#SHAREorg

SHARE

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

Liam Doherty
IBM Corporation

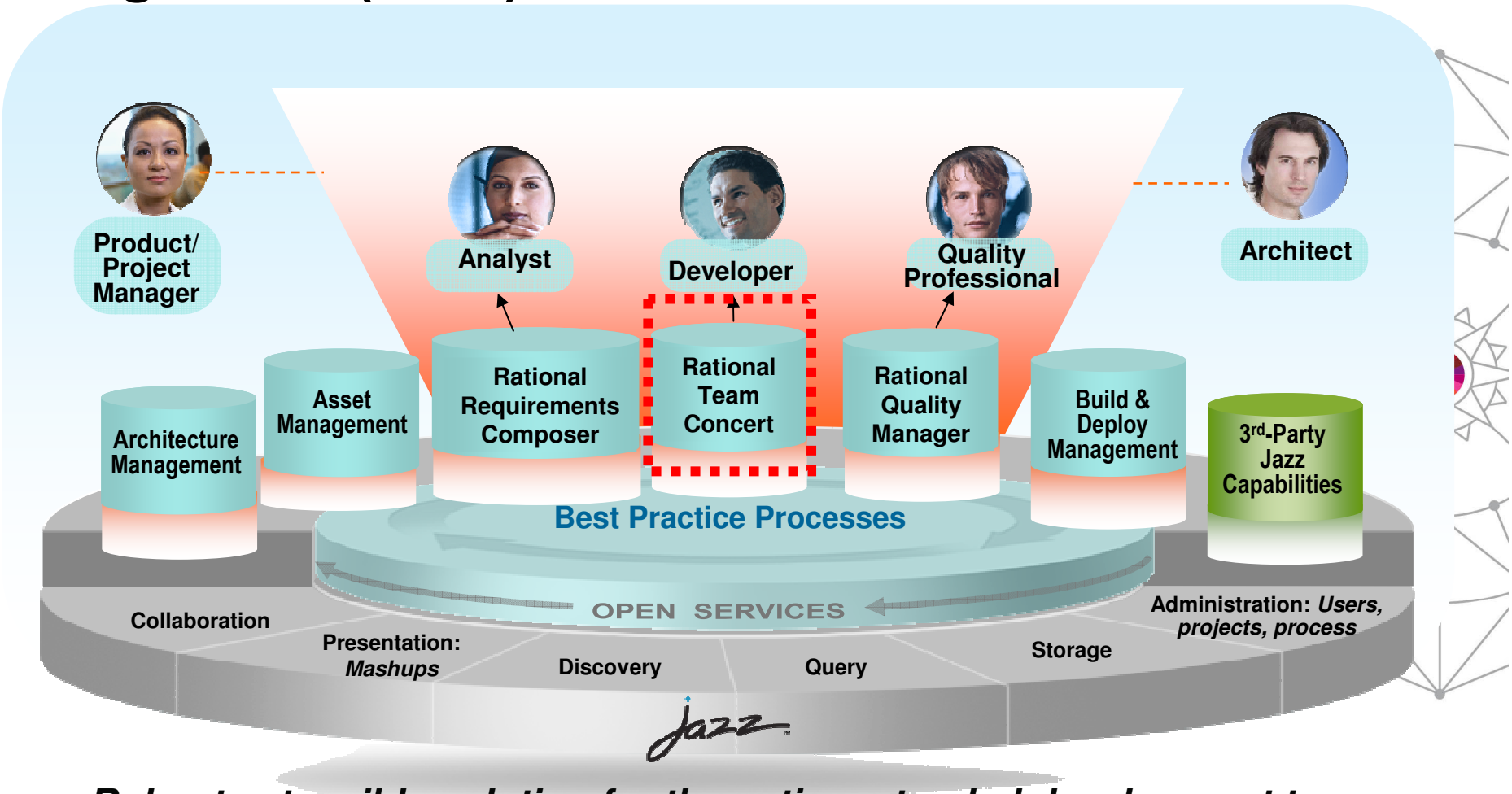
Wednesday March 12th, 2014
Session 14751

