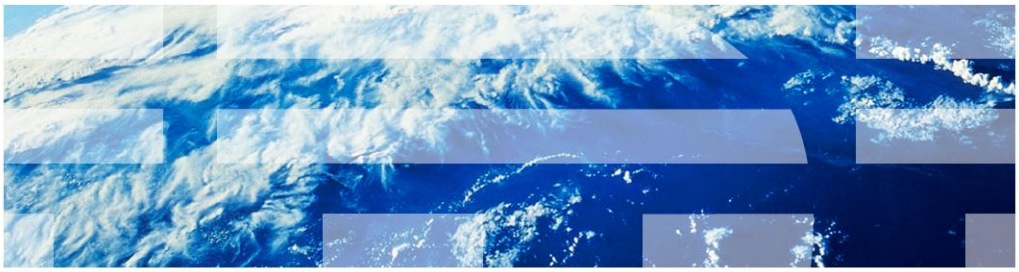
IBM WebSphere Application Server

Batch applications overview



This module provides an overview of the batch applications feature of IBM WebSphere Application Server.

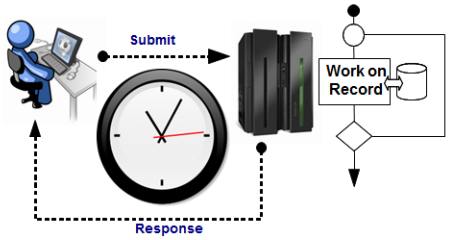
Batch processing

Comparison of online transaction processing and batch processing

Online Transaction Processing

- **A request / receive model**
  (Often but not necessarily synchronous)
- **Duration of processing relatively short**
- **Often transactional in nature**
- **Application server runtime enforces timeouts for workload**

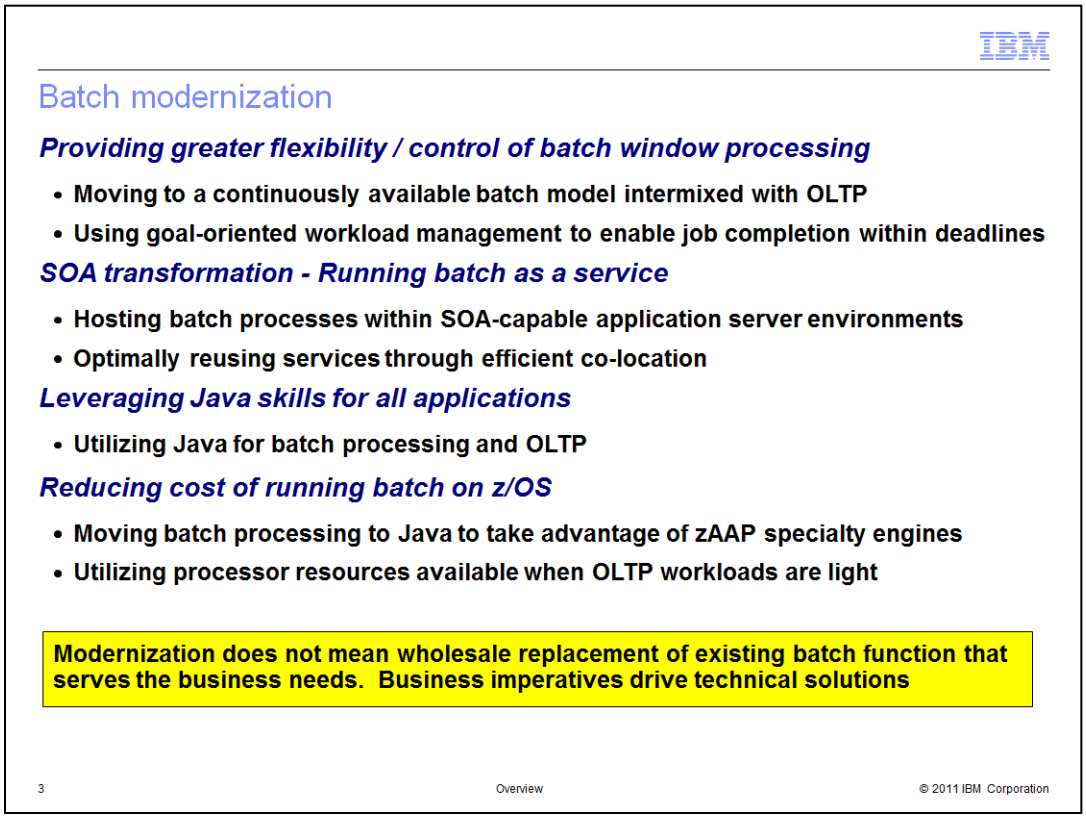
Batch Processing
(or Long Running)



- **A submit / work / result set model**
  (Asynchronous to the submitter)
- **Duration of processing a function of work to be done; can be hours/days**
- **Often transactional in nature**
- **Often multi-step processes**

Online transaction processing is a request/receive model where the duration of the processing is relatively short, and the tasks are typically transactional in nature. In this model, the application server runtime enforces timeouts for the workload.

Batch processing is a submit/work/result set model where the duration of the processing is a function of the tasks to be completed. In some cases, the tasks can require hours or even days to complete. In this model, the work tasks are typically transactional in nature and typically involve multi-step processes.



