA.SBLZLOAD
JAZZ00.V5R0M1F0.GA.SBLZMENP
JAZZ00.V5R0M1F0.GA.SBLZMENU
JAZZ00.V5R0M1F0.GA.SBLZMJPN
JAZZ00.V5R0M1F0.GA.SBLZPENP
JAZZ00.V5R0M1F0.GA.SBLZPENU
JAZZ00.V5R0M1F0.GA.SBLZPJPN
JAZZ00.V5R0M1F0.GA.SBLZSAMP
JAZZ00.V5R0M1F0.GA.SBLZSLIB

```

```

Menu Utilities View Options Help
-----
MVS269                               z/OS UNIX Directory List          Row 1 to 13 of 13
Command ==> _____ Scroll ==> CSR

Pathname . . : /var/rtc501GA/usr/lpp/jazz/v5.0.1
EUID . . . : 0
Command Filename                               Message                               Type Size      Perm
-----
.                                               .
..                                              ..
buildagent                                     Dir                               8192 755
buildsystem                                    Dir                               8192 755
ispfclient                                     Dir                               8192 755
license                                        Dir                               8192 755
properties                                     Dir                               8192 755
repotools                                      Dir                               8192 755
scmtools                                       Dir                               8192 755
searchengine                                  Dir                               8192 755
server                                         Dir                               8192 755
support                                        Dir                               8192 755
IBM                                             Dir                               8192 755

```



# Configuring RTC on z/OS

- In RTC v5.0.1 we are providing a configuration utility to ease the pain points of the installation tasks
- Configuration instructions are also provided in the online Knowledge Centre
  - For z/OS we have provided a checklist to make this easier  
[https://jazz.net/help-dev/clm/topic/com.ibm.jazz.install.doc/topics/t\\_checklist\\_zos.html](https://jazz.net/help-dev/clm/topic/com.ibm.jazz.install.doc/topics/t_checklist_zos.html)
  - In addition there is a printable PDF copy as we know how z/OS folk like the old fashioned ways:  
<http://www-01.ibm.com/support/docview.wss?uid=swg27041833>

# RTC Configuration Utility



New in  
RTC 5.0.1

- Will be released in RTC v5.0.1
- Provide a workflow based configuration, tailored to which components of RTC you plan to install on z/OS
- Provides an Installation Verification Process (IVP)

# Setting up the RTC ISPF Client

- As a checklist however the following tasks need to be performed
  - Run the BLZCPBTK job to create directories, copy config files and tailor them
    - In particular the **ispfdmn.conf**
  - Tailor and run RACF job BLZRACFT
    - Pay particular attention to the activation of the APPL and PTKTDATA classes
  - Create the ISPF daemon started tasks, by default:
    - BLZISPFD to start the daemon
    - BLZISPFS to cleanly stop the daemon
  - The **ispfdmn.conf** should already be configured sufficiently, but you may want to change some of the default configuration

# Setting up the RTC ISPF Client

- Additional system programmer tasks
  - Set one of the following
    - MAXASSIZE to 2GB in the **BPXPRMxx** member
    - ASSIZEMAX to 2GB in the OMVS segment for the ISPF Daemon started task userid
  - Make sure hlq.SBLZAUTH, which contains the BLZPASTK module, is in the linklist and APF authorised
  - Add BLZPASTK to the AUTHPGM list in **IKJTSOxx**, e.g.
    - AUTHPGM NAMES(IEBCOPY,BLZPASTK)
- Start the ISPF daemon

# RTC Configuration Utility



# Demo



# Using the RTC ISPF Client

- The ISPF Client is not a fully functioning RTC Client
  - It does not offer the planning and work item management that the Eclipse or Web Client offer
  - Concentrates on SCM operations that an ISPF developer needs to use to work on their code.
- The ISPF Client offers the ability to:
  - Log into an RTC repository
  - Load sand box data sets and HFS sandbox directories from the RTC repository
  - Edit loaded files
  - Check code back into the repository
  - Deliver code back to the Stream
  - Accept changes from the stream into your sandbox data sets
  - Kick off builds

# Using the ISPF Client - Login

