IBM Technical Summit 2013

Démarquez-vous

17 octobre | IBM Client Center Paris

IBM Technical Summit 2013

Démarquez-vous

17 octobre | IBM Client Center Paris

Les étapes de la migration mainframe vers RTC

Nicolas Dangeville

Dangeville.n@fr.ibm.com

RTC-EE Chief Architect

What is Rational team Concert?

- 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

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 Used to control access rights, facilitate sharing and reuse Theoretical limit: 50000 files **Recommended: 1000 – 2000 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

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 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

Migration planning

- The migration from a legacy mainframe SCM system to RTC involves several steps before you have an operational system.
  - Pre-migration
  - During the migration
  - Post-migration

Pre-migration

- Install RTC!
- Create project areas
- Define the structure of your streams
- Define the structure of components in your stream
- Define the delivery flow between streams
- Identify the nature of the programs and the main language definitions you will need for your IT system
- Create system definitions for data set definitions, language definitions and translators
- Set translator variables

Planning your Rational Team Concert solution

- 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.*

Planning your Rational Team Concert solution

- **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
    - eg: 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?

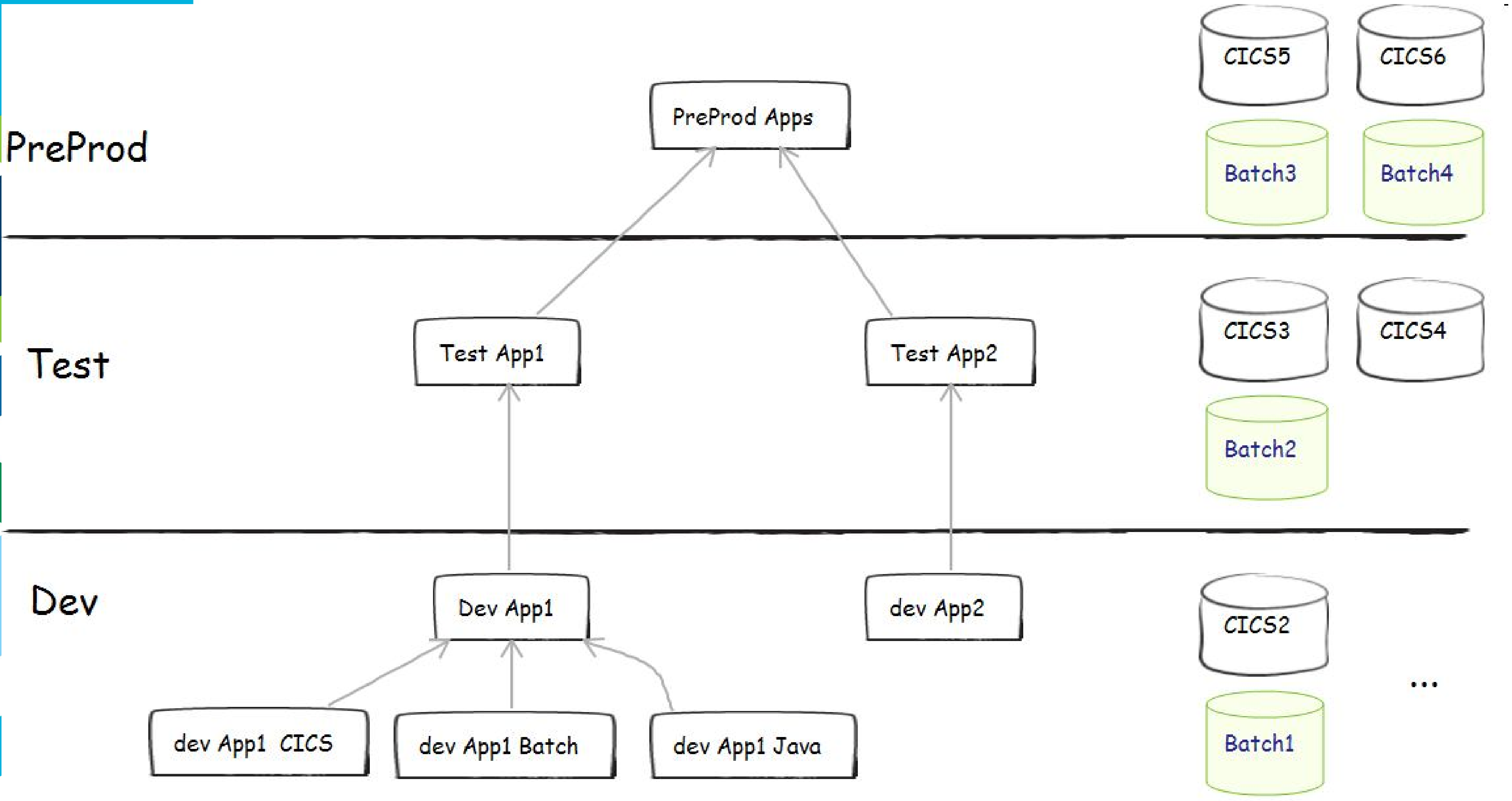
Define your projects areas

- **Project Area structure**
  - 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 and Process, ..
    - To pool the shared definitions
      - System Definitions (Language Defs, Dataset Defs, 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