Batch modernization involves these techniques:

-Providing greater flexibility and control of batch window processing by moving to a continuously available batch model intermixed with online transaction processing and using goal-oriented workload management to enable job completion within deadlines.

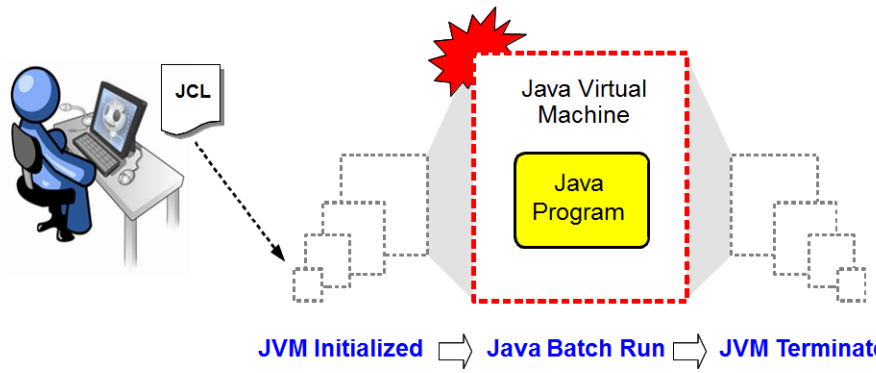
-Hosting batch processes within service-oriented architecture (SOA) capable WebSphere Application Server environments and optimally reusing services through efficient co-location.

-Leveraging Java skills for batch and online transaction processing tasks.

-On the z/OS platform, reducing costs by moving batch processing to Java to take advantage of System z Application Assist Processor (zAAP) specialty engines and utilizing processor resources available when online transaction processing workloads are light.

Use JVM launcher

Many attempt to use a JVM launcher to run Java batch programs



JVM Initialized ⇨ Java Batch Run ⇨ JVM Terminated

This approach works, but has limitations:

- High overhead associated with repeated cycling of JVMs
- Application must provide most batch processing services

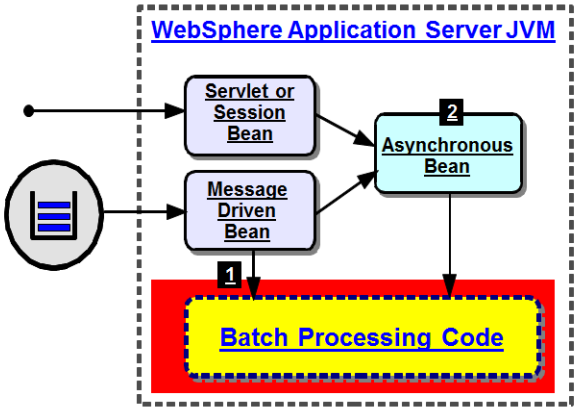
4 Overview

Java Virtual Machine (JVM) launcher technologies are easy to use and free, and the programming interfaces are very useful; however, a new JVM is created for every step, and the resulting overhead is significant. In addition, JVM launcher technologies do not provide a transaction manager. Ultimately, the application developer must author a transaction manager within their application.

On the z/OS platform, WebSphere Application Server and WebSphere XD Compute Grid complement the Java Batch Toolkit for z/OS (JZOS). The JZOS launcher can be used when only a few Java batch steps must be executed. Once the size of the Java batch infrastructure grows to the point where the overhead of creating and terminating JVMs is an issue, WebSphere Application Server and WebSphere XD Compute Grid can be used. The second component of JZOS, it's API's for accessing z/OS resources, can be used from WebSphere Application Server and WebSphere XD Compute Grid applications.

Use application server runtime

A second approach uses the application server runtime and authors the batch application as a message driven bean or an asynchronous bean.



1. When a message driven bean is used, the application must manage the timeout on the message driven bean input
2. The preferred technique (used by WebSphere Application Server and WebSphere XD Compute Grid) is to use an asynchronous bean

In both cases, the application developer must author custom middleware to support the batch processing

A second approach uses the application server runtime and authors the batch application as message driven bean or an asynchronous bean. When a message driven bean is used, the application must manage the timeout on the message driven bean input; therefore, using an asynchronous bean is the preferred technique. However, in both cases, the application developer must author custom middleware to support the batch processing, given that the solution lacks these features:

-A batch programming model, job definition language, and supporting development tools

