t	#SHAREorg



Migrating to Rational Team Concert from SCLM and Other z/OS SCMs

Liam Doherty IBM Corporation

Thursday February 7th, 2013 Session 12531





Agenda



- Planning your Rational Team Concert solution
- Installing and Configuring the Rational Team Concert server and components
- Migrating your source code to Rational Team Concert
- Migrating your build to Rational Team Concert
- Additional Considerations



IBM Rational Collaborative Lifecycle Management (CLM)





Robust extensible solution for the entire extended development team



• • • in San Francisco 2013

SHARE

Rational Team Concert (RTC): An open, extensible architecture



Supporting a broad range of desktop clients, IDEs and languages







Rational Team Concert: An Overview



SHAR **Project Transparency** Planning Customizable web based dashboards Integrated release/iteration planning Effort estimation & progress tracking taskboards Real time metrics and reports Out of the box process templates: formal or agile Project milestone tracking and status SCM Work Items **Build** Component based SCM Defects, enhancements Automated work item and enables reuse across projects change set traceability and conversations Build definitions for team View and share query results Change set based for easy and personal builds addition or removal of features Support for approvals and Local or remote build servers discussions Server-based sandboxes Multi-level continuous Query editor interface integration Can also work with SVN. Git. ClearQuest or Synergy Bridge ClearCase or Synergy Integration with Build Forge **Jazz Team Server** Team advisor for defining / refining "rules" Single structure for project related artifacts and enabling continuous improvement World-class team on-boarding / offboarding

- Process enactment and enforcement
 - In-context collaboration enables team members to communicate in context of their work



including team membership, sub-teams and

project inheritance

definition of process and capabilities

Role-based operational control for flexible

••••• in San Francisco 2013

What is Rational team Concert?



- 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
- Migrating your existing SCM to RTC is only part of the job
- Migration gives you the chance to review your current process to see how RTC or the full CLM solution can help integrate all your processes into a single tool



- The Rational Team Concert Project Area
- Rational Team Concert Definitions
- How will you organize your source code?
- Where are you going to host your server and repository?
- How will you build your applications?
- Solution adoption





• The Project Area

- Top-level container in which the work will happen
- Contains data administration
 - Members
 - Roles
 - Team Members (Developers), Contributors (Interested parties), Administrators or Stake Holders
 - Assignment of roles to members
 - Permissions
 - Process
 - WorkItem Types
 - Workflows
 - Control over the various operations (preconditions, follow-up actions)
 - Ex: A changeset must be associated with a WorkItem (WI)
 - Streams
 - Definition of Builds







- The Project Area
 - Creation facilitated by using the concepts as follows:
 - The Process Template
 - Facilitates the creation of a Project Area that suits specific needs by initializing with a default configuration
 - Begin by using an out of the box process template that best suits your needs, and try to tailor only when necessary
 - Concept of inheritance is possible between Project Areas
 - Reuses the configuration of the master Project Area
 - Project Areas can directly map to a software project or may represent functional areas in your organization
 - Create team areas to define a hierarchy of teams working on that project
 - Team areas inherit the process from the Project Area
 - They manage team membership, role assignments and team artifacts such that different teams can have slightly differing process
 - Create team areas for groups of people who need to plan and work together





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 Used to control access rights, facilitate sharing and reuse Theoretical limit: 50000 files Recommended: 500 – 1000 files / component	
Repository Workspace	Workspace for 1 user synchronized with a Stream and the "Sandbox" Situated on the RTC server	
Sandbox	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	
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 Associated with change sets created by the developer. Automatically and dynamically populate plans and reports	
Baseline	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	
Snapshot	Collection taken of all component baselines for a stream or repository workspace capturing an interesting point in time	





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 Load of the accepted changes into the sandbox is automatically performed 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	





- How will you organize your source code?
 - RTC is a Stream/Component based source control management system
 - You need to define how your development assets will be organized, and how code will be shared and flow in the repository.
 - In RTC, artifacts are organized in *components*, which are the fundamental logical structure for organizing assets in SCM.
 - You have to define how your applications can be logically divided into these *components*.
 - Components are also used to control access to code, reuse code between teams/applications, or build subsets of the system.