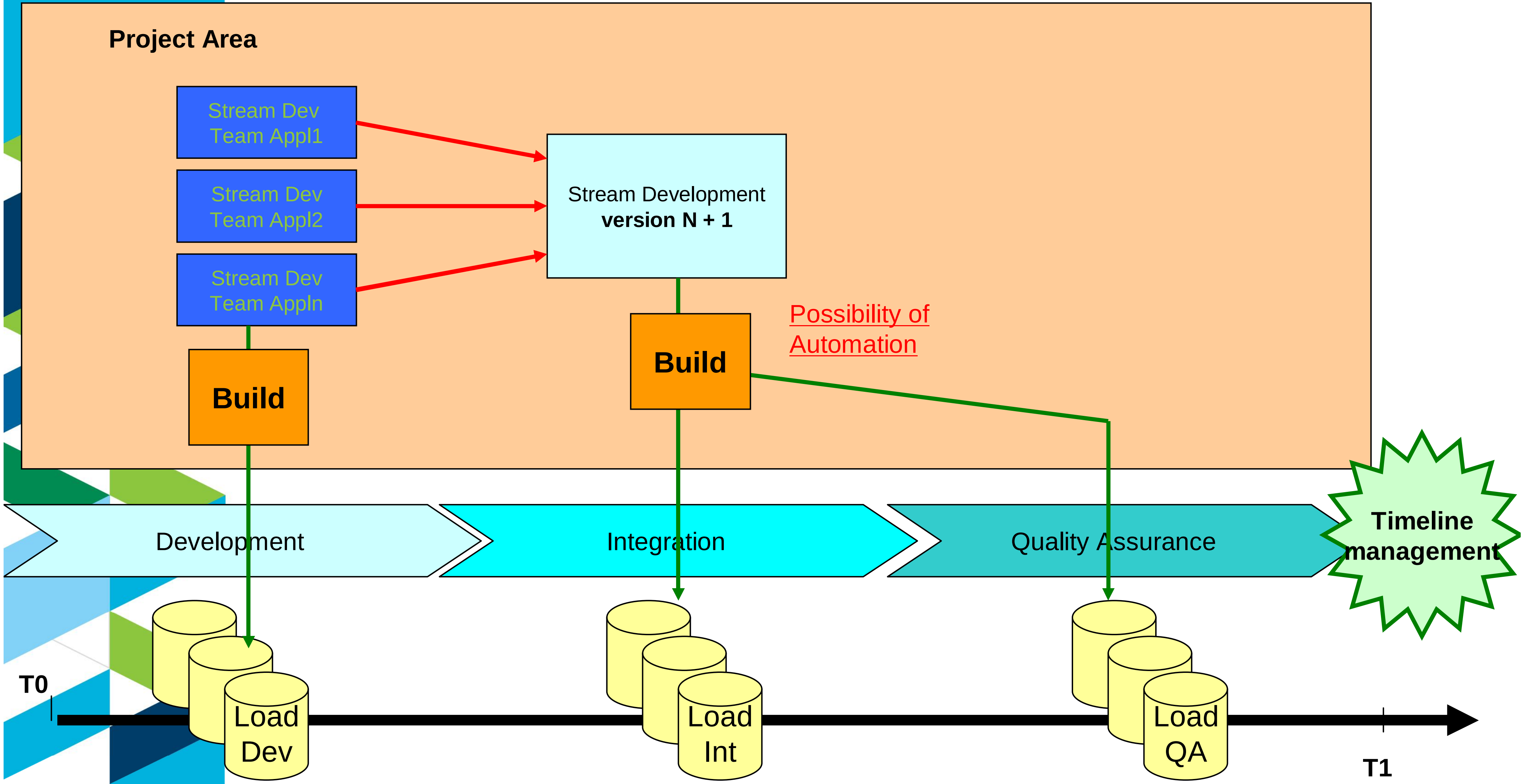
Define the structure of the streams

- 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)

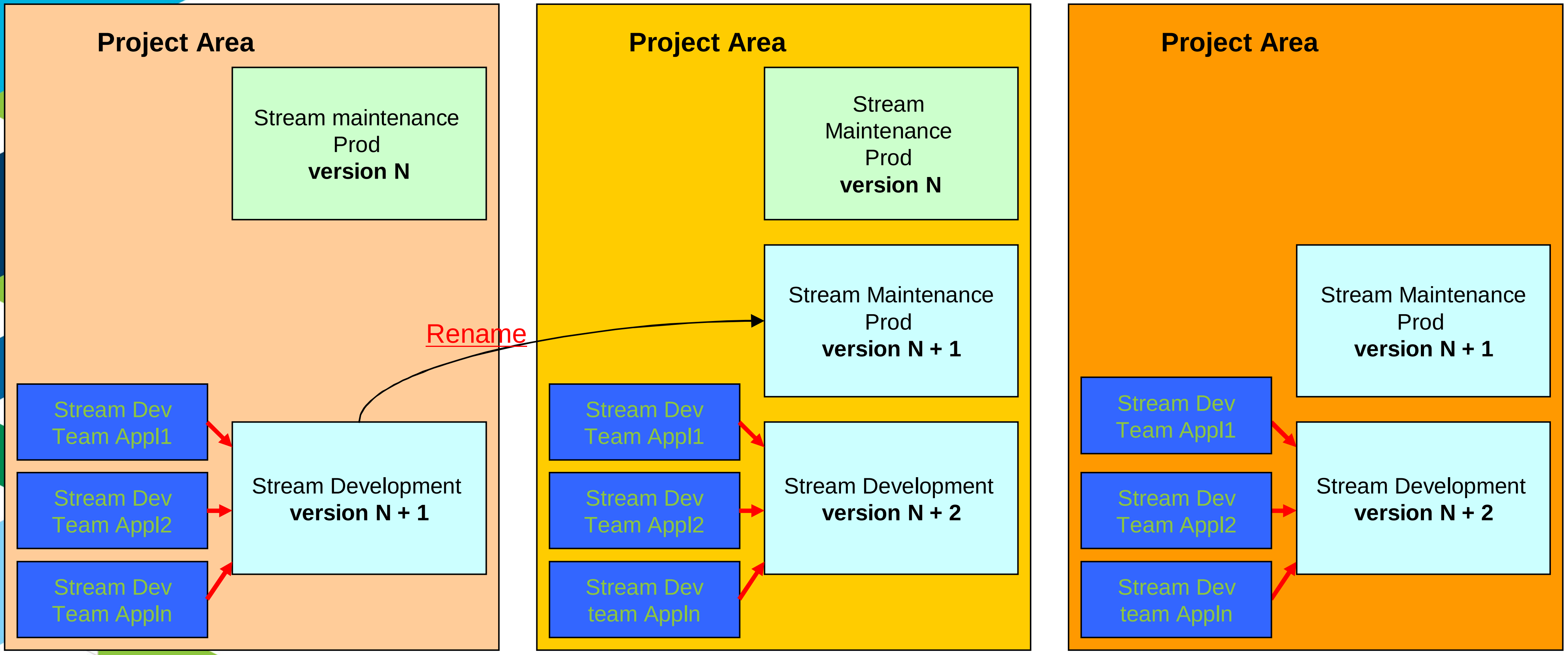
Hierarchy and Integration



Evolution of the streams over time



Evolution of the streams over time


Project Area
Stream maintenance Prod version N
Rename
Stream Dev Team Appl1
Stream Dev Team Appl2
Stream Dev Team Appln
Stream Development version N + 1
Project Area
Stream Maintenance Prod version N
Stream Maintenance Prod version N + 1
Stream Dev Team Appl1
Stream Dev Team Appl2
Stream Dev team Appln
Stream Development version N + 2
Project Area
Stream Maintenance Prod version N + 1
Stream Dev Team Appl1
Stream Dev Team Appl2
Stream Dev team Appln
Stream Development version N + 2