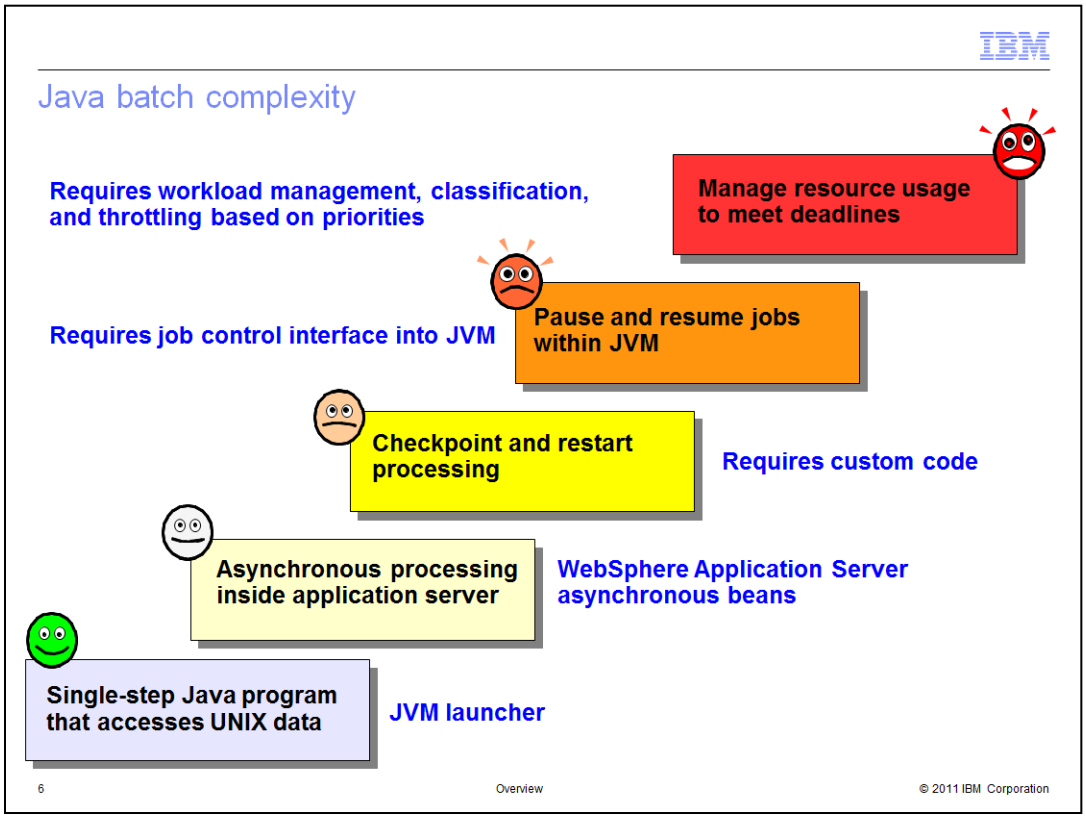
-Interweaving of online transaction processing tasks and batch tasks

-Checkpointing and restarting of jobs

-Integration with enterprise schedulers

-Graphical and command line interfaces for job monitoring and management

-And Job usage accounting and activity logging



Initial Java batch programming projects typically begin with a relatively simple Java Virtual Machine (JVM) launcher design, but it grows more complex over time as additional management features are required. These projects frequently progress in this manner:

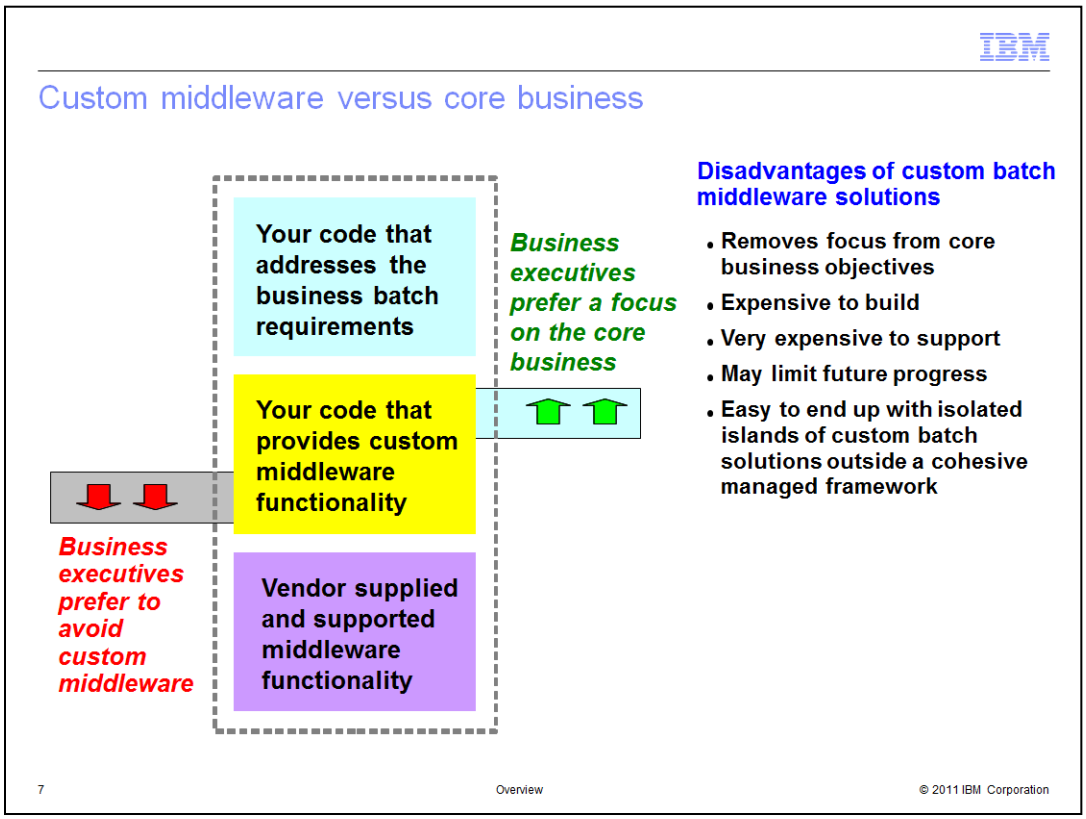
-An organization uses the Java command line interface or the Java Batch Toolkit for z/OS (JZOS) to run a single-step Java program that access UNIX data.