```

Menu Help
-----
MVS269          Enter Password          Row 1 to 11 of 11
Command ==>    MVS269                    Scroll ==> CSR
               Password _
Enter new c    sting connections for
options

User ID      Repository URI
+
bgreen      https://mvs079.rtp.raleigh.ibm.com:9538/ccm/
dohertl     https://jazzdev02.torolab.ibm.com:9443/jazz02/
dohertl     https://mvs255.rtp.raleigh.ibm.com:9738/ccm/
dohertl     https://9.190.124.221:9443/ccm/
dohertl     https://9.143.99.226:9443/ccm
dohertl     https://mvs255.rtp.raleigh.ibm.com:9858/ccm/
L dohertl    https://jazzdev.torolab.ibm.com:9443/jazz/
jazz        https://linux155.rtp.raleigh.ibm.com:9443/ccm
rradclif    https://linux235.rtp.raleigh.ibm.com:9447/jazz/
shara30     https://
xavier      https://
  
```

```

Menu Help
-----
MVS269          Rational Team Concert Primary Option Menu
Option ==>
0 Preferences Terminal and user preferences      ***** Logged in *****
1 Connections Work with connections to source    User ID . . . : dohertl
2 Workspaces  Work with repository workspaces    Server . . .  : jazzdev.torol
3 Edit        Work with source data              Project . . . : Rational Team
4 Build       Work with build options             Release . . . : 5.0.1 RC1
X Exit        Exit client
  
```

# Using the ISPF Client – Load sandbox data sets

```

Load Repository Workspace
MVS269                               Row 1 to 5 of 5
Command ==> _____ Scroll ==> CSR

Data set prefix . . DOHERTL.V5DEV
(Fully qualified, no quotes)

z/OS UNIX directory _____ +

Command - Enter "/" to select action

Component to load:
___ Build zComponents
___ Build Modules HFS generated
___ Deployment zComponents
___ ISPF Client
___ Promotion zCompd

```

```

Menu Help
MVS269                               Repository Workspaces          Row 1 to 10 of 26
Command ==> _____ Scroll ==> CSR

Enter new repository workspace name to create or "/" against existing
repository workspace for options

Names                                Load location
Data set prefix    z/OS UNIX dir.

___ v5.0                                DOHERTL.V5DEV
___ zOS Integration v5 Eclipse
___ zOS Integration Tests Build Wo
___ zOS Integration 3.x Build Work

```



# Using the ISPF Client – Editing members

```

Menu  Options  Help
-----
MVS269                z/OS Data sets                Row 1 to 13 of 18
Command ==> _____ Scroll ==> CSR_

Command - Enter "/" to select action          SCM ON/OFF : ON_

Data set name                Member      Members
                             Pattern        under SCM
e_ DOHERTL.V5DEV.GML          BLZ@I*    Yes
__ DOHERTL.V5DEV.GMLINC      BLZ@A*    Yes
__ DOHERTL.V5DEV.GMLJPN      BLZHTO*   Yes
__ DOHERTL.V5DEV.LIBPAX      _____ Yes
__ DOHERTL.V5DEV.LINK        _____ Yes
__ DOHERTL.V5DEV.OBJ         _____
__ DOHERTL.V5DEV.SBLZEXEC    _____ Yes
__ DOHERTL.V5DEV

```

```

Menu  Options  Help
-----
MVS269                Source Control                Row 1 to 4 of 4
Command ==> _____ Scroll ==> CSR_

z/OS data set : DOHERTL.V5DEV.GML

Command - Enter "/" to select action

Name      SCM  Lock  Changed          ID
__ BLZ@ISPC          2014/06/03 09:40:25 DOHERTL
__ BLZ@ISPG          2014/04/09 12:48:58 DOHERTL
e_ BLZ@IVP            2014/05/15 23:27:04 DOHERTL
__ BLZ@IVPW          2014/06/19 03:46:43 DOHERTL
***** Bottom of data *****

```

# Using the ISPF Client – Builds