Components

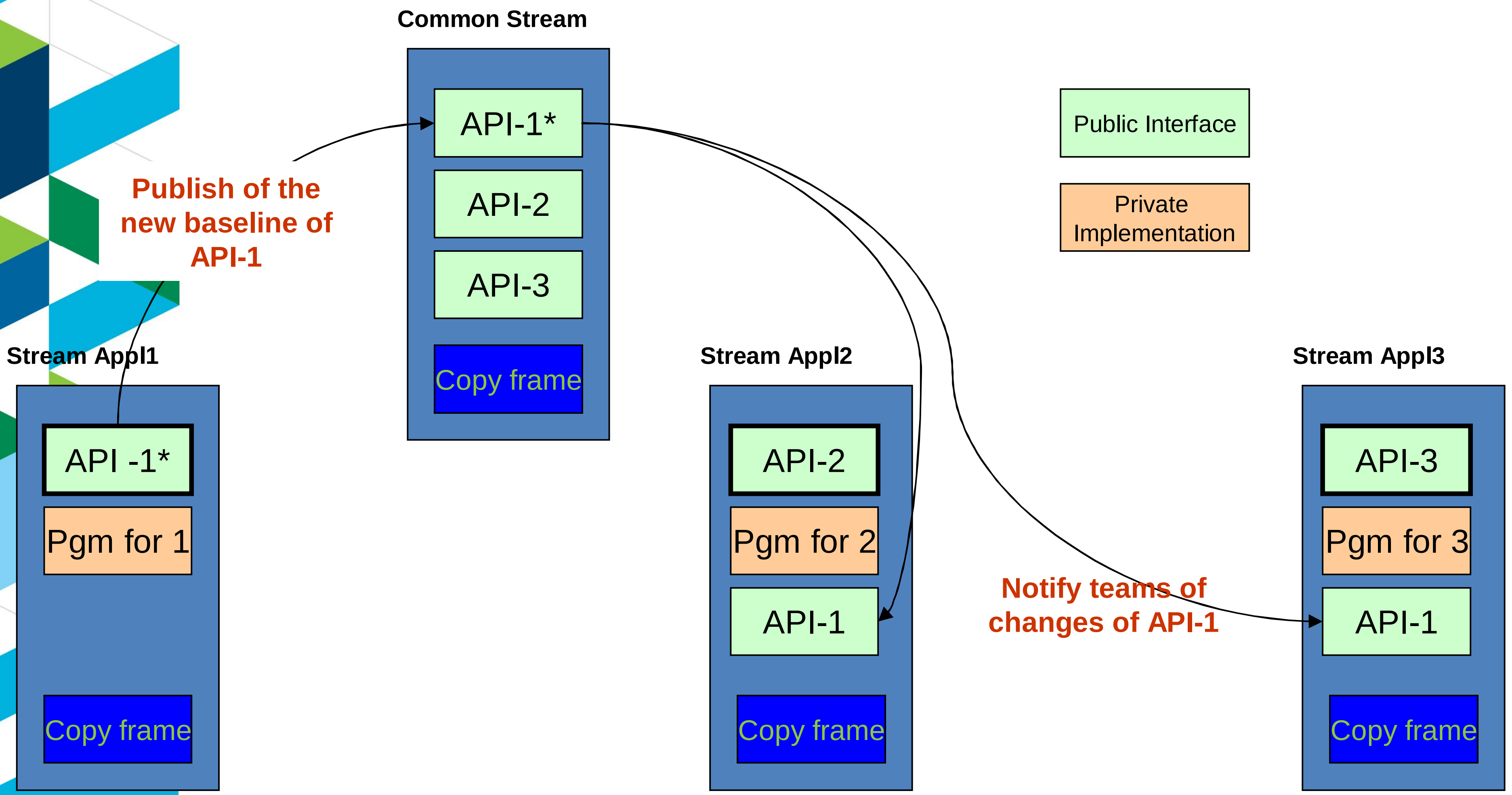
- Which logical units make up the applications (components)?
  - Put related artifacts or projects together so components make sense from code reuse, application build operations and team sharing perspective
- What are the common source elements used across several applications/modules?
  - Define components 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 for which they are responsible.
  - When structuring the components along with architectural details bear also in mind the organizational structure that will support it.

The component

- 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
  - 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
- Focus of attention for the copybooks
  - 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

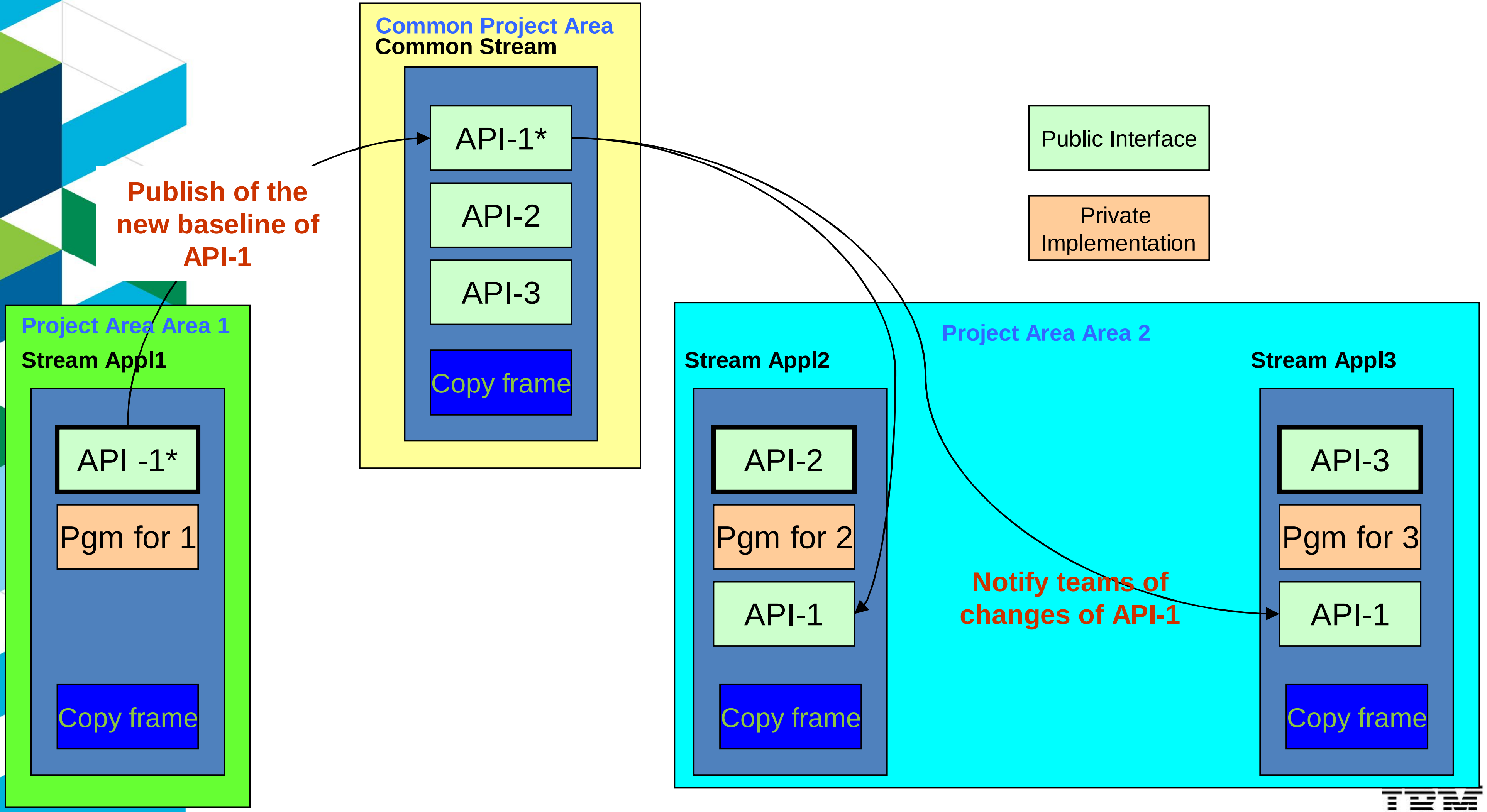
Project Area, Stream and Component Structure