-After realizing that transaction management capabilities are required, the organization moves the application to a Java Enterprise Edition (Java EE) application server.

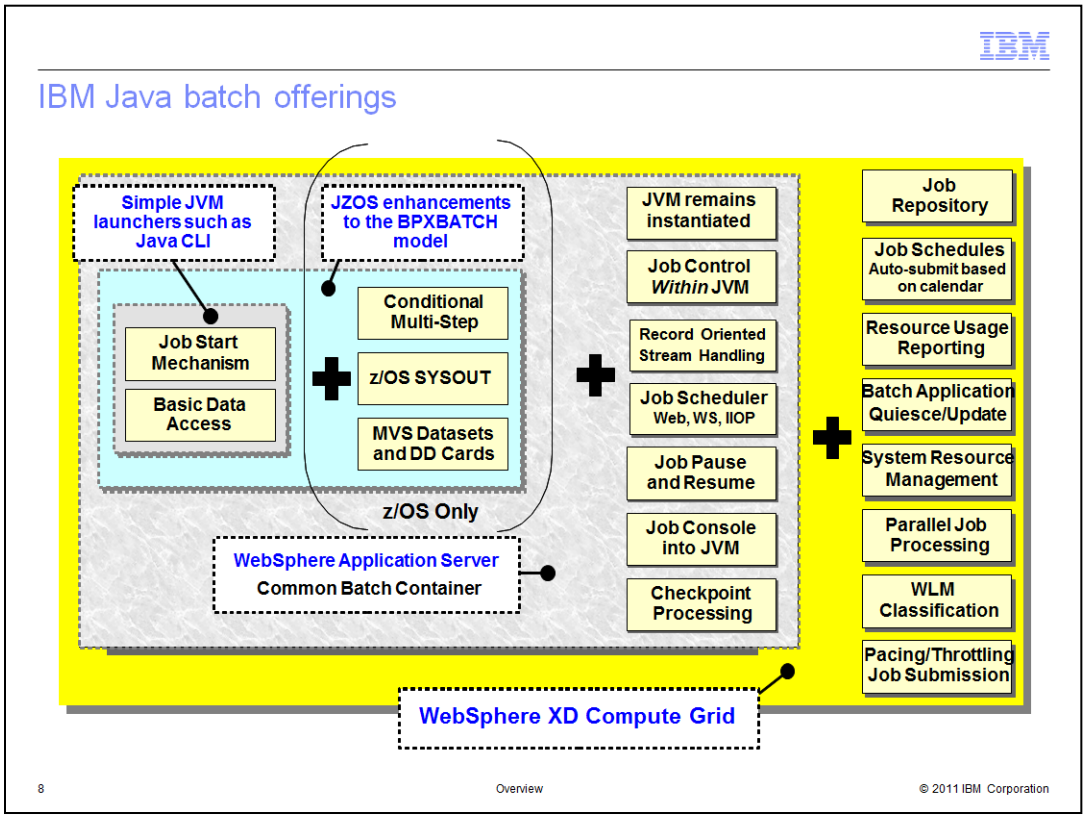
-As the organization begins to process large numbers of records as a single batch job, it finds that it is no longer practical to restart entire jobs when failures occur in the middle of a job. The organization then introduces checkpointing and restart capability.

-A large number of jobs in the batch environment compete for the same resources, and administrators need a way to manage the jobs and handle resource conflicts. The organization then builds a job control interface into the JVM that processes the batch applications.

-As batch loads increase even further, it becomes impossible to manually handle resource conflicts; as a result, the organization now requires a batch job management system that can optimally distribute workloads based on priorities and deadlines.



Organizations are frequently tempted to solve tactical issues with custom middleware code; however, the middleware code requires extensive time to develop and maintain, which ultimately decreases an organization's ability to develop custom code that addresses core business objectives.



Java batch tasks can be managed in several different ways:

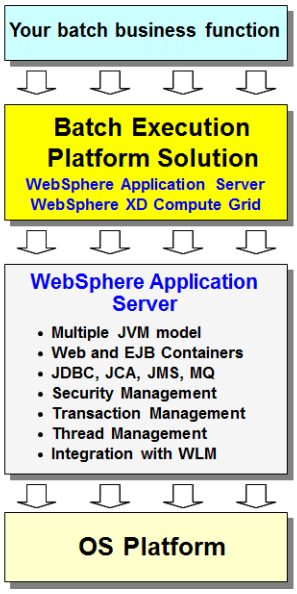
-Simple Java Virtual Machine (JVM) launchers can manage single step batch jobs that require basic data access.

-On the z/OS platform, the Java Batch Toolkit for z/OS (JZOS) enhances the BPXBATCH model by supporting conditional multi-step batch jobs with access to MVS datasets and use of Data Definition (DD) cards.

-WebSphere Application Server provides multi-step job support, a managed container for execution of batch jobs, a job control interface, job checkpoint and restart capability, and a batch application development framework.

-Finally, WebSphere XD Compute Grid adds support for job repository and schedules, workload management, job usage reporting, batch application quiesce and update, parallel job support, and pacing and throttling of jobs.