SHARE in Anaheim

# 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
- Setting up build engines/agents and build definitions

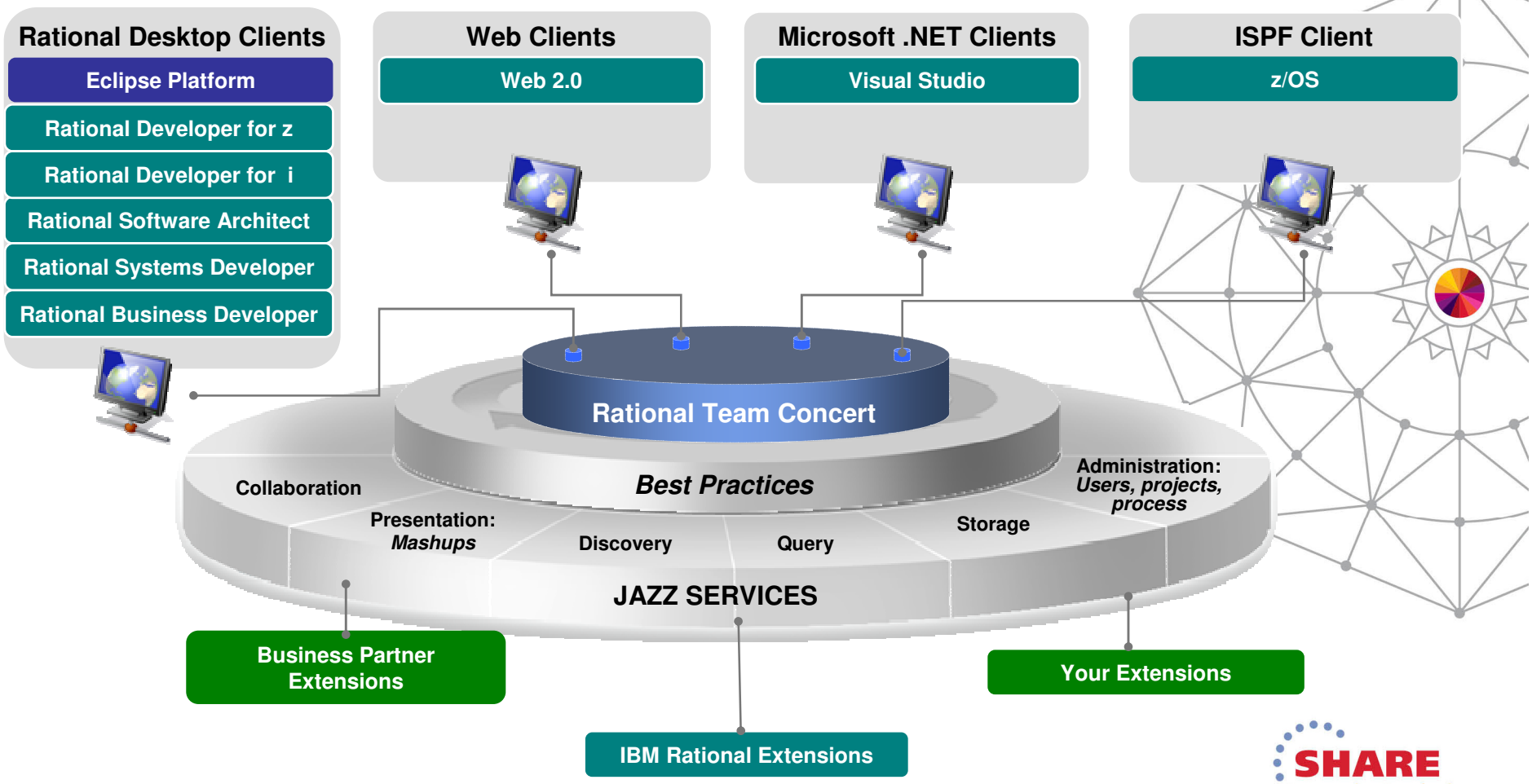
# IBM Rational Collaborative Lifecycle Management (CLM)




***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: An Overview

## Planning

- Integrated release/iteration planning
- Effort estimation & progress tracking taskboards
- Out of the box process templates: formal or agile

## Project Transparency

- Customizable web based dashboards
- Real time metrics and reports
- Project milestone tracking and status

## SCM

- Component based SCM enables reuse across projects
- Change set based for easy addition or removal of features
- Server-based sandboxes
- Can also work with SVN, Git, ClearCase or Synergy

## Work Items

- Defects, enhancements and conversations
- View and share query results
- Support for approvals and discussions
- Query editor interface
- ClearQuest or Synergy Bridge

## Build

- Automated work item and change set traceability
- Build definitions for team and personal builds
- Local or remote build servers
- Multi-level continuous integration
- Integration with Build Forge

## Jazz Team Server

- Single structure for project related artifacts
- World-class team on-boarding / offboarding including team membership, sub-teams and project inheritance
- Role-based operational control for flexible definition of process and capabilities
- Team advisor for defining / refining “rules” and enabling continuous improvement
- Process enactment and enforcement
- In-context collaboration enables team members to communicate in context of their work

# 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

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

# Rational Team Concert terminology (cont)

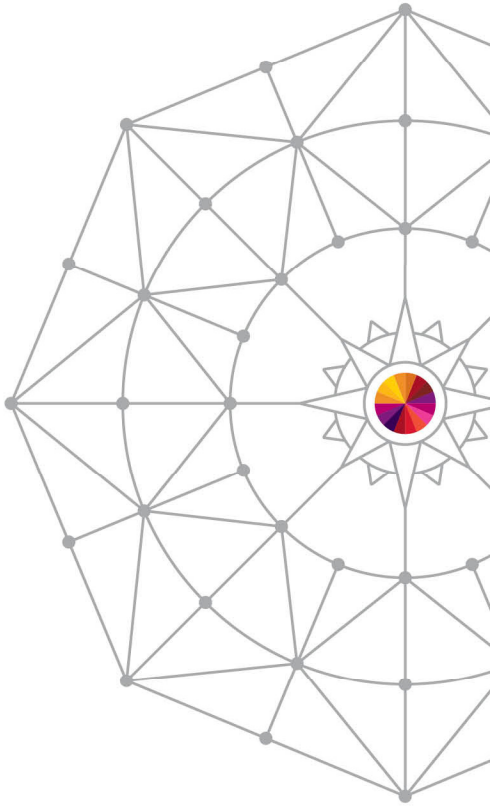
| | | |
|---|---|---|
| **Load** | Action that copies selected files and folders from the repository workspace to the sandbox (eclipse workspace or MVS data sets) | |
| **Accept** | Action that allows for synching the repository workspace reference with changes delivered to the stream by other developers<br>Load of the accepted changes into the sandbox is automatically performed<br>Note – you can also accept change sets from a WI | |
| **Check-in** | Action that allows to save local changes into the repository workspace, within a Change Set | |
| **Deliver** | 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 ISPF
- **Some functions are only offered through the Eclipse interface**
  - Enterprise Extensions definitions
  - So even through we are going to use the ISPF Client for source control, some admin 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

Work Items - com.ibm.teamz.ispf.client/zOSsrc/SOURCE/BLZACTIV.asm - Rational Team Concert

File Edit Navigate Search Project Run Window Help

Quick Access | Work Items | Data | Remote System Explorer

Team ... | Team ... | My Work | Team ...

