WebSphere Application Server for z/OS
- Batch Update -

John Hutchinson
IBM

August, 2011 - Orlando
SHARE Session 09486
# WebSphere Application Server Sessions

<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>#</th>
<th>Title</th>
<th>Speaker</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wednesday</td>
<td>3:00</td>
<td>9483</td>
<td>Using IBM's New Cross-Platform Installer on z/OS</td>
<td>Mierzejewski (Loos)</td>
<td>Oceanic 5</td>
</tr>
<tr>
<td>Thursday</td>
<td>8:00</td>
<td>9482</td>
<td>WAS Version 8 – Overview</td>
<td>Follis</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>9:30</td>
<td>9486</td>
<td>WAS Version 8 – Batch Update</td>
<td>Hutchinson</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>11:00</td>
<td>9485</td>
<td>WAS Version 8 – New z/OS Exploitation/Differentiation Features</td>
<td>Follis</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>1:30</td>
<td>9484</td>
<td>WAS Version 8 – High Availability Enhancements</td>
<td>Follis</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>3:00</td>
<td>9488</td>
<td>WAS - Back to Basics Part 1</td>
<td>Loos</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>4:30</td>
<td>9489</td>
<td>WAS - Back to Basics Part 2</td>
<td>Stephen</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Friday</td>
<td>8:00</td>
<td>9490</td>
<td>WAS for z/OS - Level 2 Update</td>
<td>Stephen</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Friday</td>
<td>9:30</td>
<td>9487</td>
<td>WAS for z/OS – PotPourri</td>
<td>Follis, Hutchinson, Loos, Stephen, etc.</td>
<td>Europe 2</td>
</tr>
</tbody>
</table>
What's New with Batch (Compute Grid) on z/OS?

- **WebSphere Application Server Version 8** includes V7 Batch FeP
  - Sub-set of function in XD Compute Grid
- **WebSphere XD Compute Grid Version 8 Enhancements:**
  - Configuration & Operational Improvements *
  - Programming Framework *
  - Job Structure Enhancements *
  - Integration with JES Schedulers *
  - Parallel Job Management *
  - Job Classification & Control *
  - Job Usage Reporting *
  - Integration with CICS & COBOL *
- **Migration from Version 6.1.1**
- **Information Center & other Resources**
Batch Feature Pack and XD Compute Grid

Here's a summary of the key features:

**Feature Pack for Modern Batch** *(now part of WAS V.8)*

- Batch container environment
- Job scheduler and dispatcher function
- Declarative job control file (xJCL)
- Development class libraries
- Batch Data Stream (BDS)
- Conditional multi-step job support
- Ckpt processing leverages WAS trans. Mgr

**WebSphere XD Compute Grid**

Everything you see under "FP for Modern Batch" plus ...

- Calendar & clock scheduling of jobs
- Integration with ext. scheduler products
- Usage reporting with SMF 120.20 & .9
- WLM transaction classification *by job*
- Application quiesce and update
- Job submission pacing and throttling
- Parallel job management & dispatching
- Integration with COBOL and CICS

---

**New in Compute Grid V. 8 on z/OS!**

- **Programming Model Enhancements**
  - QSGi Batch Applications
  - Record Processing Policy
  - Record Metrics
  - Job and Step Listener
  - Persistent Job Context
  - Configurable Transaction Mode
  - Batch Data Stream Timeout
  - COBOL Support

- **Job Definition Enhancements**
  - Multi-threading
  - Parallel Steps
  - Heterogeneous Steps

- **Operational Enhancements**
  - Group Security
  - Memory Overload Protection
  - Job Log SPI
  - SMF Type 120 Subtype 9
Batch Processing ....

...has been around for a very long time. (30-45 yrs?)

Different programming languages ...

... similar business results.

Several Different Approaches...

- Standalone Java Program
- JVM Launcher – JZOS
- WebSphere Java Batch Container
Consists of a Dispatcher & Endpoint server

xJCL Job Definition File

The job is an instance of program invocation based on the description offered in the xJCL file.

WebSphere Application Server (s)

Dispatcher provides a Control point and View for batch jobs.

Endpoint server runs the batch jobs with a long-running JVM.

Job Scheduler Function

Dispatch and Manage

Joblog

The Dispatcher & Endpoint may be in the Same server, Separate servers, or Clustered across many LPARs. Your choice. 😊
Configuration Improvements

- **WAS Version 8**
  - Includes Modern Batch Feature Pack
  - Installation Manager support (Required)
  - WebSphere Customization Toolbox (WCT) V8 supports WCG Augmentation

- **Use WCT V8 to create and augment a WAS V7 or V8 Cell with WCG V8**
  - Deployment Manager & Empty Node
  - Augment with Compute Grid

![WebSphere Customization Toolbox 8.0](image)

- **Pre-configured in Compute Grid Version 8**
  - Parallel Job Manager
  - PGCPProxy (used by CICS CN11 SupportPac)
  - COBOL Container and JAR files
Configuring WebSphere Compute Grid V8 on z/OS

ND Cell configuration Simplified with WCT Version 8:

1) Create a WebSphere V8 Deployment Manager Augmented with Compute Grid
2) Create Empty Nodes with Compute Grid (includes Augmentation)

Configure the Compute grid Dispatcher & Endpoint Servers:

3) Create Database & Data sources (DBA & ISC)
4) Configure Job Scheduler & Endpoint Server (ISC)
Operational Improvements

- **Group Level Security**
  - Control access to Jobs based on Group Membership (including the JMC)
  - **JobLog SPI (System Pgming Interface) Controls**
    - Destination: Joblog or WAS Server log, or Both, or Neither (suppress)
    - Content: SPI can modify any job log line.