Java batch foundation



Key benefits

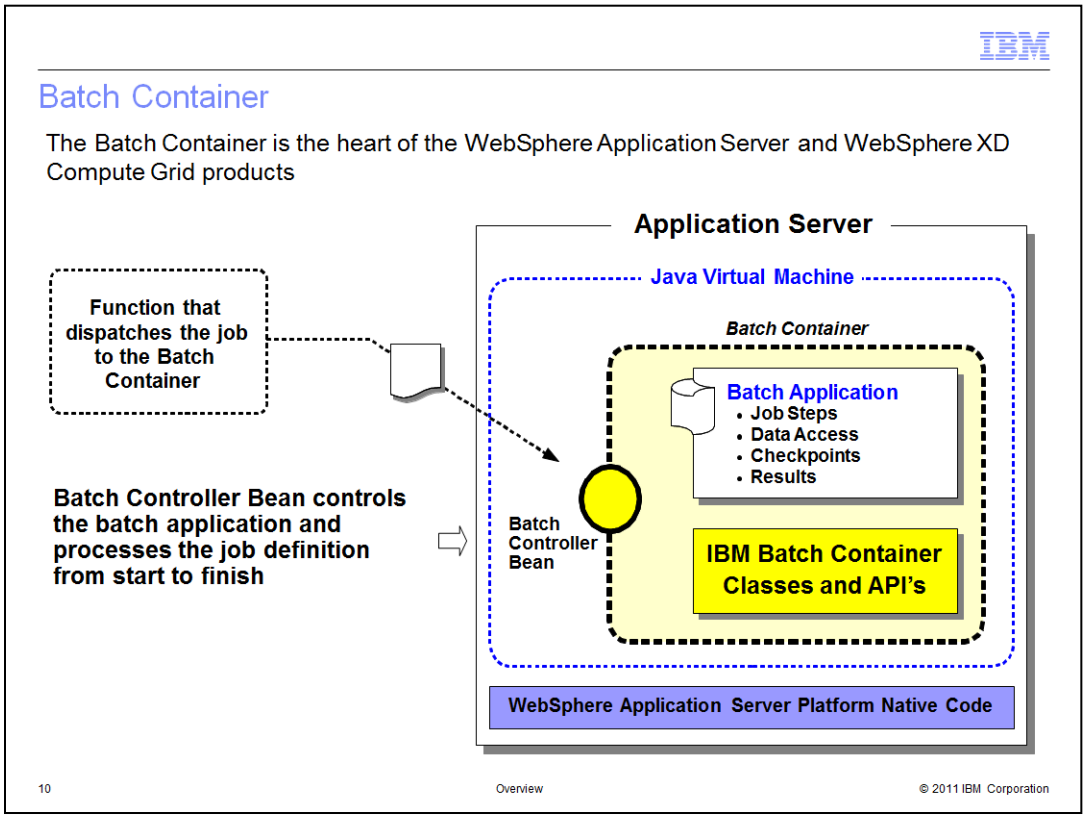
- Custom middleware code is no longer required, so developers can stay focused on core business imperatives
- Online Transaction Programming (OLTP) and batch loads can be mixed while maintaining respective service level agreements

The basic IBM Java batch foundation includes WebSphere Application Server. Customers demanding advanced workload management, accounting, and job scheduler integration capabilities can use WebSphere Application Server and WebSphere XD Compute Grid.

Both solutions provide these benefits:

-Custom middleware code is no longer required, so developers can stay focused on core business imperatives.

-Online Transaction Programming (OLTP) and batch loads can be mixed while maintaining the service level agreements for each type of traffic.



The Batch Container is the heart of the batch application support provided in the WebSphere Application Server and WebSphere XD Compute Grid offerings. It runs a batch job under the control of an asynchronous bean, which can be thought of as a container-managed thread. The batch container ultimately processes a job definition and carries out the life cycle of a job.

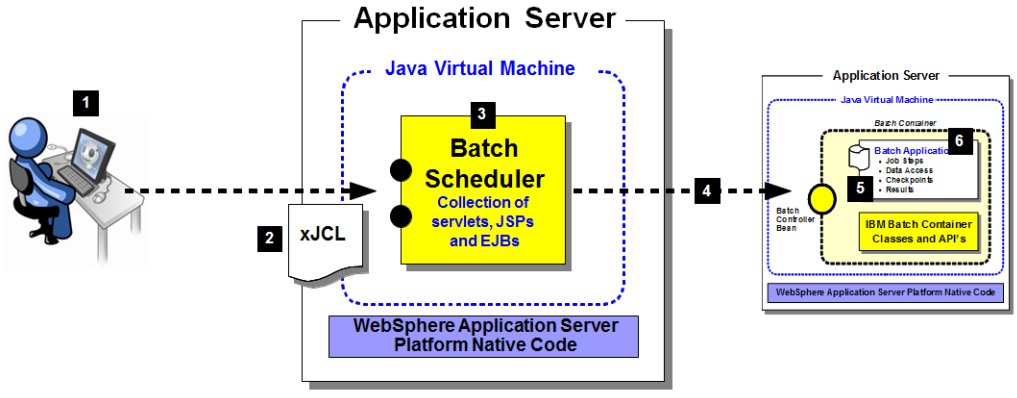
The Batch Container provides these services:

-Checkpointing, which involves resuming batch work from a selected position.