My Filter (220 of 339 areas selected)

- Repository Connections
  - dohertl@jazzdev.torolab.ibm.com
  - dohertl@localhost (CRJAZ6024E Cannot log into the s
  - rtcadmin@mvs1.centers.ihost.com
- Jazz Collaborative ALM [jazzdev.torolab.ibm.com]
- Jazz Support (Private) [jazzdev.torolab.ibm.com]
- PMC (Private) [jazzdev.torolab.ibm.com]
- Rational Team Concert [jazzdev.torolab.ibm.com]
  - Builds
  - Enterprise Extensions
  - Plans
  - Reports
  - Source Control
  - Work Items
- Debug
- Favorites
- Feeds
- My Repository Workspaces
- My Team Areas
- Work Item History

BLZACTIV.asm

```
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,MOVMNAME           .Move in module name
000082
000083 *
000084 *        Use CSVQUERY to see if module has been used
000085 *
000086
000087          CSVQUERY INFPNAME=MODNAME,SEARCH=JPA,MF=(E,CSVQE)
```

Work Items | Team Advisor | Builds | Pending Changes | Navigator | History

- com.ibm.team.enterprise.zos.build.checkaccess (zOS Integration V5 Eclipse - Build Modules HFS generated)
- com.ibm.team.enterprise.zos.build.libRTCzMVS (zOS Integration V5 Eclipse - Build Modules HFS generated)
- com.ibm.team.enterprise.zos.build.linkeditparser (zOS Integration V5 Eclipse - Build Modules HFS generated)
- com.ibm.team.enterprise.zos.ispfclient.impersonation (zOS Integration V5 Eclipse - Build Modules HFS generated)
- com.ibm.teamz.ispf.client (zOS Integration V5 Eclipse - ISPF Client)
  - .settings
  - zOSsrc
    - 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

Work Items - com.ibm.teamz.ispf.client/zOSsrc/SOURCE/BLZACTIV.asm - Rational Team Concert

File Edit Navigate Search Project Run Window Help

Quick Access | Work Items | Data | Remote System Explorer

Team Artifacts | Team Dashboa... | My Work | Team Organiz...

My Filter (220 of 339 areas selected)

- WorkItem 3.X (Tracking and Planning 3.x Maintenance)
- WorkItem 4.X (Tracking and Planning 4.x)
- zOS Integration (Enterprise Extensions) — Stream
  - Auto Test (1: Initial Baseline)
  - Build Modules HFS generated (13: zOS.integration.HFS.component
  - Build zComponents (491: zOS.integration.dependency_20130904-1
  - Deployment zComponents (135: zOS.integration_20130826-103304
  - ISPF Client (565: zOS.integration_20130905-1359270786) — Component
  - Promotion zComponents (61: zOS.integration.dependency_201308
  - zOS releng (63: zOS.integration.dependency_20130904-154322014
- zOS Integration 3.0.0.x (Enterprise Extensions)
- zOS Integration 3.x (Enterprise Extensions)
- zOS Integration 4.0.2 (Enterprise Extensions)
- zOS Integration 4.0.5 (Enterprise Extensions)
- zOS Integration 4.0.6 (Enterprise Extensions)
- zOS Integration 4.x (Enterprise Extensions 4.x Maintenance)
- zOS Integration Run Team (EE Run Team)
- zOS Integration SMPE (Enterprise Extensions)
- zOS.integration.SMPE 4.0.6 (Enterprise Extensions)
- Work Items
- Debug
- Favorites
- Feeds
- My Repository Workspaces
- My Team Areas
- Work Item History

BLZACTIV.asm

```
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 parame
000078          BCTR  R15,R0                 .Decrement for "EX" instr
000079          MVC   MODNAME,BLANKS         .Init module name to blan
000080          LA    R14,MODNAME
000081          EX    R15,MOVMNAME           .Move in module name
000082
000083 *
000084 *        Use CSVQUERY to see if module has been used
000085 *
000086
```

Work Ite... | Team A... | Builds | Pending... | Navigator | History | Reposit...

Stream zOS Integration - ISPF Client

- 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
- com.ibm.teamz.ispf.client.tests.rft

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

Work Items | Team Advisor | Builds | Pending Changes | Navigator | History

- zOS.deployment (zOS Integration V5 Eclipse - Deployment zComponents)
  - .zOSbin
  - zOSsrc
    - GML
    - LINK
    - SOURCE
      - BLZBKPZP.rex
      - BLZDEPZP.rex
      - BLZPKGZP.rex

zComponent Project

**Properties for SOURCE**

type filter text
- Resource
- Enterprise Extensions
- Jazz Source Control
- Local DBCS Options
- Property Group

**Enterprise Extensions**

Data Set Definition: Source | Browse... | Clear

**Properties for BLZBKPZP.rex**

type filter text
- Resource
- Enterprise Extensions
- Jazz Source Control
- Local DBCS Options

**Enterprise Extensions**

Language definition: REXX Compile (Rational Team Concert) | Browse... | Clear

Always load this file during dependency build

Variables

Define variables and their values to be used by this file.

| Name | Value |
|---|---|

Add...

