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

Liam Doherty
IBM Corporation

Thursday February 7th, 2013
Session 12531
Agenda

• What is Rational Team Concert?
• 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
Rational Team Concert (RTC): An open, extensible architecture

Supporting a broad range of desktop clients, IDEs and languages
Rational Team Concert Features

Work item tracking
Work items are the fundamental mechanism in Rational Team Concert to track and coordinate development tasks and workflows, all governed by your team's associated process. In addition, work items are the hub for linkage between work items, providing support for item types or more.

Continuous builds
The Team Build component integrates the team's build system into Rational Team Concert, providing build awareness, control, and traceability to the team. Team members can track build progress, view build alerts and results, request builds, and trace builds to other artifacts like change sets and work items.

Source control
The Source Control component, the Jazz platform and agile development environment, is the hub for source control and contains aspects of change sets and work items.

Planning
The Planning component supports both agile and traditional planning. It helps balance the work between teams and plans access dependencies and links to project plans.

Transparency/project health
The Team Reports and Web Dashboards components of Rational Team Concert help you to keep tabs on the health of your project. Dashboard provides an at-a-glance view of work item queries, event feeds, reports, and other items that are critical to understanding your progress. Reports provide both real-time views and historical trends of builds, streams, work items, and other artifacts that your team works with.

Administration
The Jazz Team Server provides a web-based administrative UI for setup, configuration, and administration of the Rational Team Concert environment.

Built-in Integrations
Rational Team Concert also integrates with other products, such as Rational Quality Manager, Rational ClearQuest and Rational ClearCase. The integration with Rational Quality Manager and Rational ClearQuest is part of the Collaborative Application Lifecycle Management (CLM) effort which also includes Rational Requirements Composer. Collaboration tool integrations are also built-in for IBM Sametime, IBM Connections and other popular tools.
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

Complete your sessions evaluation online at SHARE.org/SanFranciscoEval
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
  • 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
Planning your Rational Team Concert solution

- 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
Planning your Rational Team Concert solution (cont)

• **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
    • ...
Planning your Rational Team Concert solution (cont)

- **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
Planning your Rational Team Concert solution (cont)

<table>
<thead>
<tr>
<th><strong>Stream</strong></th>
<th>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)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Component</strong></td>
<td>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. <strong>Recommended: 500 – 1000 files / component</strong></td>
</tr>
<tr>
<td><strong>Repository Workspace</strong></td>
<td>Workspace for 1 user synchronized with a Stream and the “Sandbox” Situated on the RTC server</td>
</tr>
<tr>
<td><strong>Sandbox</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>Change Set</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Work Item</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>Baseline</strong></td>
<td>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</td>
</tr>
<tr>
<td><strong>Snapshot</strong></td>
<td>Collection taken of all component baselines for a stream or repository workspace capturing an interesting point in time</td>
</tr>
</tbody>
</table>
Planning your Rational Team Concert solution (cont)

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>Action that copies selected files and folders from the repository workspace to the sandbox (eclipse workspace or MVS data sets)</td>
</tr>
<tr>
<td>Accept</td>
<td>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</td>
</tr>
<tr>
<td>Check-in</td>
<td>Action that allows to save local changes into the repository workspace, within a Change Set</td>
</tr>
<tr>
<td>Deliver</td>
<td>Action to push the workspace changes from the workspace to the Stream</td>
</tr>
</tbody>
</table>
Planning your Rational Team Concert solution (Cont)

• 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.
Planning your Rational Team Concert solution (Cont)

- **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
Planning your Rational Team Concert solution (Cont)

- 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.*
Planning your Rational Team Concert solution (Cont)

- 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.
Planning your Rational Team Concert solution (Cont)

• 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 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.
Planning your Rational Team Concert solution (Cont)

• **RTC Repository Architecture**
  • **Three levels to structure your Repository**
    • Project Area
      • Stream
        • 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)
Planning your Rational Team Concert solution (Cont)

- RTC Repository Architecture (Cont)
  - 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
Planning your Rational Team Concert solution (Cont)

• RTC Repository Architecture (Cont)
  • 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
Planning your Rational Team Concert solution (Cont)

- **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
      - …
Planning your Rational Team Concert solution (Cont)

- Stream and component structure (Cont)
  - Workflow for publishing & adopting shared components
Planning your Rational Team Concert solution (Cont)

- **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*
Planning your Rational Team Concert solution (Cont)

- **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*
Planning your Rational Team Concert solution (Cont)

- Stream and component structure (Cont)
- Project Area set up
Planning your Rational Team Concert solution (Cont)

- Stream and component structure (Cont)
  - Ownership of components
Planning your Rational Team Concert solution (Cont)

- **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
Planning your Rational Team Concert solution (Cont)

- **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
Planning your Rational Team Concert solution (Cont)

- **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.
Planning your Rational Team Concert solution (Cont)

- 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
Planning your Rational Team Concert solution (Cont)

- **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?
Installing and Configuring the Rational Team Concert server and components

• **Tasks**
  • Perform the product installation
  • Perform server setup and initial configuration
  • Define your initial components and streams
  • Define your System Definitions
Installing and Configuring the Rational Team Concert server and components