- Workflow for publishing & adopting shared components



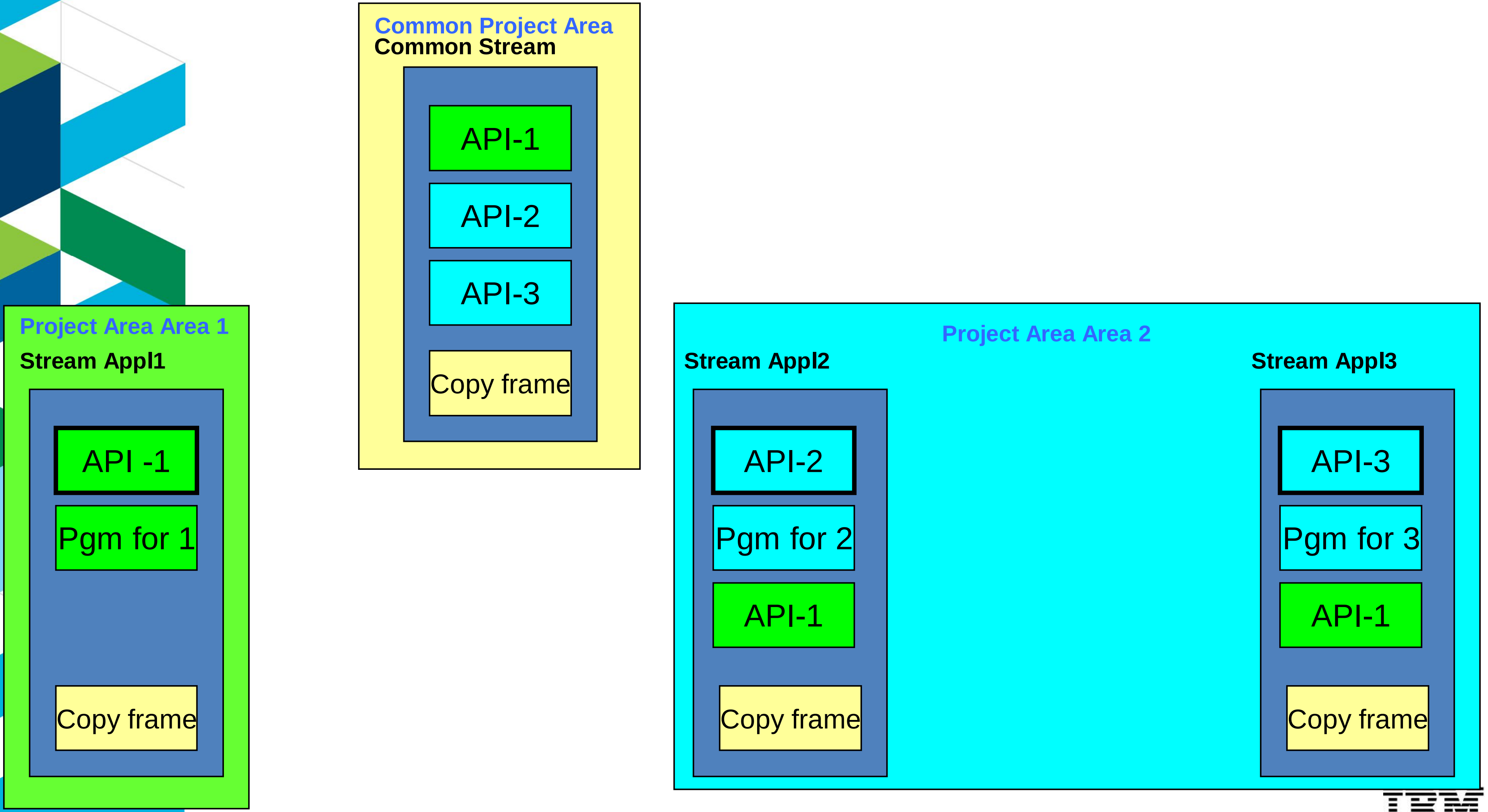
Project Area, Stream and Component Structure

- Project Area set up



Project Area, Stream and Component Structure

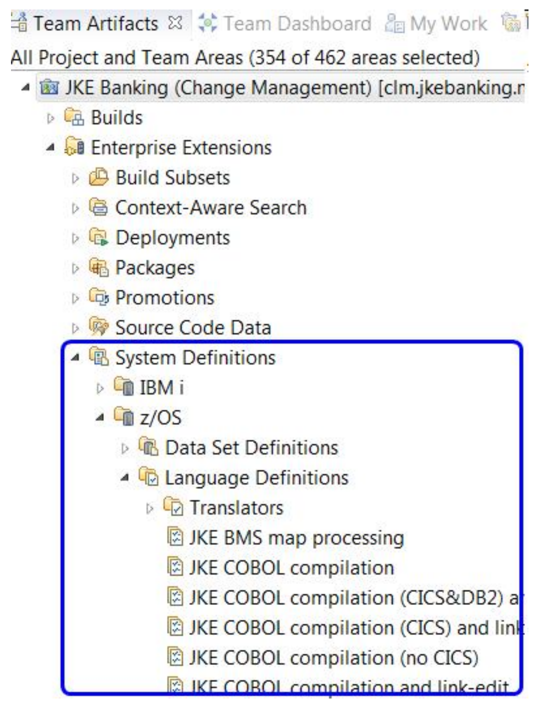
- Ownership of components



Create System Definitions

- **Data set definitions** *describe data sets* involved in the build process
  - E.g., a COBOL compiler data set definition contains the name of the actual compiler PDS and member
- **Translators** *define a single step* in the build process.

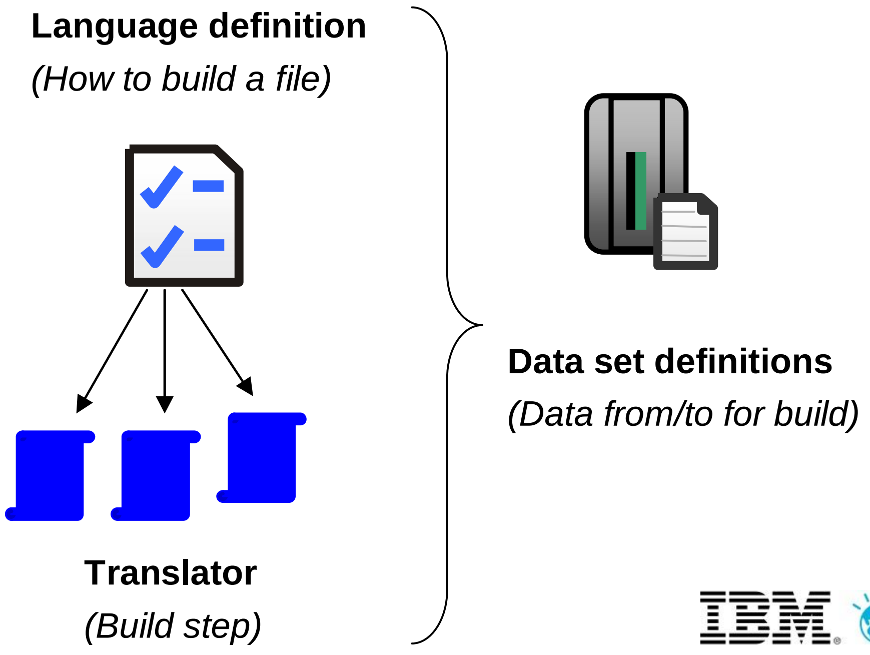
  It's in the translator that you specify that the build will perform a compilation, link-edit, etc.
  - E.g., a COBOL compilation translator contains a reference to the compiler data set definition, default compiler options, required DD concatenation and allocations, and a maximum successful return code
- **Language definitions** *order the steps* in the build process
  - E.g., a language definition for a main program contains references first to the COBOL compilation translator and second to the link-edit translator