Analyze your current setup

- Determine how applications are currently stored in your SCM
- Check the number of source files tied to each application
 - Is it granular enough? for RTC components should have a maximum of 500-1000 members
- If you have one big application that has grown historically you are going to need to break it down





- Components
 - Which logical units make up the applications (components)?
 - Put related artifacts or projects together so components make sense from code reuse, application build operations and team sharing perspective
 - What are the common source elements used across several applications/modules?
 - Define components to be reused across applications, so they can be maintained by certain teams, or to be shared for all teams within a project area.
 - your development teams will work on a set of components of which they are responsible.
 - When structuring the components along with architectural details bear also in mind the organizational structure that will support it.





- Streams
 - A stream is a collection of one or more components used to organize the work of a project, coordinate the integration activities, and capture important configuration points of your source code (promotion levels, releases, integration builds ...).
 - How many streams do you need to define?
 - Start with the basic streams to support your integration activities and the levels of code build and promotion performed within your organization.
 - Decide how many "levels" you require to promote your changes from Development to Production
 - How do you flow your changes?
 - For the different streams you plan for your SCM, you have to define how code versions changes will flow between them and what are the integration points.
 - At what levels do you build your code?
 - Code at some stream levels will be used to build, whilst others are just for configuration traceability.



16 Complete your sessions evaluation online at SHARE.org/SanFranciscoEval



- Streams (cont)
 - Various aspects of your current workflow will influence your final stream strategy, for example:
 - How do you plan to do version/release maintenance
 - How do you want to handle emergency fixes
 - Do you you require different integration levels for different teams.
 - Keep it simple when you first start
 - As teams work in RTC and get familiar with how the SCM works, you can define new streams that will support additional needs.





- Three levels to structure your Repository
 - Project Area
 - Štream
 - Component
- Project Area
 - Corresponds to a consistent group of applications
 - Criteria determinations vary depending on the company:
 - A field of business
 - Development team
- The stream
 - Maps to:
 - An application for an environment
 - A stream must be complete
 - That means that it contains all the dependencies needed by all the programs in the stream
 - Including common elements (framework)
 - Can include Components from another stream (from same or another Project Area)





- The component (1)
 - Corresponds to a part of an application
 - Divided by a topology of component types
 - Component is owned by a team
 - Single Platform
 - Simple grouping criteria
 - Stream by Platform / Team
 - Component smaller than for multi-platform
 - A lot of components if dealing with a complex system
 - Multi-Platform
 - No de-synchronization between client and server
 - Forces the same lifecycle for all technologies





- The component (2)
 - Mono-Type
 - One given type of resources per component
 - Multi-Types
 - A component contains resources of different types
 - Mono or multi-platform
 - Focus of attention for the copy
 - Copy for public interface
 - By public interface we mean ...Copy used by an application to call another application modules
 - Copy framework (cross-cutting)
 - By framework we mean ... copy such as authentication or security related, not owned by a particular application





• Stream and component structure

- Proposed architecture:
 - 1 stream per Application
 - 1 stream for common components
 - Contains components for framework resources
 - Contains components for interfaces provided by a given application that other applications can consume (public interface)
 - Allows to broadcast a change in interface when it is validated
 - Development teams are warned only when the new interfaces are stable
 - Also useful to store all system definitions
 - For each application:
 - 1 component for the public interface (API)
 - e.g. copy used for linkage section
 - 1 component per type of COBOL programs
 - Main program
 - Functional sub programs
 - DB2 Accessor

• ...





- Stream and component structure (Cont)
 - Workflow for publishing & adopting shared components





Stream and component structure (Cont)

- 1 Project Area per line of business or application
 - This depends on the team structure & relationships between applications of the same LOB
- 1 Common Project Area
 - To pool the RTC setting
 - Roles
 - Process, ..
 - To pool the shared definitions
 - System Definitions (Language Definition, Dataset Definition, etc.)
 - Build engines (?)
 - Propagation by inheritance to other PA
 - Defines the stream that publishes common components & frameworks
 - Access control
 - Read/write to Admins only
 - Read-only for all team members





Stream and component structure (Cont)

- Component ownership
 - Framework components are owned by the common project area
 - Unless there's a dedicated team for maintaining the frameworks. In this case it's a candidate for its own PA
 - Public interface components are owned by the project area of the application
 - Recommendation : 1 single stream in the Common Project Area
 - Facilitates the implementation of notifications for development teams in case of changes



Stream and component structure (Cont)

..............



Technology - Connections - Results

- Stream and component structure (Cont)
 - Ownership of components

................



Technology - Connections - Results



• How will you build your applications?