**Properties for BLZBKPZP.rex**

type filter text
- Resource
- Enterprise Extensions
- Jazz Source Control
- Local DBCS Options
- Property Group
- Run/Debug Settings

**Jazz Source Control**

File Properties

See File Property Preferences for a complete listing of file property mappings.

Executable: false

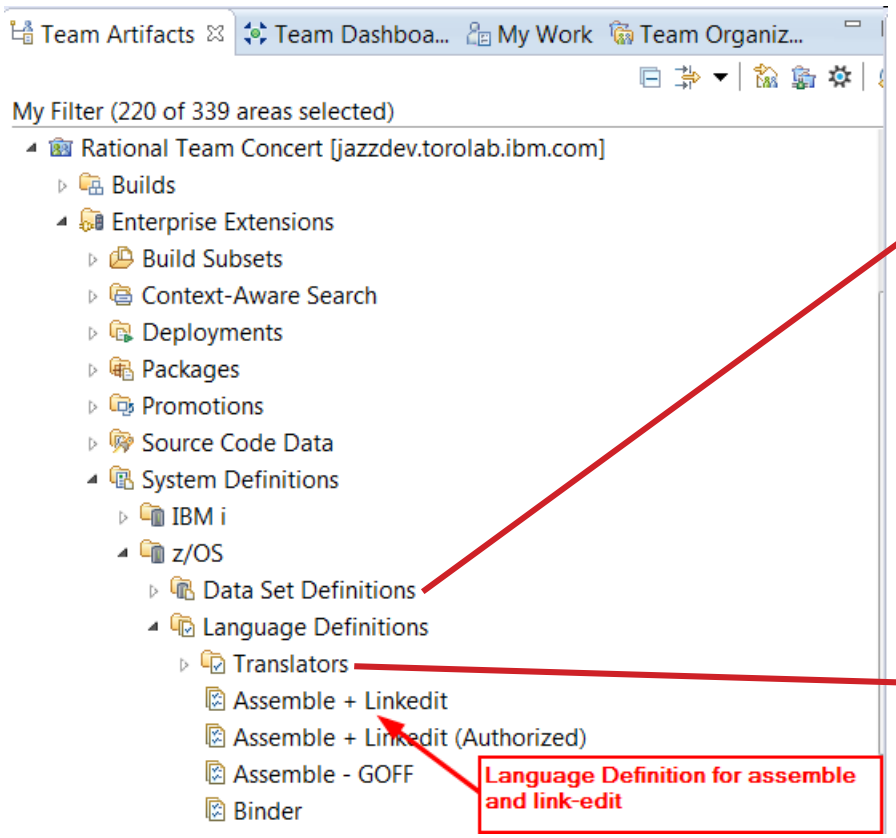
MIME Type: application/unknown

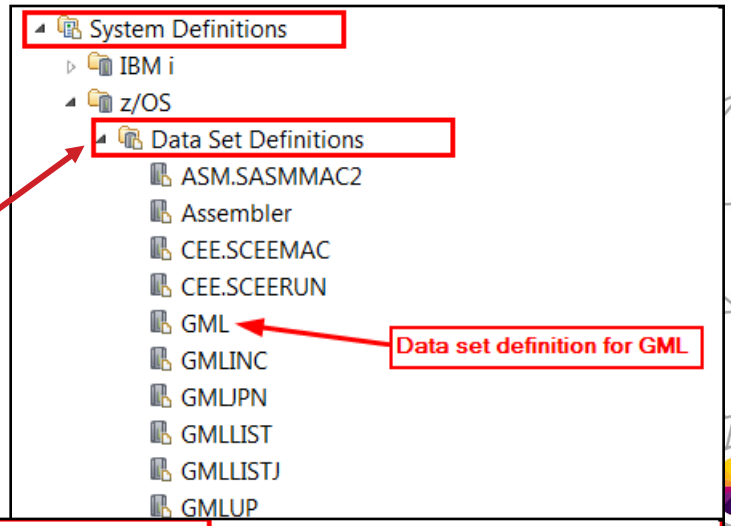
Line Delimiter: Platform

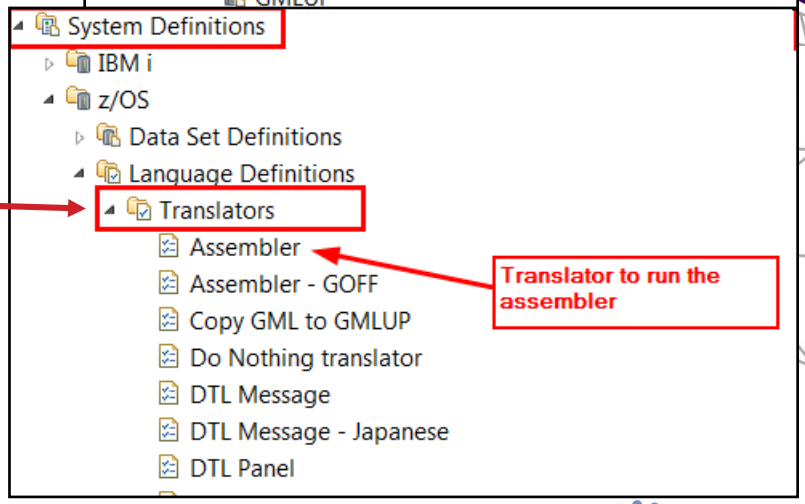
# Setting up Enterprise Extensions System Definitions