-Result processing, which involves intercepting and processing step and job return codes.

-And Batch data stream management, which involves reading, positioning, and repositioning data streams to files, relational databases, native z/OS datasets, and many other different types of input and output resources.

Batch components and workflow



1. Job is submitted
2. Job Control Definition is specified
3. Scheduler analyzes the request
4. Job is dispatched to batch endpoint
5. Batch endpoint begins execution
6. Batch application is invoked

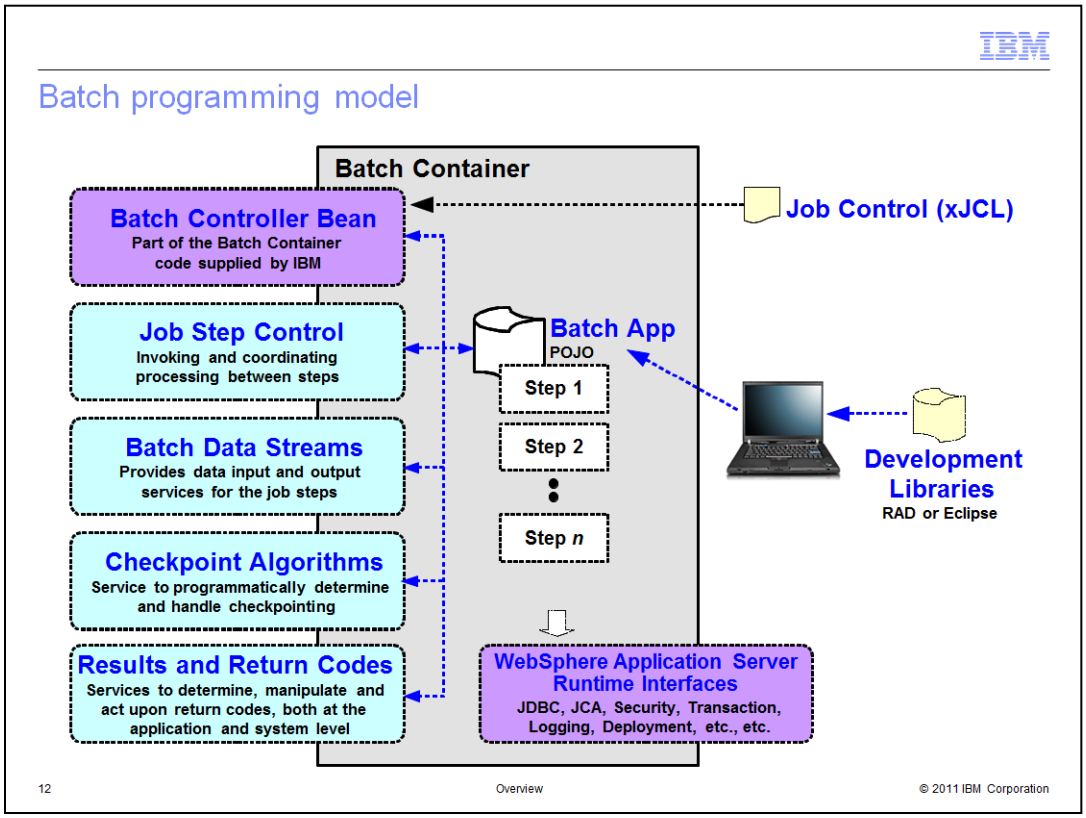
-Batch jobs are submitted to the system using the Job Management Console or programmatically by way of Enterprise Java Beans (EJB), Java Message Service (JMS), or web services.

-Each job is submitted in the form of an XML Job Control Language (xJCL) document.

-The Job Dispatcher selects the best endpoint for job execution based on several different metrics.

-The endpoint sets up the jobs in the Batch Container and executes the batch steps based on the definitions in the xJCL.

-The Job Dispatcher aggregates job logs and provides life-cycle management functions like start, stop, cancel, and so on.