- **SMF Type 120 Subtype 9**

- **Memory Overload Protection**
  - Protects against over-scheduling jobs to an Endpoint, & Java OutOfMemory
  - Batch Container monitors job memory demand against available JVM heap
  - Automatic real time job memory estimation with declarative xJCL override
**The Job Management Console**

Browser-based view into the batch environment for Monitoring & Control:

**Browser**

A web interface allows very simple access.

**Command Line, Web Services, IIOP and JMX interfaces as well**

---

** Actions against select jobs**

<table>
<thead>
<tr>
<th>Select action</th>
<th>Apply</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select action</td>
<td></td>
</tr>
</tbody>
</table>

**ID that submitted the job**

<table>
<thead>
<tr>
<th>Submitter</th>
<th>Last Update</th>
<th>State</th>
<th>Node</th>
<th>Application Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>xadmin</td>
<td>2010-08-31 00:36:36.071</td>
<td>Ended</td>
<td>xdsnec</td>
<td>xdsr02c</td>
</tr>
<tr>
<td>xadmin</td>
<td>2010-08-31 00:36:01</td>
<td>Ended</td>
<td>xdsnec</td>
<td>xdsr02d</td>
</tr>
<tr>
<td>xadmin</td>
<td>2010-08-31 00:36:03</td>
<td>Ended</td>
<td>xdsnec</td>
<td>xdsr02d</td>
</tr>
<tr>
<td>xadmin</td>
<td>2010-08-31 00:36:34</td>
<td>Ended</td>
<td>xdsnec</td>
<td>xdsr02c</td>
</tr>
</tbody>
</table>

**Time stamps from the database**

**Job state**

**Node and server job dispatched to**

**Job log accessible under these links. Download button also available**

---

**Create schedule**

Specify the name of the schedule to:

- **Name:**
  - Daily at Noon

- **Start date (yyyy-MM-dd):**
  - 2011-06-16

- **Start time (HH:mm:ss):**
  - 12:00:10

- **Interval:**
  - Daily

---

**Compute Grid Job Management Console**

- Welcome
  - Job Management:
    - View jobs
    - Submit a job
  - Job Repository:
    - View saved jobs
    - Save a job
  - Schedule Management:
    - View schedules
    - Create a schedule

---

**Batch Update**

**August 11, 2011**
The Job Scheduler Interfaces

The previous chart tended to focus on the web interface, which is certainly the easiest to use. But others are present and offer great value:

- **Browser**
  - A web interface allows very simple access.

- **Command Line**
  - Automation through shell script programming.

- **Web Service**
  - Expose without requiring access to the JMC

- **RMI**
  - Expose to EJB clients

- **JMX**
  - Expose to Java JMX client

- **MDB** *(Compute Grid only)*
  - Used to integrate with enterprise schedulers

A wide variety of access methods

Blend to meet your business needs
Programming Model Enhancements in WCG V8

- **Persistent JobStepContext object**
  - Exists for life of job; Step-specific context reset at each job step.
  - New persistent user data object stored across checkpoint/restart.

- **Job and Step Listener**
  - Notification of Job/Step Start/End thru JobStepContext object.

- **Configurable Transaction Mode**
  - Select job step transaction mode: Local or Global

- **Record Processing Policy**
  - Skip bad records, Retry, or Stop Job Processing controls

- **Batch Data Stream Timeout** — Configurable by Job Step
  - Some BatchDataStreams need Short timeouts, others Long.

- **Record Metrics**
  - Skipped record count, Retry count,
  - Records/Second, Processing time
  - Written to Joblog
  - Available to batch application thru JobStepContext object.

- **OSGi Batch Applications**
  - Deploy batch applications as OSGi bundles

- **COBOL Container Support**
Traditional batch jobs or Schedulers can extend their reach to Compute Grid with the “WSGRID” utility running as a “Proxy” for an XD batch job:

- Proxy job stays active & receives joblog messages in SYSOUT file until XD job ends.
- WSGRID utility notifies the Scheduler of XD job Return Code.

```
//CGPROXY JOB 1,...
//SUBMIT EXEC PGM=WSGRID
//SYSPRINT DD SYSOUT=*
//WGCNTL: DD *
queue-manager-name=MQW1
scheduler-input-queue=WASIQ
scheduler-output-queue=WASOQ
//WGJOB DD PATH='/jcl/xJCL.xml'
//WGSUBS DD *
```

**Integrate with JES Schedulers**

 MDB interface Can use MQ or SIBus.
16:37:34:249 EDT] Job SimpleCIEar:00012 is queued for execution

********** BOTTOM OF DATA **********
How to Integrate with JES Batch Jobs

MDB interface to the dispatcher using MQ or imbedded messaging

Configuration simplified with Compute Grid V.8 ....