- **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



# 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**

Save

Name: Source

Project Area: Rational Team Concert

**General**

Description: Source

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: SOURCE

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: | Cylinders |
| Primary quantity: | 10 |
| Secondary quantity: | 5 |
| Directory blocks: | 0 |
| Record format: | FB |
| Record length: | 80 |
| Block size: | 27920 |
| Data set type: | Library(PDSE) |

# Language Definitions and Translators

- Language definitions are required for source to load
  - Define a dummy langauge 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 defir order during the build.

Assembler
IEWBLINK

## Translator

Name: Assembler  Project Area: Rational Team Concert

**Call Method**

**Data Set Properties**

DD concatenations:

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

Add... Edit... Remove

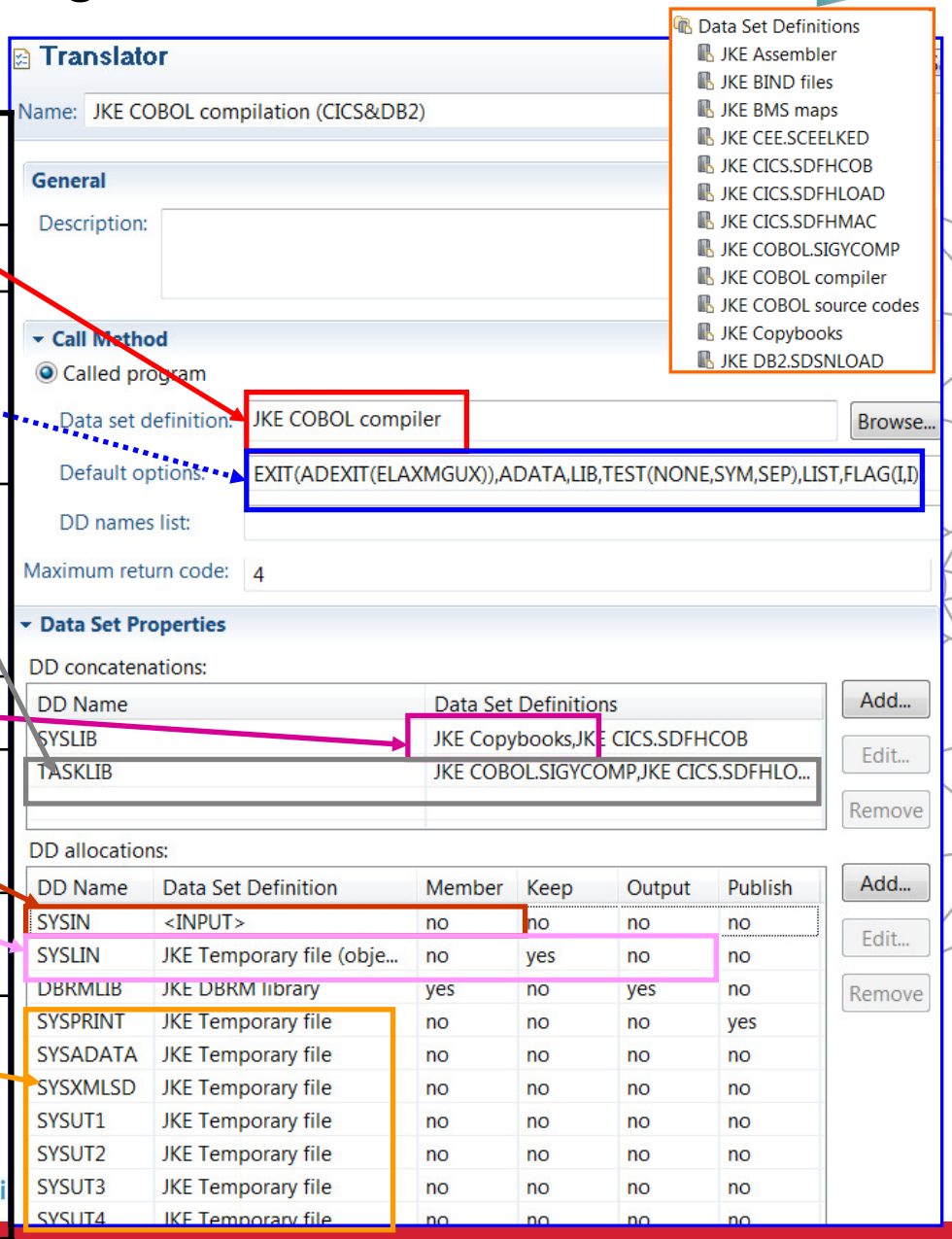
DD allocations:

