• z/OS Batch Runtime allows for interoperability between PL/I, COBOL, and Java applications that run on z/OS.
- It also allows for applications to be re-engineered to use the IBM Java Batch Common Programming Model Support
- It is a program designed to provide a managed environment that enables shared access to a DB2 connection by PL/I, COBOL and Java programs.
- Updates to DB2 are committed in a single transaction.

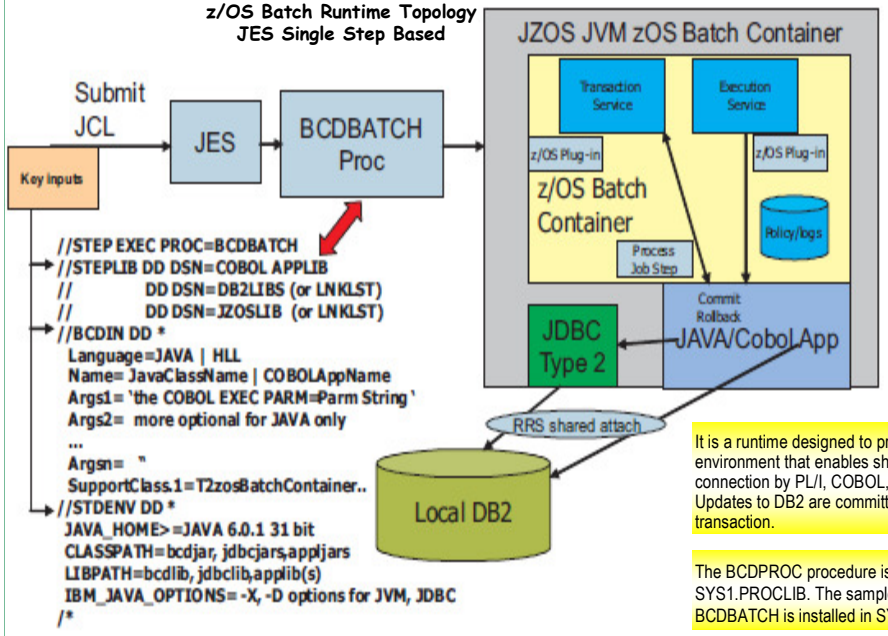
Note: Updates to multiple databases are not supported.

Keep your heritage of existing z/OS PL/I, COBOL, and Java batch applications, while taking advantage of the larger developer skill base and many language features of Java.

• The figure below shows a high-level overview of the z/OS Batch Runtime environment.
- The batch container performs the initialization that sets up the environment for PL/I, COBOL, Java, and DB2 interoperability.
- This includes the following tasks:
> Setting up the proper Language Environment for the PL/I or COBOL programs to run
> Setting up the job step under the umbrella of a Resource Recovery Services (RRS)-managed global transaction
> Initiating the DB2 JDBC driver in this special "Batch Container" mode Invoking the DB2 JDBC driver to create a DB2 connection and attachment thread to Invoke the primary PL/I, COBOL or Java application after the environment is properly initialized.

z/OS Batch Runtime is provided as part of z/OS V2R1

z/OS Batch Runtime Topology
JES Single Step Based



A new feature of z/OS Batch Runtime is support for the IBM Java Batch Programming Model. It enables pure Java batch applications to be both data source and sink neutral with an XML like definition.

z/OS Batch Runtime allows for inter-operability between PL/I, COBOL, and Java applications that run on z/OS.

It is a runtime designed to provide a managed environment that enables shared access to a DB2 connection by PL/I, COBOL, and Java programs. Updates to DB2 are committed in a single transaction.

The BCDPROC procedure is installed into SYS1.PROCLIB. The sample JCL BCDBATCH is installed in SYS1.SAMPLIB.

Support for launching PL/I applications

• In z/OS V2R1, the batch runtime is enhanced to include PL/I as a supported language.
- When mixing COBOL, PL/I, and Java programs in a batch environment, the z/OS Batch Runtime provides a managed environment to govern the different programming models being used.
- Just as with the IBM COBOL support, the environment is bound and centered around traditional JES submitted job steps.
• In particular, this managed environment provides a framework and APIs to enable shared access to a local DB2 for z/OS database connection by COBOL, PL/I, and Java programs.
- Updates to the database across language boundaries are committed within a single RRS managed transaction scope.
- Also new in z/OS V2R1 is transparent support for Transactional VSAM (TVS) across these same language environments.
• Support is added to allow the z/OS Batch Runtime to launch an Enterprise PL/I main routinesimilar to the launch of COBOL that was done in V1R13.

The Batch Container is implemented in Java and installs into existing path /usr/lpp/bcp.

IBM SDK for z/OS, Java Technology Edition, V6.0.1 or higher is required.

Usage

• Identical to the IBM COBOL support, z/OS Batch Runtime control statements are read from the BCDIN DD defined in the batch JCL
- The **bcd.applicationLanguage=** statement is changed to accept the new LE value to indicate the program being launched is written in either COBOL or PL/I.
- The COBOL language value is still accepted, but the new LE value is the preferred syntax.
- The batch runtime processes both COBOL and PL/I applications in the same way. An example is shown in the figure below.

```
//BCDIN DD *
bcd.applicationLanguage=LE
bcd.applicationName=PLITEST
```

{ Invoke a PL/I program using Batch Runtime

When planning use of z/OS Batch Runtime, a good application to consider using is a native procedural z/OS COBOL or PL/I application that you want to functionally enhance with Java method calls.