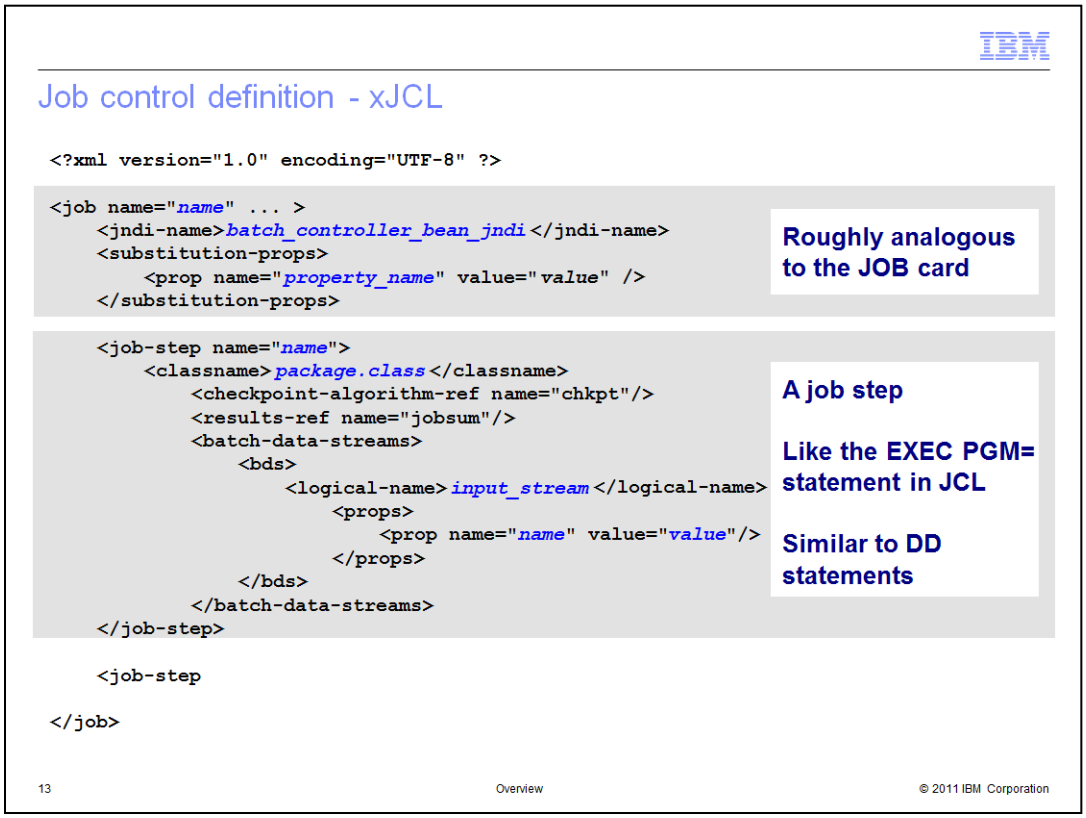
The batch programming model consists of four principal interfaces, two of which are essential to building a batch application, and two which are optional and intended for advanced scenarios:

-The batch job step defines the interaction between the batch container and the batch application.

-The batch data stream abstracts a particular input source or output destination for a batch application and defines the interaction between the batch container and a concrete BatchDataStream implementation.

-A checkpoint policy algorithm defines the interaction between the batch container and a custom checkpoint policy implementation. A checkpoint policy is used to determine when the batch container will checkpoint a running batch job to enable a restart to occur after a planned or unplanned interruption. WebSphere Application Server and WebSphere XD Compute Grid include two ready-to-use checkpoint policies.

-A results algorithm defines the interaction between the batch container and a custom results algorithm. The purpose of the results algorithm is to provide the overall return code for a job. The algorithm has visibility to the return codes from each of the job steps. WebSphere Application Server and WebSphere XD Compute Grid include one ready-to-use results algorithm.

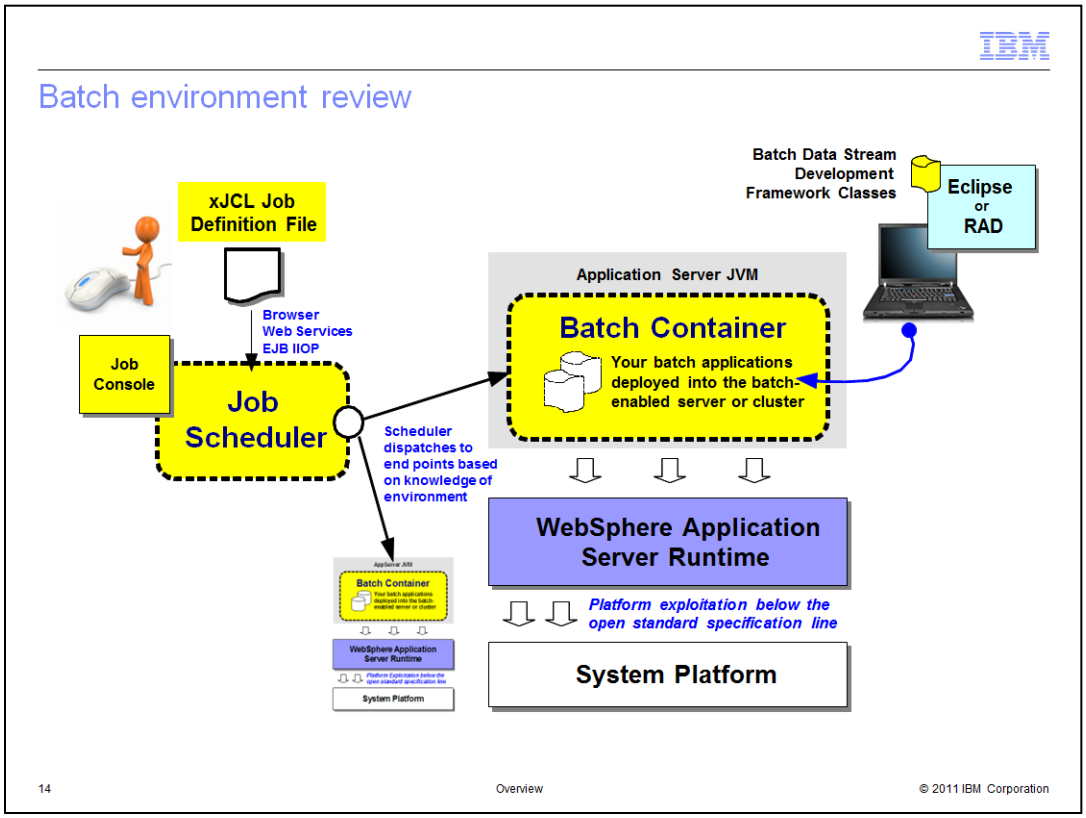


Jobs are expressed using an Extensible Markup Language (XML) dialect called XML Job Control Language, or xJCL.