| 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 |

Add... Edit... Remove

# 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))',<br>XX 'ADATA',<br>XX 'LIB',<br>XX 'TEST(NONE,SYM,SEP)',<br>XX 'LIST',<br>XX 'FLAG(I,I)'&CICS&DB2&COMP) | No DSD |
| XXSTEPLIB DD DISP=SHR,<br>...<br>..JCL - DISP=SHR,DSN=COBOL.V4R2.SIGYCOMP<br>...JCL- DISP=SHR,DSN=RDZ.V8R0M3.SFEKLOAD<br>...JCL- DISP=SHR,DSN=CICSTS.V4R1.CICS.SDFHLOAD<br>...JCL- DISP=SHR,DSN=DB2.DB40.SDSNLOAD | COBOL.SIGYCOMP<br>WDZ.SFEKLOAD<br>CICS.SDFHLOAD<br>DB2.SDSNLOAD |
| COBOL.SYSLIB DD DISP=SHR,<br>DSN=F057699.TEST.RTC.COPY | Copybooks |
| COBOL.SYSIN DD DISP=SHR,<br>// 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)),<br>// UNIT=SYSDA, DISP=(NEW,PASS)<br>// DCB=(RECFM=FB,LRECL=256,BLKSIZE=2560) | Temporary file (object deck) |
| SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))<br>SYSUT2 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))<br>SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))<br>SYSUT4 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))<br>SYSUT5 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))<br>SYSUT6 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))<br>SYSUT7 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) | Temporary file |



# Setting up the RTC ISPF Client

- The Rational Team Concert ISPF Client is installed as part of the Build System Toolkit FMID (HRBT406)
- Consists of normal ISPF components, panels, messages, 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

# Installed components

```
  Menu   Options   View   Utilities   Compilers   Help
MVS269 - Data Sets Matching JAZZ00.V4R0M6F0.SBLZ*
Command ===>

Total Tracks:        189 non-x:        189 Data Set
-----------------------------------------------------
Command - Enter "/" to select action
-----------------------------------------------------
         JAZZ00.V4R0M6F0.SBLZAUTH
         JAZZ00.V4R0M6F0.SBLZDBRM
         JAZZ00.V4R0M6F0.SBLZEXEC
         JAZZ00.V4R0M6F0.SBLZJCL
         JAZZ00.V4R0M6F0.SBLZLOAD
         JAZZ00.V4R0M6F0.SBLZMENP
         JAZZ00.V4R0M6F0.SBLZMENU
         JAZZ00.V4R0M6F0.SBLZMJPN
         JAZZ00.V4R0M6F0.SBLZPENP
         JAZZ00.V4R0M6F0.SBLZPENU
         JAZZ00.V4R0M6F0.SBLZPJPN
         JAZZ00.V4R0M6F0.SBLZSAMP
**********
```

```
  Menu   Utilities   View   Options   Help
MVS269
Command ===>

Pathname . : /var/rtc406/usr/lpp/jazz/v4.0.6
EUID . . . : 0
Command   Filename                          Message          Type
-------------------------------------------------------------------
_________ .                                                  Dir
_________ ..                                                 Dir
_________ buildagent                                         Dir
_________ buildsystem                                        Dir
_________ ispfclient                                         Dir
_________ license                                            Dir
_________ properties                                         Dir
_________ repotools                                          Dir
_________ scmtools                                           Dir
_________ searchengine                                       Dir
_________ server                                             Dir
_________ support                                            Dir
_________ IBM                                                Dir
*******************************************************************
```

# Setting up the RTC ISPF Client

- Follow the configuration instructions in the online infocenter
  - For z/OS we have provided a checklist to make this easier

    https://jazz.net/help-dev/clm/index.jsp?re=1&topic=/com.ibm.jazz.install.doc/topics/t_checklist_zos.html&scope=null
  - In addition there is a printable PDF copy as we know how z/OS folk like the old fashoned ways:

    http://www-01.ibm.com/support/docview.wss?uid=swg27041016
- In RTC v5.0 we hope to provide a config utility to ease the pain of the installation tasks
  - More of that later