• The z/OS Batch Runtime launches an application by fetching and calling it.
- As a result, the PL/I external procedure being launched must specify the "fetchable" and "assembler" options.
- The "main" option cannot be used. So at a minimum, any launched PL/I "Main" routine must be at least slightly modified as shown in this figure:

```
PLITEST: Procedure Options( Fetchable
                            Assembler );
```

{ Specifying the Fetchable and Assembler options

.- Additionally, the assembler option is also needed so that any launched PL/I "Main" application can set a return code using the PLIRETC function upon return to the z/OS Batch Runtime.

• When binding the PL/I application, the ENTRY CEESTART must not be used ; instead, the entry point should specify the name of the external procedure.
- For example, the procedure shown in the previous figure would be bound as shown in figure on the right:.

```
//BIND.SYSIN DD *
ENTRY PLITEST
NAME PLITEST(R)
```

{ Binder options to specify a name for the entry point

Commit and Rollback helper functions

• To simplify the process of calling the batch runtime commit and rollback routines, the batch runtime is providing the convenience methods bcdcommit() and bcdrollback() that can be called directly from a PL/I application.
- Use of the helper functions replaces the JNI calls to callback to the batch container
• The methods reside in a *new* DLL **libbcduser.so** that will be shipped with z/OS V2R1.
- For PL/I callers, an include file is provided in SYS1.SAMPLIB(BCDPLIH) that defines the entry points.

Installation

• z/OS Batch Runtime is provided as part of z/OS V2R1.
- The batch Container is implemented in Java and installs into existing path **/usr/lpp/bcp**.
- The BCDPROC procedure is installed into **SYS1.PROCLIB**; The sample JCL BCDBATCH is installed in **SYS1.SAMPLIB**.

IBM batch programming model support

• Further enhancements are made in z/OS V2R1 Batch Runtime to support the IBM Batch Programming Model.
Note: This support, fully documented in **Appendix A of *z/OS Batch Runtime:***
***Planning and User's Guide*, SA23-1376**, is for JES submitted new Java applications described in an XML like policy (termed xJCL) and following the rules intrinsic to the IBM Batch Programming Model. This is the *same* descriptive language used in IBM Websphere Batch support, albeit with a traditional JES submission and limited to a single non-persistent JVM per job step. With this support applications can be deployed on WAS distributed, WAS z/OS, or if compliant with limitations of z/OS Batch Runtime, a z/OS batch job.

Invocation and Restart

• In the IBM batch programming model, a job is described by the xJCL language.
- Use Rational® Application Developer (RAD) xJCL editor to create xJCL files. xJCL is an XML based file which describes the batch job and its steps.
- Each batch step contains business logic to run a portion of the job and typically reads a record, performs business logic then gets the next record.
• To support the IBM batch programming model, z/OS Batch Runtime procedure, **BCDPROC**, now supports two new DD cards
- The **BCDXJCL DD** names a file containing the xJCL.
> This file usually resides in a file system.
- The **BCDPROP DD** names a file containing substitution properties to be applied to xJCL
>. This file is usually provided inline within the job.

NOTE: BCDBATCH sample JCL is provided in **SYS1.SAMPLIB** and shows how to use the new DD cards.
• To run a Java program using the IBM batch programming model, **bcd.applicationLanguage=XJCL** must be specified under the BCDIN DD.
- The **bcd.xJCLEncoding=encoding-name** option can be used to set the xJCL file encoding.
NOTE: The default is ISO8859-1.
• Job restart is supported through the check pointing process.
- The xJCL defines a checkpoint policy that determines when a checkpoint is to be taken.
> When a checkpoint occurs, the Batch Runtime initiates a commit sequence.
• If a step *fails*, the job is terminated.
- When restart is requested, the Batch Runtime resumes the job at the failing step and restart processing from last commit.
- The input and output to the step is repositioned such that processing can restart based on the last committed records processed.

NOTE: By default, restart is not enabled for a job. Use the **bcd.xJCLRestartEnabled** control statement to enable the job for restart. Enabling this option will cause the Batch Runtime to write persistence data to the file system which is used to keep track of state.

• The restart jobid is not the same as the JES jobid.
- When the job is submitted, an internal jobid is created and used by the Batch Runtime to manage the job.
- The jobid assigned is contained in message BCD0310I when the job starts. For a restart, use this jobid on the **bcd.xJCLRestartJobId** statement.

```
//BATCH EXEC BCDPROC,REGION=0M,LOGLVL='+T',VERSION=61
//*
//BCDXJCL DD PATH='yourpathxJCL.xml',
// PATHOPTS=ORDONLY
//*
//BCDXPROP DD *
inputDataStream=inputFile.txt
outputDataStream=outputFile.txt
fileEncoding=IBM-1047
debugEnabled=true
//BCDIN DD *
bcd.applicationLanguage=XJCL
bcd.xJCLEncoding=ibm-1047
bcd.xJCLRestartEnabled=true
#
bcd.supportClass.1=com.ibm.db2.jcc.t2zos.T2zosBatchContainerSupport
```

The illustration on the left displays a fragment of the batch JCL used to invoke an xJCL job. See the format of the BCDXPROP DD input where each line specifies a substitution property name and its value.

These correspond to the property names defined in the xJCL. When the xJCL is processed, the values specified here will override those defaulted in the xJCL.