The xJCL definition is very similar to the traditional JCL.

The xJCL definition of a job is not part of the batch application, but is constructed separately and submitted to the job scheduler to run. The job definition identifies which batch application to run, and it's inputs and outputs. It also identifies which checkpoint algorithms and results algorithms to use.

The Job Scheduler uses information in the xJCL to determine where and when the job should be run.

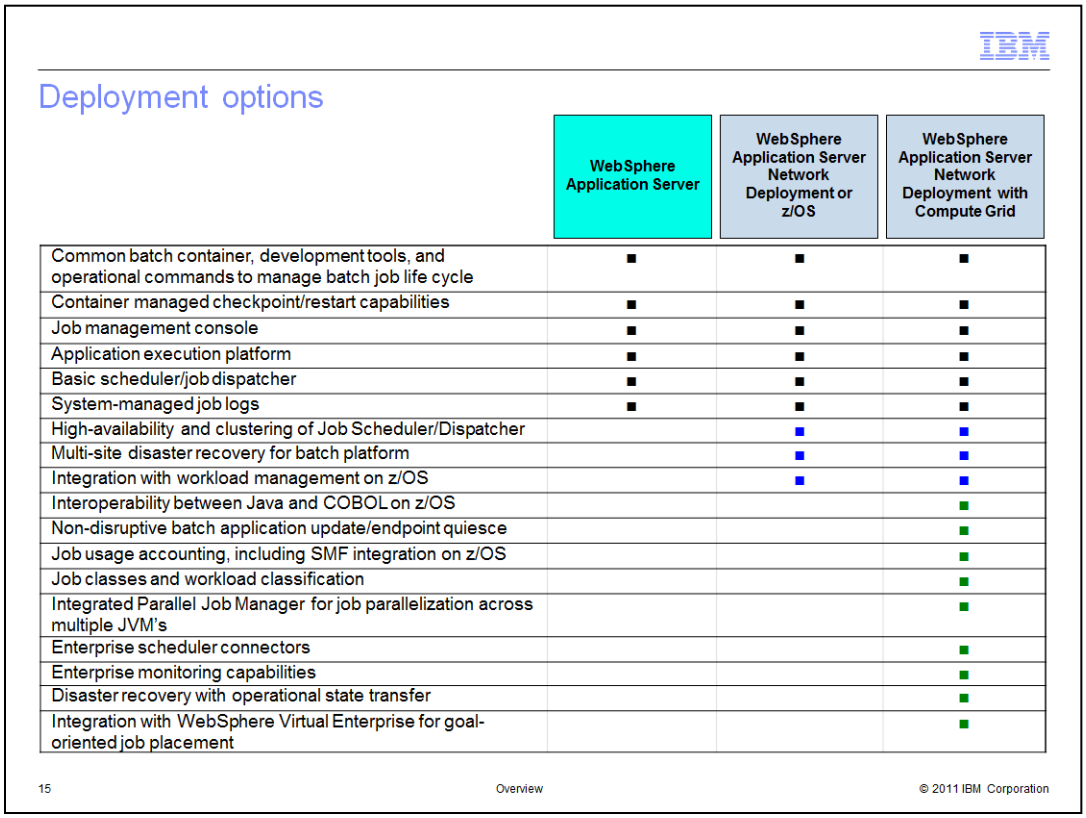


The Job Scheduler provides the job management functions such as submit, cancel, restart, and so on. It maintains a history of all job activity, including waiting jobs, and running jobs, and completed jobs. The job scheduler is hosted in a WebSphere Application Server or a server cluster.

Jobs are described using job control language called XML Job Control Language (xJCL), which identifies the batch application to run, it's inputs and outputs.

The Batch Container provides the execution environment for batch jobs. There can be multiple Batch Containers in a WebSphere cell.

Batch applications are regular WebSphere Java Enterprise Edition (Java EE) applications, deployed as Enterprise Archive (EAR) files.



IBM offers several different Java batch environments:

-WebSphere Application Server Base Edition

-WebSphere Application Server Network Deployment Edition

-WebSphere Application Server for z/OS

And WebSphere XD Compute Grid

WebSphere Application Server Base Edition provides these features:

-A managed batch container

-A batch application framework

-A job management console

-A job dispatcher

-AndSystem-managed job logs

WebSphere Application Server Network Deployment Edition and WebSphere Application Server for z/OS provides these additional features:

-High-availability and clustering of the job dispatcher

-Multi-site disaster recovery support

-And Integration with z/OS workload management

WebSphere XD Compute Grid provides these features beyond those provided by WebSphere Application Server:

-Job classes and workload classification

-Job parallelization across multiple Java Virtual Machines

-Job usage accounting

-Connectors for integration with enterprise schedulers

-Enterprise monitoring capabilities

-Interoperability between Java and COBOL on z/OS

-Non-disruptive batch application update support

-And Integration with WebSphere Virtual Enterprise for goal oriented job placement

Trademarks, disclaimer, and copyright information