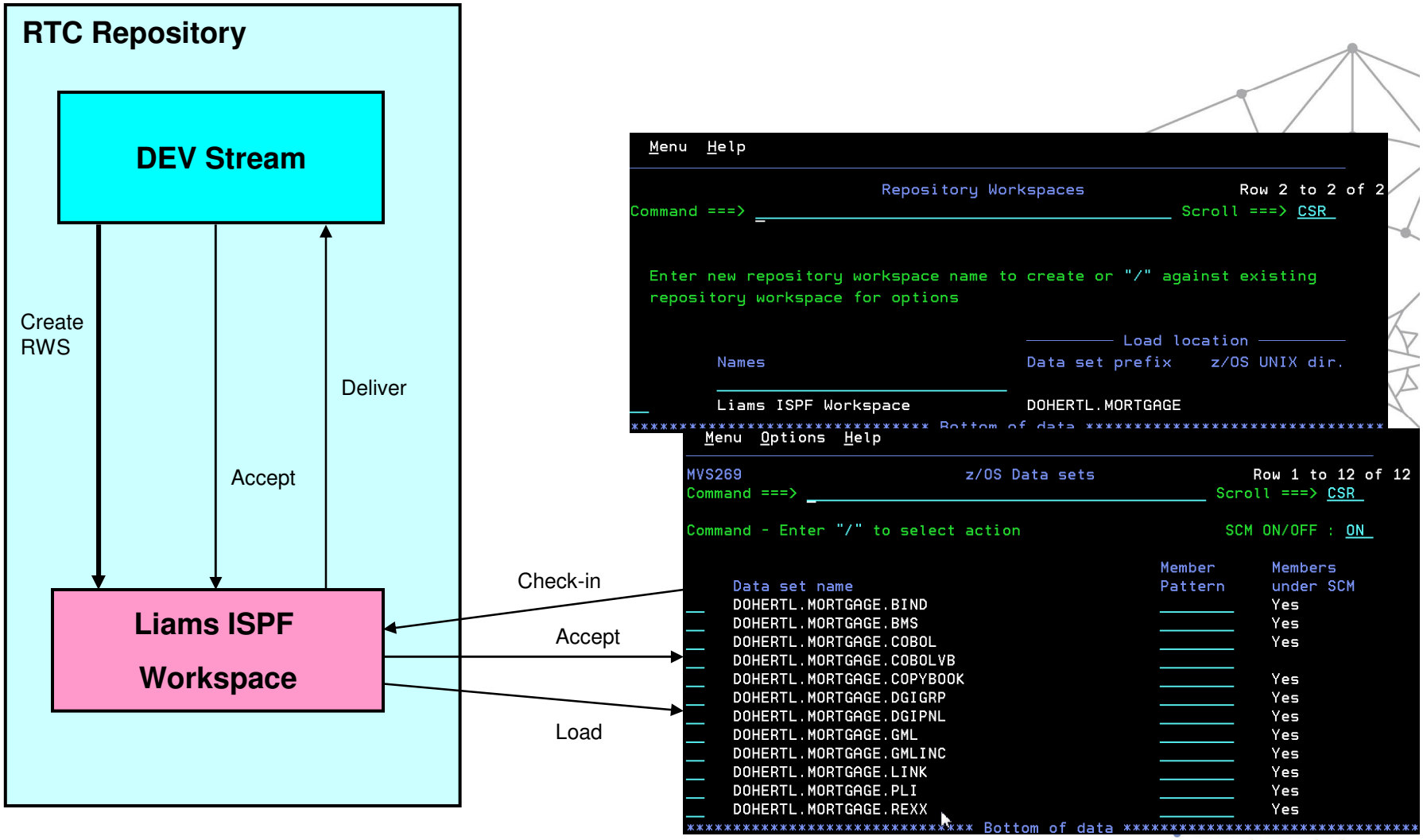
# Setting up the RTC ISPF Client

- 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

# Using the RTC ISPF Client



# Using the RTC ISPF Client

# RTC Configuration Utility

New in RTC 5.0

- Will be offered as a technical preview in RTC v5.0
- Provide a workflow based configuration, tailored to which components of RTC you are installing on z/OS
- Provide an Installation Verification Process (IVP)

Demo

# 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

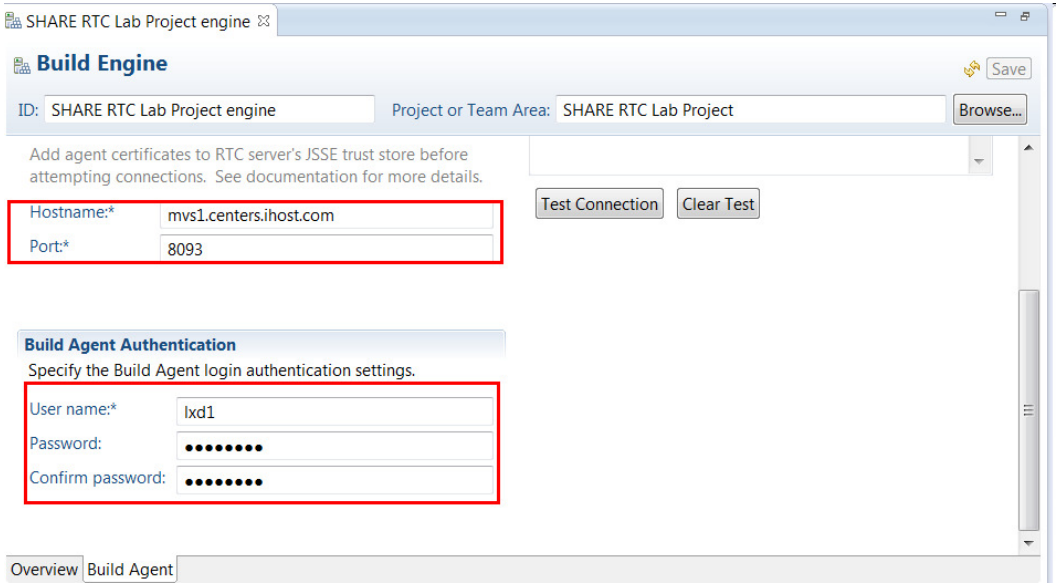
- As a checklist however the following tasks need to be performed, you may have done these already as part of the ISPF Client set-up
  - Run the BLZCPBTK job to create directories, copy config files and tailor them
    - In particular the **startbfa.sh**
    - and **bfagent.conf**
  - Tailor and run 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



# 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: SHARE RTC Lab Project build

Project or Team Area: SHARE RTC Lab Project

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:* SHARE RTC Lab Project Build Workspace

Select... Create...

**Load Options**

Specify file extraction details. Properties can be referenced using ${propertyName}.