Team Artifacts ☒ Team Dashboard My Work

All Project and Team Areas (354 of 462 areas selected)

- JKE Banking (Change Management) [clm.jkebanking.r
  - Builds
  - Enterprise Extensions
    - Build Subsets
    - Context-Aware Search
    - Deployments
    - Packages
    - Promotions
    - Source Code Data
    - System Definitions
      - IBM i
      - z/OS
        - Data Set Definitions
        - Language Definitions
          - Translators
          - JKE BMS map processing
          - JKE COBOL compilation
          - JKE COBOL compilation (CICS&DB2) a
          - JKE COBOL compilation (CICS) and lin
          - JKE COBOL compilation (no CICS)
          - JKE COBOL compilation and link-edit

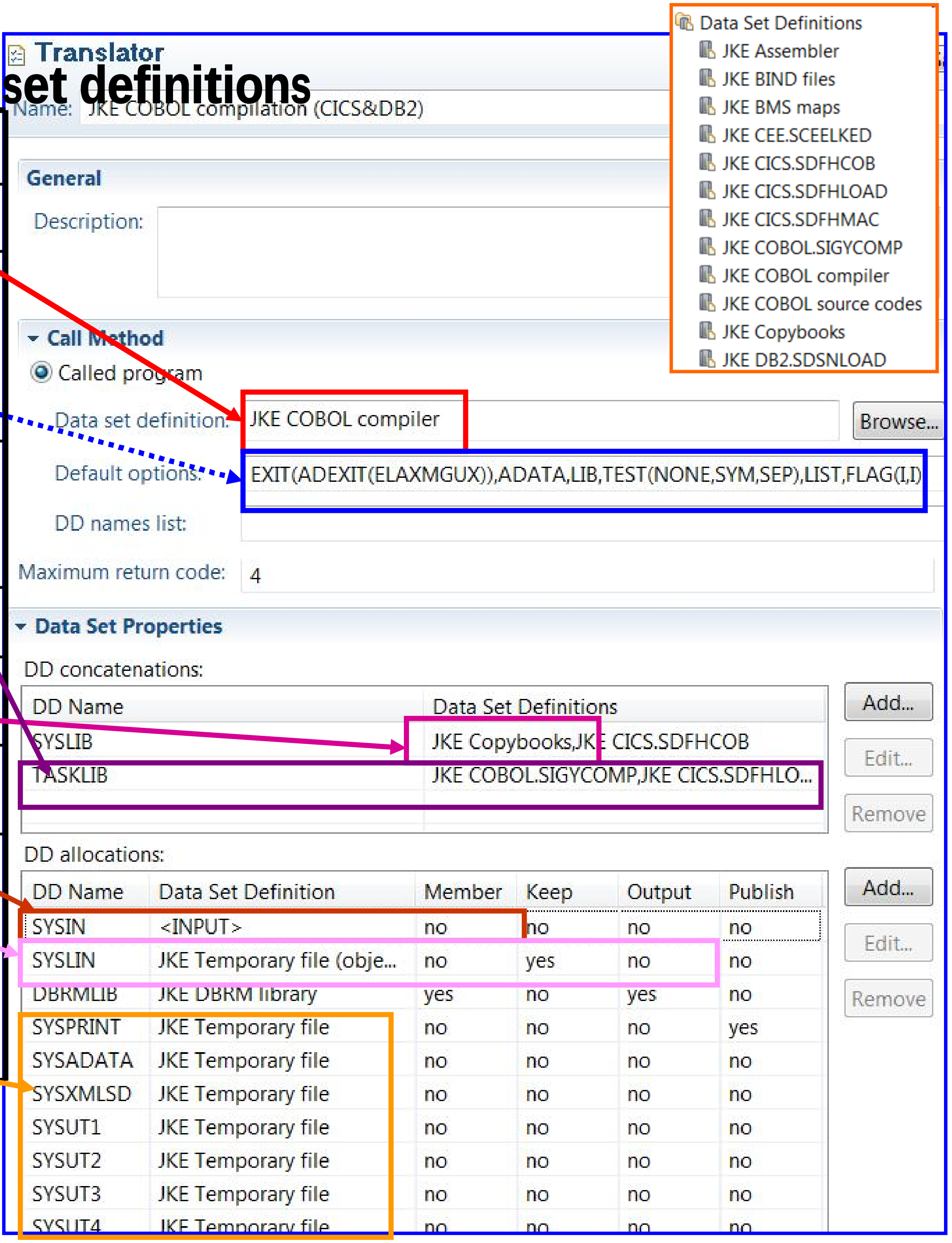


Translator comparison to JCL using data set definitions

JCL Line	Corresponding data set definition name
COBOL EXEC PGM=IGYCRCTL,REGION=2048K,	COBOL Compiler
XX PARM=('EXIT(ADEXIT(ELAXMGUX))', XX 'ADATA', XX 'LIB', XX 'TEST(NONE,SYM,SEP)', XX 'LIST', XX 'FLAG(I,I)'&CICS&DB2&COMP)	No DSD
XXSTEPLIB DD DISP=SHR, ...JCL - DISP=SHR,DSN=COBOL.V4R2.SIGYCOMP ...JCL- DISP=SHR,DSN=RDZ.V8R0M3.SFEKLOAD ...JCL- DISP=SHR,DSN=CICSTS.V4R1.CICS.SDFHLOAD ...JCL- DISP=SHR,DSN=DB2.DB40.SDSNLOAD	COBOL.SIGYCOMP WDZ.SFEKLOAD CICS.SDFHLOAD DB2.SDSNLOAD
COBOL.SYSLIB DD DISP=SHR, DSN=F057699.TEST.RTC.COPY	Copybooks
COBOL.SYSIN DD DISP=SHR, // DSN=F057699.TEST.RTC.COBOL(EPSCMORT)	<INPUT> represents the source file associated with the language definition being built

//COBOL.SYSLIN DD DSN=&&OBJ,SPACE=(TRK,(3,3)),
// UNIT=SYSDA, DISP=(NEW,PASS)
// DCB=(RECFM=FB,LRECL=256,BLKSIZE=2560)
	Temporary file (object deck)

SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT2 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT4 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT5 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT6 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1)) SYSUT7 DD UNIT=SYSALLDA,SPACE=(CYL,(1,1))	Temporary file



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

Translator

Name: JKE COBOL compilation (CICS&DB2)

General

Description:

Call Method

(●) Called program

Data set definition: JKE COBOL compiler

Default options: EXIT(ADEXIT(ELAXMGUX)),ADATA,LIB,TEST(NONE,SYM,SEP),LIST,FLAG(I,I)