• Perform the product installation
  • You have decided where your server will be installed so you can download the install packages from [www.jazz.net](http://www.jazz.net) 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 [https://jazz.net/help-dev/clm/index.jsp](https://jazz.net/help-dev/clm/index.jsp)
Installing and Configuring the Rational Team Concert server and components

- 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. Choose 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*
Installing and Configuring the Rational Team Concert server and components

• 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
Installing and Configuring the Rational Team Concert server and components

- 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).
Installing and Configuring the Rational Team Concert server and components

• Define your system definitions (cont)
  • Can be automatically generated
    • zimport will create data set definitions for you
    • Through the www.jazz.net 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
Migrating your source code to Rational Team Concert

- 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
Migrating your source code to Rational Team Concert

• **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-generated 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
Migrating your source code to Rational Team Concert

• 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
Migrating your source code to Rational Team Concert

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.
Migrating your source code to Rational Team Concert

Handling non-roundtrip-pammable characters

- “Non-roundtrip-pammable” 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-roundtrip-pammable 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.
Migrating your source code to Rational Team Concert

- 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

<table>
<thead>
<tr>
<th>JCL Line</th>
<th>Corresponding data set definition name</th>
</tr>
</thead>
<tbody>
<tr>
<td>COBOL   EXEC PGM=IGYCRCTL,REGION=2048K,</td>
<td>COBOL Compiler</td>
</tr>
<tr>
<td>XX PARM=&quot;EXIT(ADEXIT(ELAXMGUX))&quot;, XX 'ADATA', XX 'LIB', XX 'TEST(NONE,SYM,SEP)', XX 'LIST', XX 'FLAG(U,L)&amp;CICS&amp;DB2&amp;COMP)</td>
<td>No DSD</td>
</tr>
<tr>
<td>XXSTEPLIB DD DISP=SHR,</td>
<td>Copybooks</td>
</tr>
<tr>
<td>// COBOL.SYSLIB DD DISP=SHR, DSN=F057699.TEST.RTC.COPY</td>
<td>JKE Copybooks,JKE COBOL.SIGYCOMP,WZ.SFEKLOAD,CICS.SDFHLOAD,</td>
</tr>
<tr>
<td>COBOL.SYSIN DD DISP=SHR,</td>
<td>Temporary file (object deck)</td>
</tr>
<tr>
<td>// DSN=F057699.TEST.RTC.COBOL(EPSCMORT)</td>
<td>SYSIN</td>
</tr>
<tr>
<td>//COBOL.SYSLIN DD DSN=&amp;&amp;OB1,SPACE=(TRK,(3,3)), // UNIT=SYSDA, DISP=(NEW,PASS) // DCB=(RECFM=F8,RECL=256,BKLSIZE=2560)</td>
<td>Temporary file</td>
</tr>
<tr>
<td>SYSSYS1 DD UNIT=SYSALLOA,SPACE=(CYL,11,1)</td>
<td>JKE Temporary file</td>
</tr>
<tr>
<td>SYSSYS2 DD UNIT=SYSALLOA,SPACE=(CYL,11,1)</td>
<td>JKE Temporary file</td>
</tr>
<tr>
<td>SYSSYS3 DD UNIT=SYSALLOA,SPACE=(CYL,11,1)</td>
<td>JKE Temporary file</td>
</tr>
<tr>
<td>SYSSYS4 DD UNIT=SYSALLOA,SPACE=(CYL,11,1)</td>
<td>JKE Temporary file</td>
</tr>
<tr>
<td>SYSSYS5 DD UNIT=SYSALLOA,SPACE=(CYL,11,1)</td>
<td>JKE Temporary file</td>
</tr>
<tr>
<td>SYSSYS6 DD UNIT=SYSALLOA,SPACE=(CYL,11,1)</td>
<td>JKE Temporary file</td>
</tr>
<tr>
<td>SYSSYS7 DD UNIT=SYSALLOA,SPACE=(CYL,11,1)</td>
<td>JKE Temporary file</td>
</tr>
</tbody>
</table>

Complete your sessions evaluation online at SHARE.org/SanFrancisco
Migrating your source code to Rational Team Concert

- 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.
Migrating your source code to Rational Team Concert

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

Tasks run on the host, such as compilation. A build can handle several different tasks in the order shown.

Language definition
- Language Definitions
  - Translators
    - JKE BMS map processing
    - JKE COBOL compilation
    - JKE COBOL compilation (CICS+DB2) and link-edit

File Extension
- Corresponds to a STEP in the process to run on the host
- contributes to the step to execute.
- Corresponds to programs/files/PDS lines used by EXEC, DD, SYSIN,… as in JCL

ATTENTION
The name of the directory on USS must be unique to each build definition.
Additional Considerations

- Security
  - [https://jazz.net/wiki/bin/view/Main/ZosBuildAgentSec](https://jazz.net/wiki/bin/view/Main/ZosBuildAgentSec)
  - [https://jazz.net/wiki/bin/view/Main/DependencyBuildScenarioOpenSSLSetup](https://jazz.net/wiki/bin/view/Main/DependencyBuildScenarioOpenSSLSetup)
- ISPF Client set up
- Promotion
- Deployment
- RDz Integration
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
- zimport additional resources
  - System z mass import tool overview (Information Center)
  - Getting my MVS files into the RTC repository (and getting them back out again)
- Developerworks resources on migration