- RTC supports capabilities for the operations of building, promoting your code, packaging and deploying it, as such;
 - You will need to understand the current build process of your applications: what are different technologies in use for building your applications and how you build them.
 - "Stages" of your source code and where do you build applications, just at DEV or at various levels of the hierarchy
 - How are your applications deployed, with all the details of your target deployment locations and your runtime





- How will you build your applications (cont)?
 - RTC provides a set of features to support all these operations for your enterprise environment.
 - System definitions
 - Language Definitions Similar to JCL procedures
 - Translators Similar to JCL job steps
 - Data Set definitions Similar to JCL DD statements
 - Build definitions
 - Use of a build definition that defines a build engine, build script and build properties to control how a build is performed
 - For z/OS development a "Smart" build that will only build assets that have been modified
 - Promotion definitions
 - Package and deployment definitions





• Solution adoption

- Define the naming convention for all the elements you plan to create in Rational Team Concert
 - This should be documented and followed, so the solution is easily maintained and understood by the team members.
- Identify a pilot application (or a set of them)
 - The pilot application should be relevant in terms of technology and processes. Ideally, the team for the pilot application will just be focused on that (or minimum other work), so they don't have to duplicate efforts in different tools.
- Define the code migration process
 - You have to identify where in your system is the source code, how to extract and import it to target SCM structure, to which some reorganization of the PDSs may be needed.





Solution adoption (Cont)

- Implement the elements for supporting your solution design
 - Component and Stream design for the SCM
 - Build processes according to the information gathered
- Migrate the pilot applications/teams
- Validate and adjust the implementation
- Plan for gradual adoption for the rest of the teams





- Where are you going to host your server and repository?
 - The RTC server can run on a multitude of environments
 - Windows, Linux, AIX, Unix, zLinux, z/OS, IBMi
 - The repository database can be hosted on a multitude of environments
 - DB2 on LUW, DB2 on zLinux, DB2 on z/OS, Microsoft SQL Server, Oracle to name a few
 - The server can be run on one system with the data base on another
 - For example: Server running on zLinux and database running on DB2 on z/OS
 - You need to choose the best topology for the size and complexity of your implementation
 - What are your current server administration skills?



Tasks

- Perform the product installation
- Perform server setup and initial configuration
- Define your initial components and streams
- Define your System Definitions









Perform the product installation

- You have decided where your server will be installed so you can download the install packages from <u>www.jazz.net</u> and install the server components
 - If you are installing on z/OS these will be in SMP/E format
- Regardless of where your server is running you will need to install the SMP/E installable Build System Toolkit on z/OS
- Server installation is covered in detail in the **Installing** section in <u>https://jazz.net/help-dev/clm/index.jsp</u>





Perform server setup and initial configuration

- Perform initial setup through set up wizard
- Create users and assign licenses
- Create a project area and decide on a process template
 - When creating the project area you can deploy the default process templates, such as Scrum or OpenUP process. Chose one that fits your project needs
- Assign Project Administrator and members to the project
 - You can now assign who will be an administrator of a project and who will be just a member
 - For the project members you can also assign process roles
 - Roles define the level of authority a user has in the process





- Define your initial components and streams
 - As part of your investigation you should have defined a strategy for stream and component creation
 - Log into the repository in Eclipse and connect to your project area
 - You can rename the default stream name and default component
 - You can also create new streams and components as required





Define your system definitions

- In order for anything to build you will need to create your data set definitions, language definitions and translators
- Create a System Definitions project area?
 - The System Definitions elements are repository wide. Using a common project area for the system definitions that are generally used by all the projects eases administration and maintainability of your solution.
 - Create the System Definitions project area in the same way you created the normal project area
 - Make sure that this project area is accessible from the project areas that will utilize these common system definitions (grant read permissions).