```

Menu Help
-----
MVS269                Builds                Row 1 to 11 of 11
Command ==> _____ Scroll ==> CSR

Filter zos*

Command - Enter "/" to select action

Build ID                Last Result
- zOS.integration       Completed
- zOS.integration.ant-EE Completed
- zOS.integration.tests Failed
- zOS.integration.ISPF  Completed
- zOS.integration.ISPF.407
- zOS.integration.SMPE
- zOS.integration.SMPE.407
- zOS.integration.3x
- zOS.integration.4x
- zOS.integration.402
- zOS.integration.407

```

```

Menu Help
-----
MVS269                Submit Build Request        Row 1 to 4 of 9
Command ==> _____ Scroll ==> CSR

S Submit Build

Build ID                zOS.integration.ISPF

/ Personal Build        = Override Build Agent Authentication

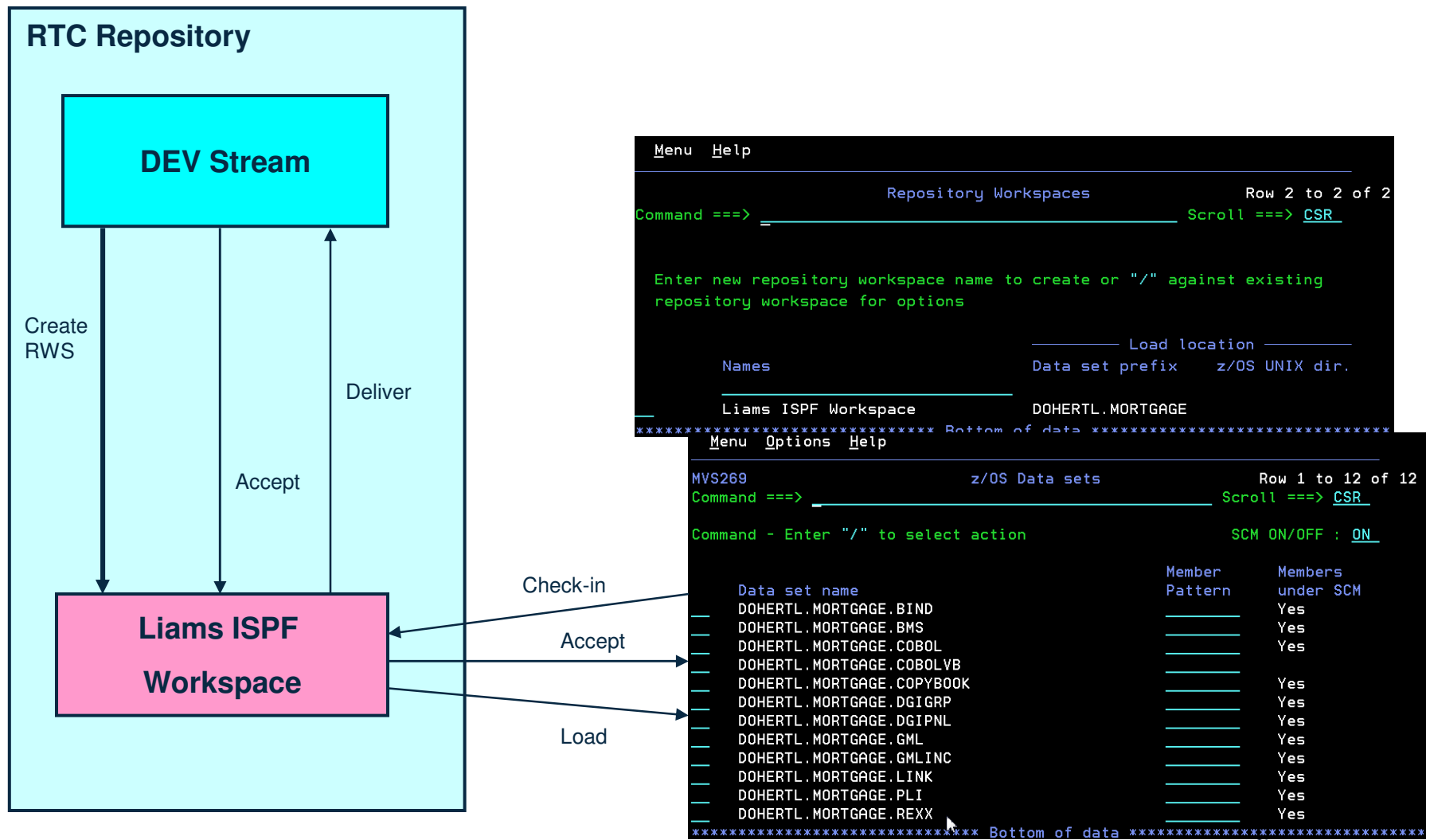
Enter new property name to create or "/" against existing property for
options