DD names list:

Maximum return code: 4

Data Set Properties

DD concatenations:

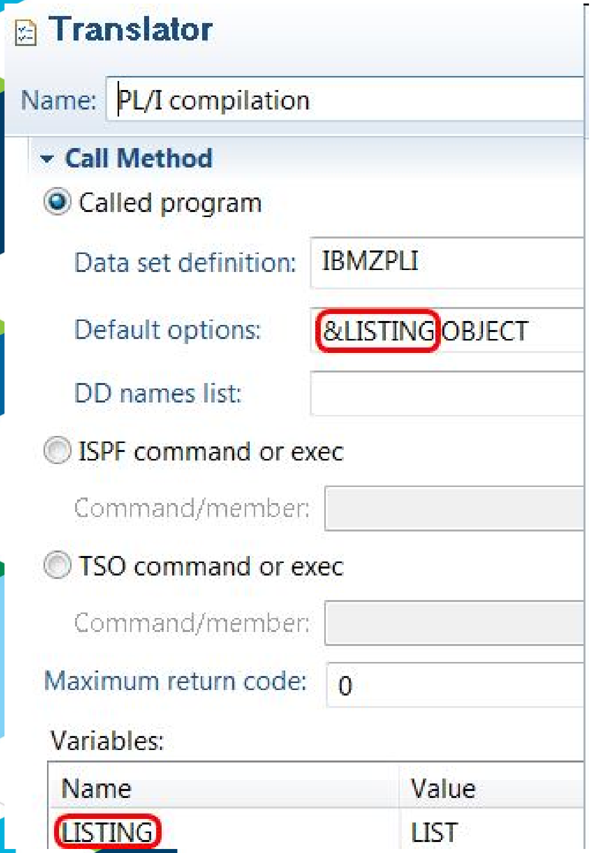
DD Name	Data Set Definitions
SYSLIB	JKE Copybooks,JKE CICS.SDFHCOB
TASKLIB	JKE COBOL.SIGYCOMP,JKE CICS.SDFHLO...

DD allocations:

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

Translator Variables

- Variable overrides



Translator

Name: PL/I compilation

Call Method

[✓ Called program]

Data set definition: IBMZPLI

Default options: &LISTING OBJECT

DD names list:

ISPF command or exec

Command/member:

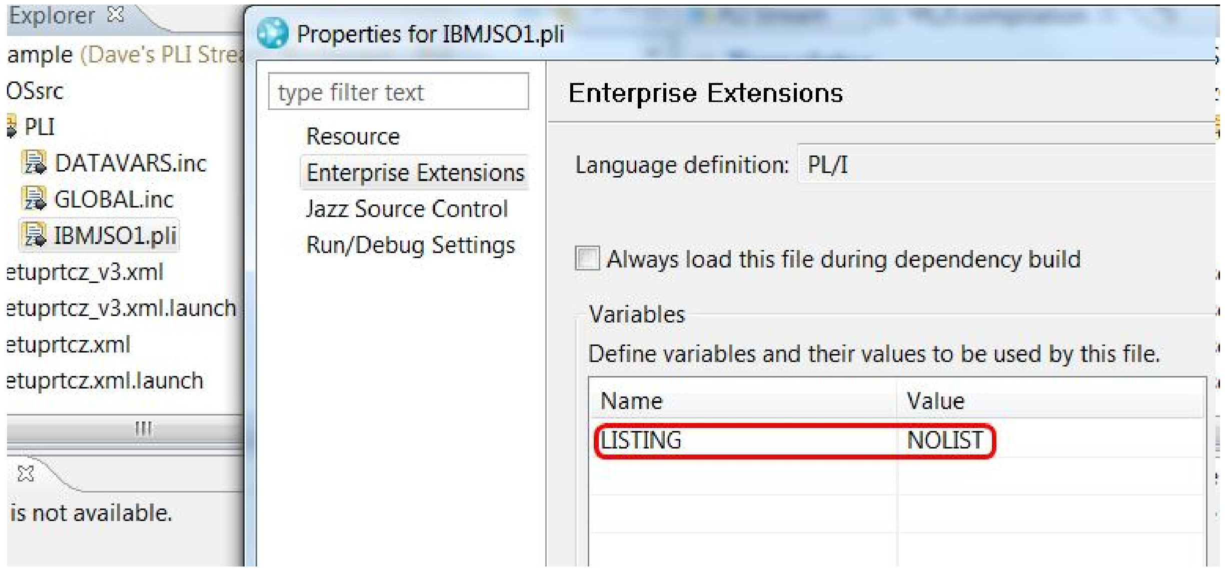
TSO command or exec

Command/member:

Maximum return code: 0

Variables:

Name	Value
LISTING	LIST



Properties for IBMJSO1.pli

Enterprise Extensions

Language definition: PL/I

Always load this file during dependency build

Variables

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

Name	Value
LISTING	NOLIST

Migration

- Define the baselines that you want to import so you can capture history in the RTC SCM
- Import a baseline from the legacy system and dispatching them to the right component or project
  - Methods of migration, zimport, ISPF Client
- Iterate to capture the needed history till the current version
- Initially migrate a subset of modules that cover all the different types of source code
  - Use this subset to test your builds to make sure language definitions are correctly defined and that everything that needs to be built actually does

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
- **You can also use the RTC ISPF Client to import a new PDS**

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-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
  - Gotchas
    - Zimport 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

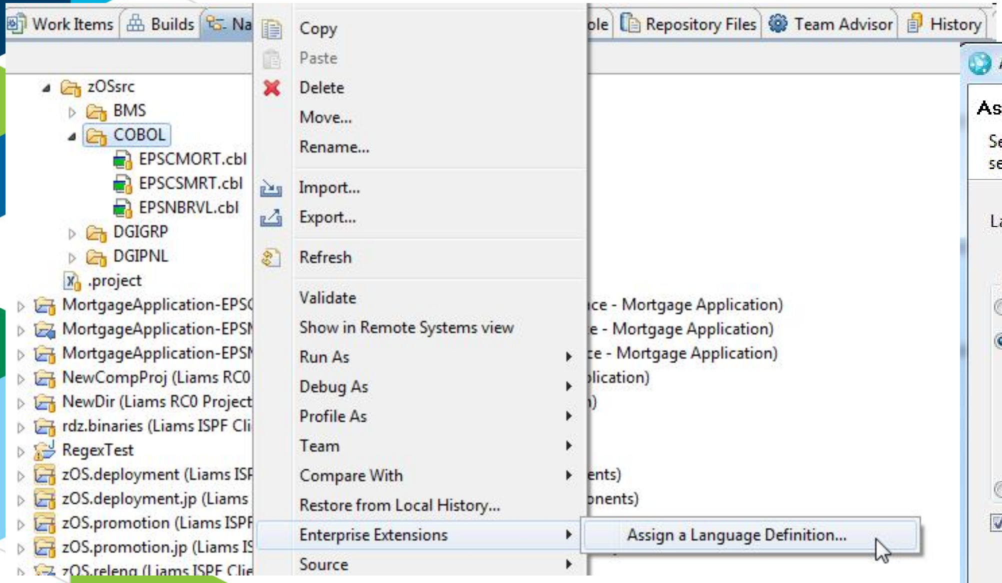
- **What to import?**
  - ▶ All source, recommendation is that the production baseline versions are imported
- Cobol, PL/I, JCL, Procs, etc
  - Adding in all versions will be very costly and time consuming
    - If the IBM services team is engaged they have additional tools to help
- For example - A 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