Define your system definitions (cont)

- Can be automatically generated
 - zimport will create data set definitions for you
 - Through the <u>www.jazz.net</u> Wiki's there is a page that tells you how to import the sample mortgage application and run the system definitions set up:
 - Import Mortgage Application Sample
 - By following these instructions a set of definitions can be created that can then be easily modified
- Create manually any required data set definitions
- Create language definitions for your build processes. The details of how things are built can be defined later





- RTC provides an import utility called zimport
 - The zimport SCM command line tool (aka "mass import tool") imports your PDS members directly into the repository
 - Automatically creates the proper zComponent project structure
 - Automatically creates a data set definition based on characteristics of data set on host
 - Automatically (optionally) associates language definitions with each member
 - You can build a source code version history of your major releases by running a series of zimports with the same repository workspace





zimport preparation of data

- By Component
 - Separate out by type into a PDS
 - Cobol
 - Cobol/DB2
 - Assembler
- Line numbers if they exist must be stripped before import If they are getting re-gened in PDS – also it look cleaner
 - Cobol
 - 72-80 can leave if there are comments you want to keep
 - 1-7
 - Others?
- Zimport will scan the entire Catalog looking for the datasets you define
 - Make them a unique HLQ Userid as HLQ for example
- Problem
 - It will try to recall dataset from HSM. When it scans the catalog and does not find the dataset - It will fail over and over
 - Line numbers will cause RTC merge to fail



• What to import?

- All source, recommendation is that the production baseline versions are imported
 - Cobol, PL/I, JCL, Procs, etc
 - Adding in all versions will be very costly and time consuming
 - If the IBM services team is engaged they have additional tools to help
 - For example A procedure has been developed to off load older versions that can be viewed through ISPF when necessary
- SCLM Language definitions, Endevor processors or Changeman skeletons need to be converted to RTC definitions
 - Language Definitions
 - Translators





Code pages and line delimiters

- Files are typically stored in the Jazz repository in UTF-8; files on z/OS are typically stored in EBCDIC.
- When we zimport, we convert from EBCDIC to UTF-8. When we load the files back out (zload, build, etc), we convert from UTF-8 to EBCDIC.
- To avoid the conversion on zimport, specify -b for binary. In that case, we set the line delimiter to none and do no conversion. Otherwise, we convert and set the line delimiter to platform. We determine which encoding to use for the conversion based on the ZLANG environment variable. If ZLANG is set, use it. If ZLANG is unset, use the system's default encoding.
- Files with line delimiter of none are not converted when loaded back out. Otherwise:
 - To determine which encoding to use when loading to MVS, we:
 - Check the mvsCodePage versionable property of the file. If present and non-blank, it is used.
 - If no mvsCodePage, default to the value of the ZLANG environment variable, if ZLANG is set.
 - If the mvsCodePage property is not present, and ZLANG is unset, default to using the system's default encoding.
 - To determine which encoding to use when loading to USS, we:
 - Check the mvsCodePage versionable property of the file. If present, and non-blank, it is used.
 - If the mvsCodePage property is present but blank:
 - If ZLANG is set, use the value of ZLANG.
 - If ZLANG is unset, default to the system's default encoding.
 - If the mvsCodePage property is not present, load the file as it is in the repository no conversion is done.





Handling non-roundtrippable characters

- "Non-roundtrippable" refers to characters that cannot be converted from EBCDIC to UTF-8 (to store in the SCM) and back to EBCDIC (to load back to MVS for build or edit).
- For example, your source files may include strings containing control characters such as a Carriage Return (0x'0D' in EBCDIC) for printing or any other embedded hex code. Your zimport will fail in this case if you do not use binary mode, because the perceived line delimiter in the middle of the line is inconsistent with the rest of the file (recall the MVS files are record length-based and do not contain line breaks at the end of each line). Other special characters may not break zimport but will still get mangled when roundtripped.
- Limitations on files containing non-roundtrippable characters:
 - Must be zimported using the binary option
 - Must be viewed and edited using the ISPF client. Eclipse client cannot handle these files.
 - RDz can be used to edit files directly on MVS after they are loaded using the ISPF client or zload.
 - Compare and merge capability is supported only in the ISPF client or using RDz.





- Once zimport has been complete you can set up the rest of your system definitions
 - Create any additional Data set definition
 - zimport will have created data set definitions for "inputs"
 - COBOL, PL/I, ASM, JCL
 - Use RTC dialog to create data set definitions for "outputs"
 - OBJ, DBRM, LOAD
 - Also create data set definitions for temporary files
 - Convert your build processors (SCLM lang defs, Endevor processors or changeman skeletons) to RTC definitions