----- Build Properties -----

Name                    Value
-----+-----+
- build.NL                true
- buildzOSSamples         true
- buildISPF                true
- buildVersion             V501

```

# Using the RTC ISPF Client



# Using the RTC ISPF Client



# Demo



# Less common stuff stored in RTC

- SDF-II objects
  - <http://www.ibm.com/developerworks/rational/library/screen-definition-ii-rational-team-concert/index.html>
- ISPF DTL
  - <http://www.ibm.com/developerworks/rational/library/configure-rational-team-concert-build-dtl-components>
- Other usefull stuff...
  - <https://www.ibm.com/developerworks/community/blogs/LiamDoherty/?lang=en>

# Additional Resources

- Jazz.net
  - <https://jazz.net/library/>
    - Articles, videos, tips, documentation, and more
  - <https://jazz.net/library/#type=video&project=rational-team-concert>
    - Videos on various RTC features. Just search for keywords

## Some extra stuff to read at home!

- Setting up builds in RTC

# Setting up Builds in RTC

- In order to build our programs we need to configure a number of things
  - Build Agent configuration
  - Start Build Agent on z/OS
  - Build Engine to point to the Build Agent in the RTC Repository
  - Build Definition
    - Including a Build workspace



# Setting up Builds in RTC

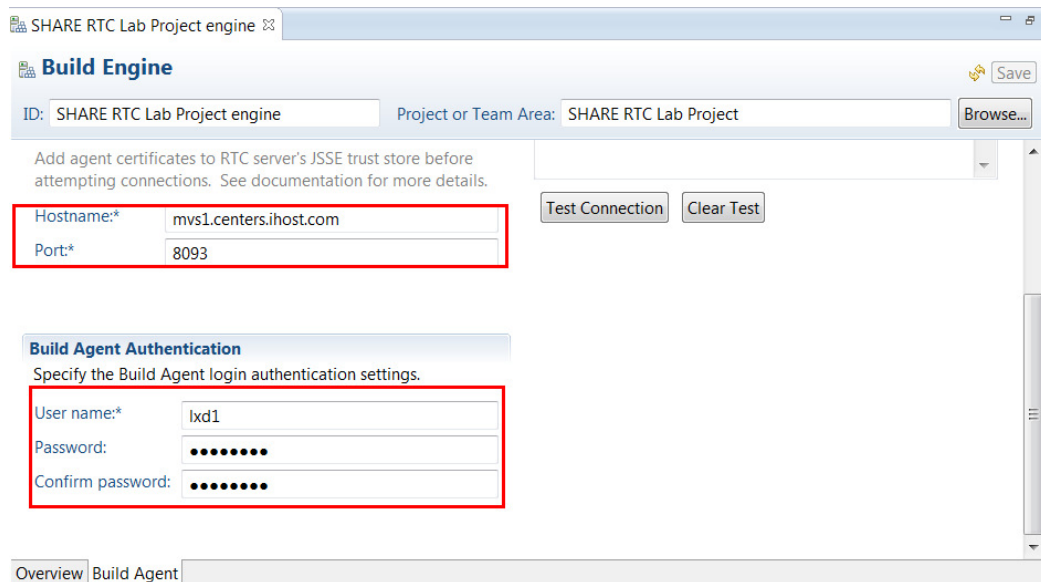
- As a checklist however the following tasks need to be performed, you may have done these already as part of the ISPF Client set-up
  - Run the BLZCPBTK job to create directories, copy config files and tailor them
    - In particular the **startbfa.sh**
    - and **bfagent.conf**
  - Tailor and run RACF job **BLZRACFT**
  - Create a password file using job **BLZBPASS**
  - The **startbfa.sh** and **bfagent.conf** will be partially configured, but you will need to change some of the default configuration:
    - port number in the **bfagent.conf**
    - Build userid and location of the password file in **startbfa.sh**
  - Create and start the Build Forge Agent started task, by default:
    - **BLZBFA**
    - Alternatively start the agent directly from the HFS

# Setting up Builds in RTC

- Gotcha...
  - If the userid that started the agent on z/OS is not **UID=0** then...
    - In **bfagent.conf** you will need to modify the **magic\_login** directive
      - Navigate to the **bfagent** directory where the product is installed: (*/usr/lpp/jazz/v4.0.6/buildagent*) and issue command: **bfagent -P <password>**
      - Cut/paste the returned password into the **magic\_login** directive