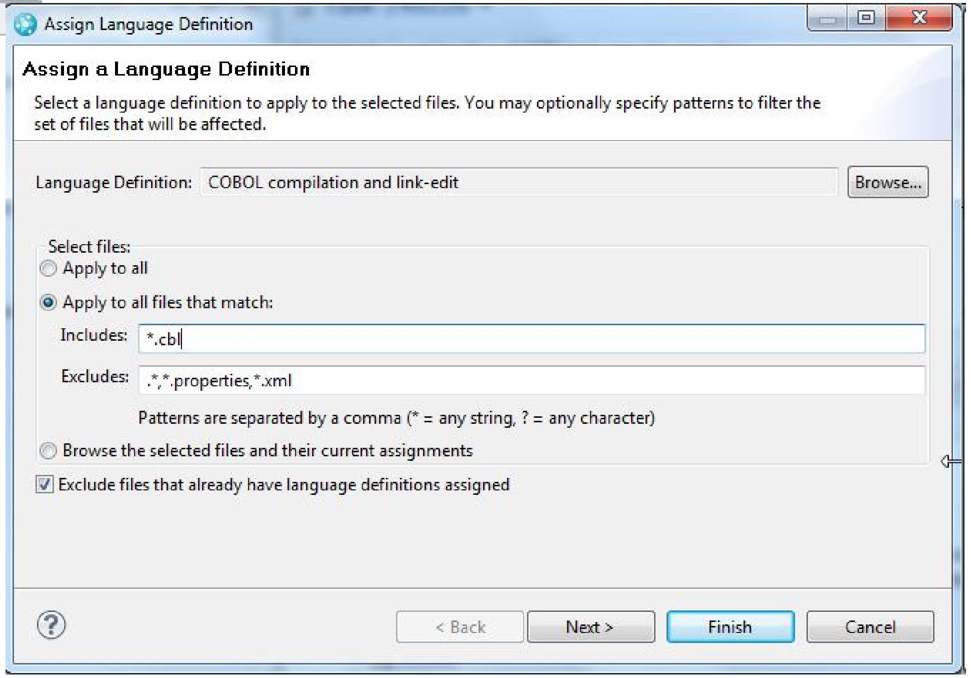
Migrating your source code to Rational Team Concert

- Once zimport has been complete you can set up the rest of your system definitions
  - Create any additional Data set definitions
    - zimport will have created data set definitions for “inputs”
      - COBOL, PL/I, ASM, JCL
    - Use RTC dialogs to create data set definitions for “outputs”
      - OBJ, DBRM, LOAD
    - Also create data set definitions for temporary files

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 if you didn't do that in zimport





Assign Language Definition

Assign a Language Definition

Select a language definition to apply to the selected files. You may optionally specify patterns to filter the set of files that will be affected.

Language Definition: COBOL compilation and link-edit [Browse...]

Select files:

- ( ) Apply to all
- (●) Apply to all files that match:
  - Includes: *.cbl
  - Excludes: .*,*.properties,*.xml
  - Patterns are separated by a comma (* = any string, ? = any character)
- ( ) Browse the selected files and their current assignments

[✓] Exclude files that already have language definitions assigned

< Back | Next > | Finish | Cancel

Post-migration

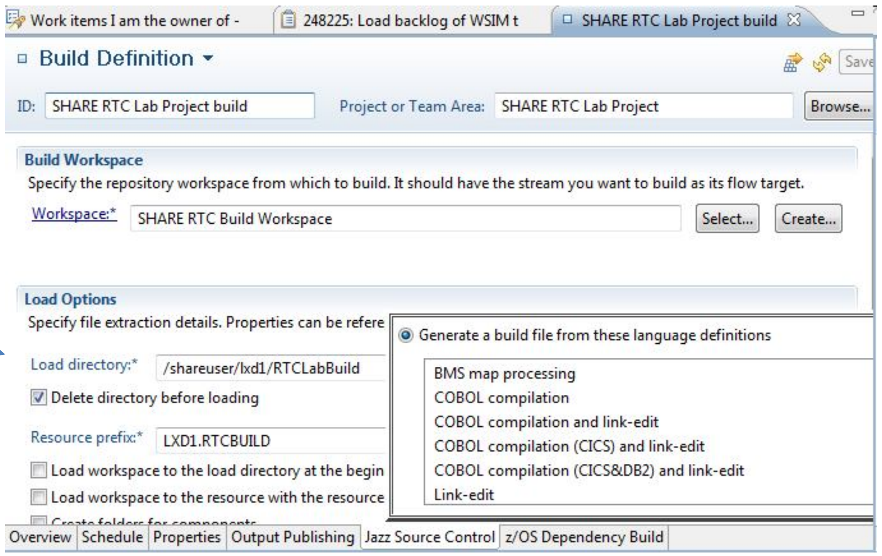
- Migrate your builds
- Tune the system definitions by running builds on a representative subset of your applications
  - You need to ensure all your code can be built and your language definitions are correctly defined
- Prepare the system to trigger dependency builds that involve only the new developments in RTC:
  - Run combination of builds and simulation builds to create the build maps
  - Populate your stream hierarchy
  - Populate the build maps in your stream hierarchy

Migrating your build to Rational Team Concert

- 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 locations

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



Work items I am the owner of - | 248225: Load backlog of WSIM t | SHARE RTC Lab Project build

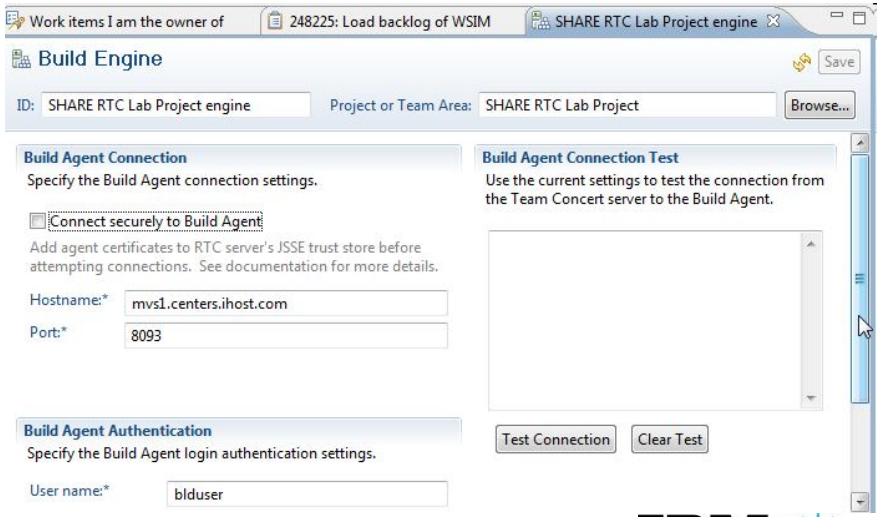
Build Definition

ID: SHARE RTC Lab Project build
Project or Team Area: SHARE RTC Lab Project

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 Build Workspace

Load Options
Specify file extraction details. Properties can be refere
Load directory:* /shareuser/lxd1/RTCLabBuild
[✓] Delete directory before loading
Resource prefix:* LXD1.RTCBUILD
[ ] Load workspace to the load directory at the begin
[ ] Load workspace to the resource with the resource