Translator comparison to JCL using data set definitions

						<u>ش</u> د	Data Set Defin	itions
		🕞 Translate	or			6	🖁 JKE Assemb	ler
						1	🖁 JKE BIND fil	es
		Name: JKE CO	OBOL compilation (CICS&DE	2)		1	🖁 JKE BMS ma	ps
JCL Line	Corresponding data		•			1	JKE CEE.SCE	ELKED
	set definition name	Consent				1	JKE CICS.SD	FHCOB
		General				1	L JKE CICS.SD	FHLOAD
		Description				1	L JKE CICS.SD	FHMAC
COBOL EXEC PGM=IGYCRCTL,REGION=2048K,	COBOL Compiler					0	L JKE COBOL.	SIGYCOMP
						í.	JKE COBOL	compiler
XX PARM=('EXIT(ADEXIT(ELAXMGUX))',	No DSD					1	L JKE COBOL	source codes
XX 'ADATA',		- Call Meth	od			1	🖁 JKE Copybo	oks
XX 'LIB', YY 'TEST/NONE SYM SED\'		Called pr	ocxam			ĺ.	L JKE DB2.SD	SNLOAD
XX 'LIST',	******			Mari				
XX 'FLAG(I,I)'&CICS&DB2&COMP)		Data set o	definition. JKE COBOL comp	biler				Browse.
			Address of the second sec					
		Default o	ptions. EXII (ADEXII (ELA	(XMGUX)),A	DATA,LIB	,TEST(NONE	:,SYM,SEP),LI	ST,FLAG(I,I)
XXSTEPLIB DD DISP=SHR,		DD name	s list [,]					
 JCL - DISP=SHR,DSN=COBOL.V4R2.SIGYCOMP	COBOL.SIGYCOMP	D D Hume	5 Hou					
JCL- DISP=SHR,DSN=RDZ.V8R0M3.SFEKLOAD	WDZ.SFEKLOAD	Maximum ret	urn code: 4					
JCL- DISP=SHR,DSN=CICSTS.V4R1.CICS.SDFHLOAD	CICS.SDFHLOAD	-						
JCL- DISP=SHR,DSN=DB2.DB40.SDSNLOAD	DB2.SDSNLOAD	 Data Set Pr 	operties					
		DD concaten	ations					
COBOL.SYSLIB DD DISP=SHR,	Copybooks	DD Name		Data Cat	Definition			Add
DSN=F057699.TEST.RTC.COPY		DD Name		Data Set	Definitio	ns		Add
		SYSLIB		JKE Copy	ybooks,JK	E CICS.SDFH	ICOB	Edit
COBOL.SYSIN DD DISP=SHR,	<input/> represents the source file associated with	TASKLIB		JKE COB	OL.SIGYC	OMP,JKE CIC	S.SDFHLO	
// DSN=F057699.TEST.RTC.COBOE(EPSCMORT)	the language definition							Remove
	being built	DD allocatio	201					
		DD allocatio	ns.					
	T C C C C C C C C C C	DD Name	Data Set Definition	Member	Кеер	Output	Publish	Add
//COBOL.STSLIN DD DSN=&&OBJ,SPACE=(TRK,(3,3)), // UNIT=SYSDA, DISP=(NEW,PASS)	i emporary file (object deck)	SYSIN	<input/>	no	no	no	no	Edit
// DCB=(RECFM=FB,LRECL=256,BLKSIZE=2560)		SYSLIN	JKE Temporary file (obje	no	yes	no	no	Eurt
		DBRMLIB	JKE DBRM library	yes	no	yes	no	Remove
SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))	Temporary file	SYSPRINT	JKE Temporary file	no	no	no	ves	Incinove
SYSUT2 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))		SYSADATA	IKE Temporary file	00	00	no	00	
SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT4 DD UNIT=SYSALLDA.SPACE=(CYL,(1,1))		SUSVALCO	IKE Temporary file	10	10	10	10	
SYSUT5 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))		STSXIVILSD	The Temporary file	no	no	no	no	
SYSUT6 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))		SYSUT1	JKE Temporary file	no	no	no	no	
SYSUT7 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))		SYSUT2	JKE Temporary file	no	no	no	no	
-	-		The second se					
4 Complete your sessions evaluation online at S	HAFE.org/SanFrancisco	SYSUT3	JKE Temporary file	no	no	no	no	