      - Remember to enter the correct userid, which must be the TSO userid that you specify on the build engine screen:  
**magic\_login lxd1:8d7d38d8430b164572f36c5b2e91ba8df1cbbf9f363258c6**

# Setting up Builds in RTC

- Create a build engine through the Eclipse interface
  - Specify the machine where the agent is running
  - Specify the port it is running on
  - Specify a TSO userid/password on that machine



The screenshot shows the Eclipse IDE configuration window for a 'Build Engine'. The window title is 'SHARE RTC Lab Project engine'. The 'Build Engine' section contains the following fields and controls:

- ID:** SHARE RTC Lab Project engine
- Project or Team Area:** SHARE RTC Lab Project
- Save** button
- Browse...** button
- Instruction: Add agent certificates to RTC server's JSSE trust store before attempting connections. See documentation for more details.
- Hostname:** mvs1.centers.ihost.com
- Port:** 8093
- Test Connection** and **Clear Test** buttons

The **Build Agent Authentication** section contains the following fields:

- User name:** lxd1
- Password:** [masked with dots]
- Confirm password:** [masked with dots]

At the bottom, there are tabs for **Overview** and **Build Agent**.

# Setting up Builds in RTC

- Create a build definition through the Eclipse interface
  - Specify the build agent to use
  - Contains the build characteristics
  - Repository workspace that flows to team stream containing the source code
  - Repository workspace must be readable by the build user
  - What do I want to build? Whole repository workspace or subset of programs
  - Language definitions to be built
  - Sandbox location (PDS HLQ)

# Setting up Builds in RTC

SHARE RTC Lab Project build

**Build Definition** Save

ID:  Project or Team Area:  Browse...

**Build Workspace**  
Specify the repository workspace from which to build. It should have the stream you want to build as its flow target.  
Workspace:\*  Select... Create...

**Load Options**  
Specify file extraction details. Properties can be referenced using \${propertyName}.

Load directory:\*

Delete directory before loading

Resource prefix:\*

Load workspace to the load directory at the beginning of the build.  
 Load workspace to the resource with the resource prefix at the beginning of the build.  
 Create folders for components

Overview | Schedule | Properties | Output Publishing | Jazz Source Control | z/OS Dependency Build

# RTC z/OS builds : How it all hangs together