(●) Generate a build file from these language definitions
BMS map processing
COBOL compilation
COBOL compilation and link-edit
COBOL compilation (CICS) and link-edit
COBOL compilation (CICS&DB2) and link-edit
Link-edit



Work items I am the owner of | 248225: Load backlog of WSIM | SHARE RTC Lab Project engine

Build Engine

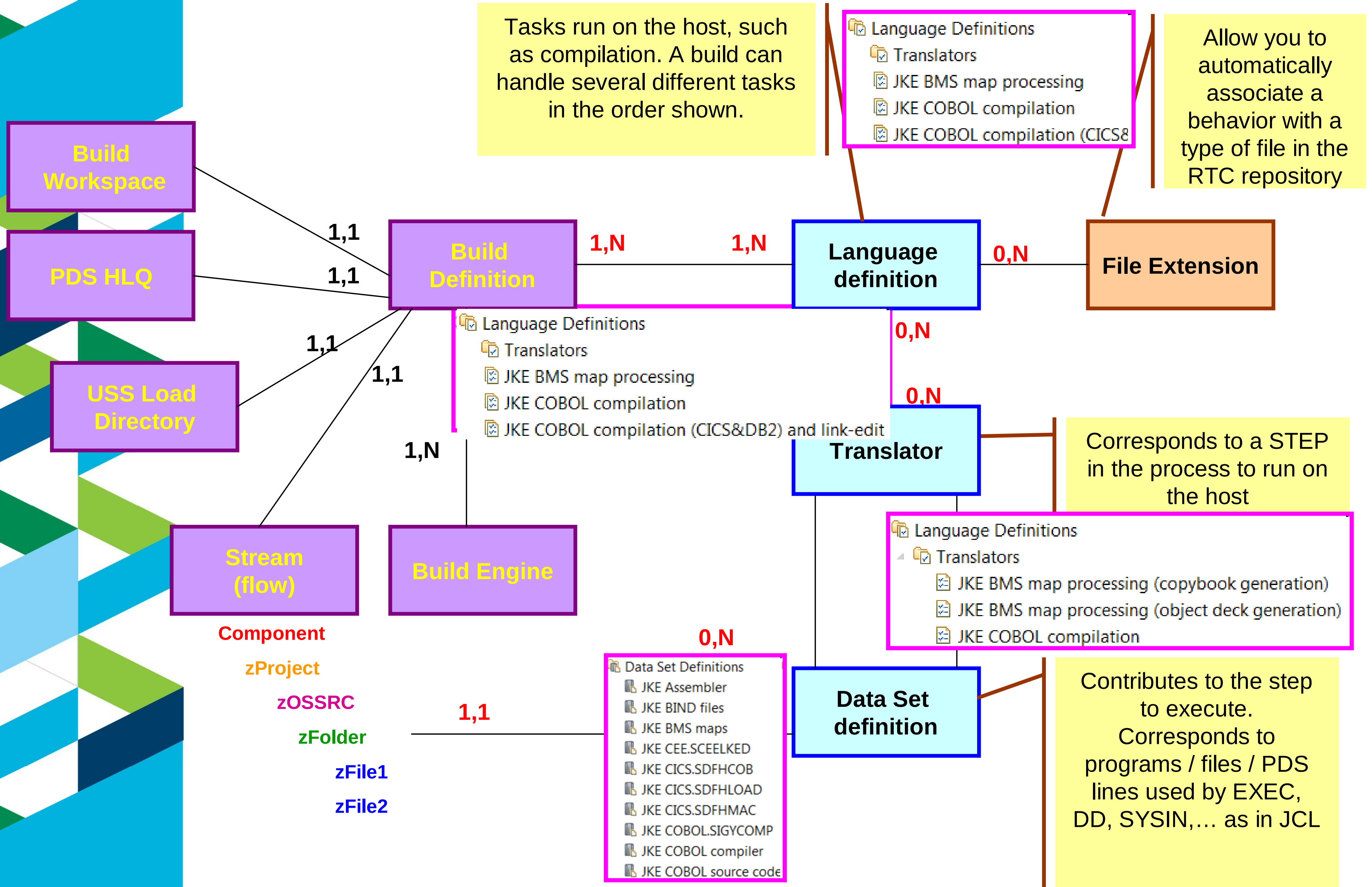
ID: SHARE RTC Lab Project engine
Project or Team Area: SHARE RTC Lab Project

Build Agent Connection
Specify the Build Agent connection settings.
[ ] Connect securely to Build Agent
Add agent certificates to RTC server's JSSE trust store before attempting connections. See documentation for more details.
Hostname:* mvs1.centers.ihost.com
Port:* 8093

Build Agent Connection Test
Use the current settings to test the connection from the Team Concert server to the Build Agent.

Build Agent Authentication
Specify the Build Agent login authentication settings.
User name:* blduser

How it all hangs together



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

Allow you to automatically associate a behavior with a type of file in the RTC repository

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

Language Definitions
- Translators
- JKE BMS map processing
- JKE COBOL compilation
- JKE COBOL compilation (CICS8

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

Language Definitions
- Translators
  - JKE BMS map processing (copybook generation)
  - JKE BMS map processing (object deck generation)
  - JKE COBOL compilation

Data Set Definitions
- JKE Assembler
- JKE BIND files
- JKE BMS maps
- JKE CEE.SCEELKED
- JKE CICS.SDFHCOB
- JKE CICS.SDFHLOAD
- JKE CICS.SDFHMAC
- JKE COBOL.SIGYCOMP
- JKE COBOL compiler
- JKE COBOL source code

Simulation Build

New in
RTC 4.0.4

- When you initially migrate you may not want to rebuild all your source
  - This is a time consuming task
  - If you rebuild you really need to retest
- You only want to build things that have changed since your migration
- Simulation build will go through your code and create/update build maps so that the code you migrate looks current
- Any subsequent changes will then force a rebuild of just the changed modules

Additional Considerations

- Security
  - **https://jazz.net/wiki/bin/view/Main/ZosBuildAgentSec**
  - http://publib.boulder.ibm.com/infocenter/clmhelp/v3r0m1/index.jsp?topic=%2Fcom.ibm.team.build.doc%2Ftopics%2Fr_antz_security.html
  - https://jazz.net/wiki/bin/view/Main/DependencyBuildScenarioOpenSSLSetup
- ISPF Client set up
  - https://jazz.net/help-dev/clm/topic/com.ibm.jazz.install.doc/topics/c_client_ispf_installation.html
- 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
  - http://www.ibm.com/developerworks/rational/library/migrate-rational-team-concert-zos-application-development/index.html

http://www.ibm.com/software/fr/rational/