🔓 Team Artifact 🛛 🔪 🔅 Team Dashbo 🔚 My Work 🔞 Team Organiz) 増 Packa My Filter (261 of 261 areas selected) E 🎝 SHARE RTC Lab Project [mvs1.centers.ihost.com] b 🙀 Builds Enterprise Extensions Build Subsets D Context-Aware Search Deployments B Reckages D Promotions b 😡 Source Code Data A R System Definitions 🛍 IBM i Þ ⊿ 👘 z/OS Data Set Definitions Language Definitions Translators BMS map processing (copybook generation) BMS map processing (object deck generation) COBOL compilation COBOL compilation (CICS) COBOL compilation (CICS&DB2) COBOL compilation (for subroutines) 🔄 Link-edit 🔄 Link-edit using linkage-editor deck BMS map processing 🕅 COBOL compilation COBOL compilation (CICS&DB2) and link-edit COBOL compilation (CICS) and link-edit COBOL compilation and link-edit Copybook (no translators)

- Capture and maintain supporting information required to build your mainframe applications :
 - Data set definitions represent all MVS data sets involved in building your application from the source data sets (to be created and loaded with source from the repository) to the output data sets (generated by the build) and even the compiler itself
 - **Translators** represent a step in the build process, such as a compile or link-edit
 - Language definitions represent the complete ordered set of steps (translators) required to build a given source





• Once you have all your translators and language definitions defined, you can assign them to the relevant files





Migrating your build to Rational Team Concert

- Build Definition
 - 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
- Build Engine
 - RTC representation of a process running on a build machine that executes build requests
- Build Agent
 - Executes the build
 - Located on z/OS (for mainframe)
 - Accesses RTC to retrieve source code and other information

Build request and build result

 Representations of the request to run a build and the output from the build run

D: SHARE RTC	Lab Project build	Project o	or Team Area:	SHARE RTC Lab Proj	ect	Bro
Build Workspac	e				- huild it- flour	
Workspace:*	SHARE RTC Build	Workspace	it should have	the stream you want t	Select	Create
Load Options						
Load Options Specify file extra	action details. Prop	erties can be refere	Generate a	build file from these l	anguage definition	15
Load Options Specify file extra Load directory:	action details. Prop * /shareuser/lxd	erties can be refere 1/RTCLabBuild	Generate a BMS ma	build file from these l p processing	anguage definition	15
Load Options Specify file extra Load directory: Delete direct	action details. Prop * /shareuser/lxd tory before loading	erties can be refere	Generate a BMS ma COBOL of	build file from these l p processing compilation	anguage definition	15
Load Options Specify file extra Load directory: Ø Delete direct Resource prefix	action details. Prop * /shareuser/kxd tory before loading ** LXD1.RTCBUIL	erties can be refere (1/RTCLabBuild D	Generate a BMS ma COBOL c COBOL c COBOL c	build file from these l p processing compilation compilation and link-e compilation (CICS) an	anguage definition dit d link-edit	15
Load Options Specify file extra Load directory: Delete direct Resource prefix Load worksp	action details. Prop * /shareuser/kxd tory before loading ** LXD1.RTCBUIL pace to the load dir	erties can be refere 1/RTCLabBuild D ectory at the begin	Generate a BMS ma COBOL c COBOL c COBOL c COBOL c COBOL c	build file from these l p processing compilation compilation and link-e compilation (CICS) an compilation (CICS&DE	anguage definition dit d link-edit l2) and link-edit	15







RTC z/OS builds : How it all hangs together



Additional Considerations



• Security

- https://jazz.net/wiki/bin/view/Main/ZosBuildAgentSec
- <u>http://publib.boulder.ibm.com/infocenter/clmhelp/v3r0m1/index.jsp?t</u> opic=%2Fcom.ibm.team.build.doc%2Ftopics%2Fr antz security.ht <u>ml</u>
- <u>https://jazz.net/wiki/bin/view/Main/DependencyBuildScenarioOpenS</u>
 <u>SLSetup</u>
- ISPF Client set up
 - <u>https://jazz.net/help-</u> <u>dev/clm/topic/com.ibm.jazz.install.doc/topics/c_client_ispf_installati</u> <u>on.html</u>
- Promotion
- Deployment
- RDz Integration

49 Complete your sessions evaluation online at SHARE.org/SanFranciscoEval



Additional Resources

Jazz.net

- <u>https://jazz.net/library/</u>
 - 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
- zimport additional resources
 - <u>System z mass import tool overview (Information Center)</u>
 - <u>Getting my MVS files into the RTC repository (and getting them back out again)</u>
- Developerworks resources on migration
 - <u>http://www.ibm.com/developerworks/rational/library/migrate-</u> rational-team-concert-zos-application-development/index.html