Load directory:* /shareuser/lxd1/RTCLabBuild

☑ Delete directory before loading

Resource prefix:* LXD1.RTCBUILD

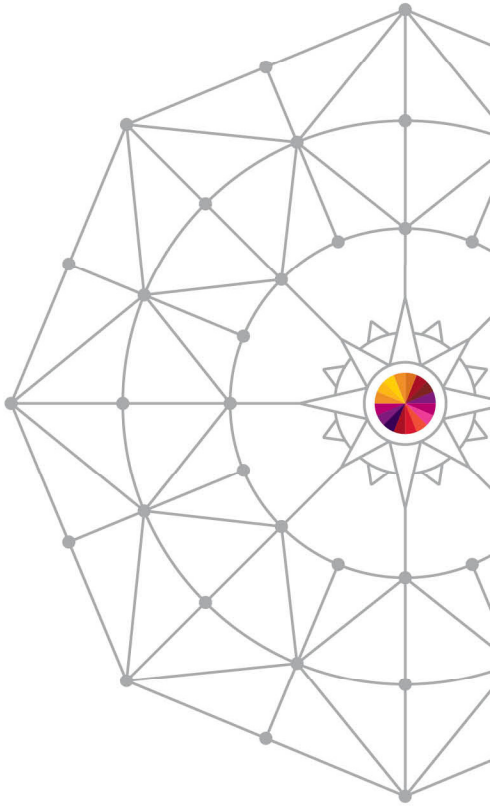
☑ Load workspace to the load directory at the beginning of the build.

☐ Load workspace to the resource with the resource prefix at the beginning of the build.

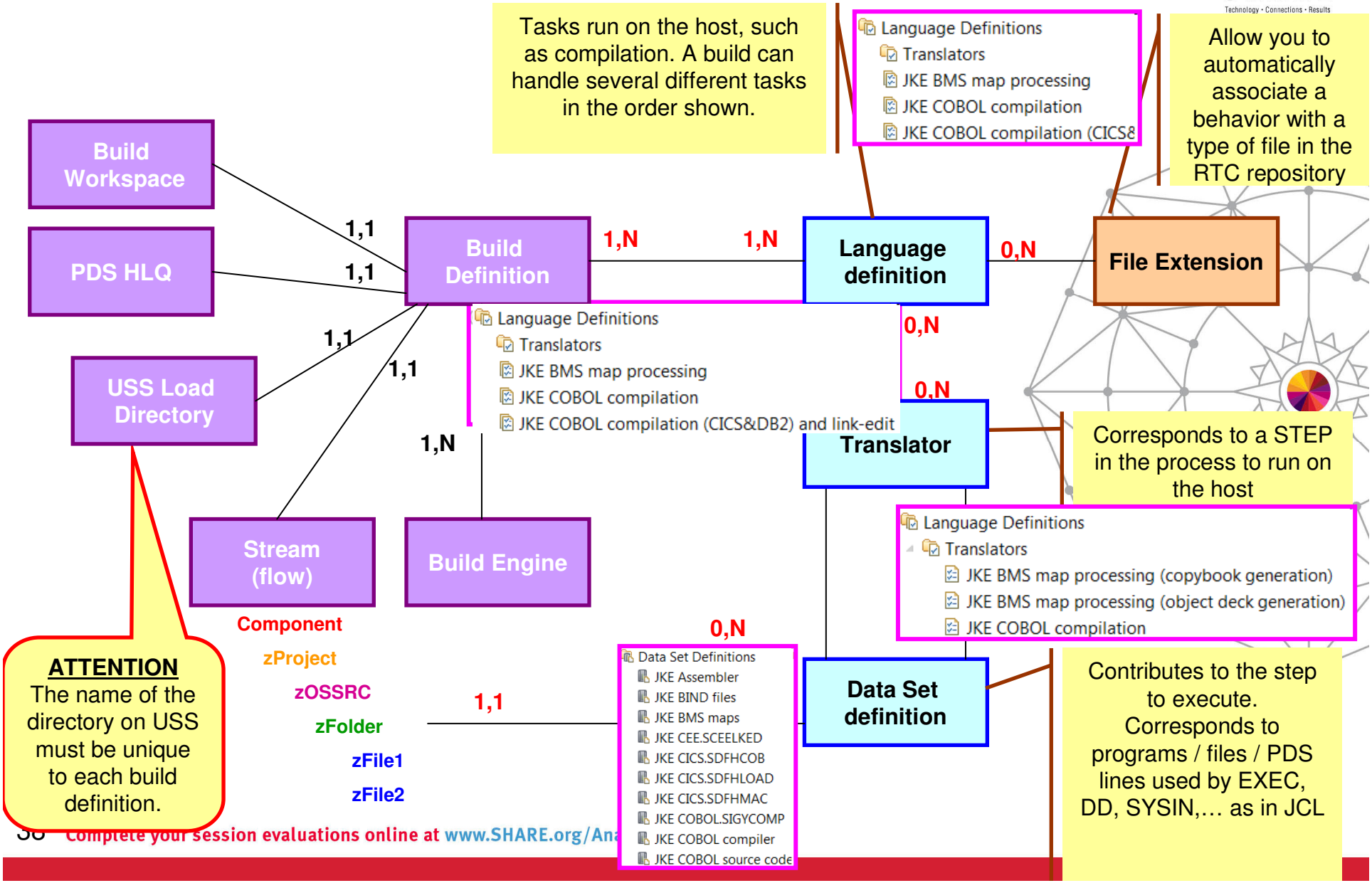
☐ Create folders for components

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

# Using the builds in RTC

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



# 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