1. Define WebSphere MQ input & output queues.
2. Configure runtime variables to access MQ libraries.
3. `installWSGridMQ.py` script
   - Sets up JMS Connection Factories, Queues & ListenerPort
   - Installs WSGRID system application.
4. Create WSGRID load module in an executable library.
Failure Scenarios

What happens if . . .

• Proxy job fails (or canceled)
  • XD job is canceled.
• XD job fails (or canceled)
  • Proxy job fails
• Endpoint server fails
  • XD job fails
  • Proxy job will timeout
• Scheduler server fails
  • Proxy job fails
  • XD job fails
• MQ Fails
  • Proxy job fails,
  • XD job is canceled.

In all cases . . .
• XD job may be restarted.

WSGRID Return Codes

<table>
<thead>
<tr>
<th>Return code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Job ended normally</td>
</tr>
<tr>
<td>-1</td>
<td>Internal protocol error - WSGrid utility</td>
</tr>
<tr>
<td>-2</td>
<td>Input parameter error - WSGrid utility</td>
</tr>
<tr>
<td>-4</td>
<td>Job was suspended</td>
</tr>
<tr>
<td>-8</td>
<td>Job was canceled</td>
</tr>
<tr>
<td>-10</td>
<td>Job was forcibly canceled (z/OS® only)</td>
</tr>
<tr>
<td>-12</td>
<td>Job failed and is in restartable state</td>
</tr>
<tr>
<td>-14</td>
<td>Job failed and is in execution failed state**</td>
</tr>
<tr>
<td>-16</td>
<td>Catastrophic failure - WSGrid utility</td>
</tr>
</tbody>
</table>
Parallel Job Manager (PJM)

Batch processing often lends itself to running the work in parallel.
- WebSphere Compute Grid facilitates this with functions to cut up the work into “sub-jobs” and collect back the results:

Ah! A parallel job request!

Job Scheduler Function

Parallel Job Manager (PJM) Function

Batch Application exploiting WCG parallel job APIs

Working on:
- A - E
- F - J
- V - Z

Sophisticated parallel job coordination fabric
Configuring the Parallel Job Manager

How the PJM in V8 differs from previous versions

- Parallel job manager integrated into the batch container.
  - Not a separate system application as before.
  - No need to install and configure the PJM, or separate DB2 tables.
  - No shared library required - PJM APIs in batch utility JAR.
- The contents of the xd.spi.properties file are now part of the xJCL.
  - No `xd.spi.properties` file required.
- Only a single xJCL file is required.
  - Combines the top-level job xJCL with subordinate jobs.
- PJM applications built for CG V6 can run as is on WCGv8.
  - Migrate a WCGv6 PJM application to WCGv8:
    - Add the API implementation classes to the application EAR.
    - Reauthor xJCL as described in V8 InfoCenter.
“Job Class” Controls for Batch Jobs

- **Job Class** specified in xJCL:

```xml
<job name="SimpleCI" class="Compute"
```

- **Job Class can be used to limit:**
  - **MaxExecutionTime** – Before Jobs are canceled
  - **MaxConcurrentJob** – # Allowed to run concurrently
  - **MaxClassSpace** – Joblogs exceeding this size are Purged
  - **MaxFileAge** – Joblogs older than this are Purged
  - **MaxJob** – Max. # of Jobs (Oldest are Purged)
  - **MaxJobAge** – Jobs older than this are Purged from Output Queue

- **Job Class** can also be used to assign a Transaction Class (next foil....)
Classifying Batch Jobs with Compute Grid

Job Scheduler Function

WAS z/OS Controller Region

Batch Controller Bean

IBM Batch Container Classes and APIs

Your Batch Processing Application

Simple POJO model

WAS z/OS Servant

Function provided by WLM.

Function provided by WLM.

Actual WLM classification occurs in the controller region

CR queues the work to the servant

TCLASS assignment rules

TCLASS

Dispatch and Manage

Classify batch jobs according to your service goals.

→ Unique to z/OS platform.
**Classifying Batch Jobs:**

**ISC:Settings:** Job Scheduler > Classification Rules

- **Transaction Class** assigned based on:
  - Submitter Identity or Group
  - **Job name** or **Job class**
  - Application name or Application type
  - Platform or Time
  - Else, Default class = TCBATCH

**WLM Classification Rules:**
- CB Rules assign **Service Class** based on:
  - Generic Server (Cluster) name (CN)
  - **Transaction class** (TC)
  - assign by Job Scheduler > Classification Rules

If jobclass = 'Compute'
then classify to transaction_class TCOMP

If jobname LIKE 'XDCGIVT%'
then classify to transaction_class TCIVT

If no rules apply, then classify to transaction class TCBATCH

---

**Workload Manager CB Classification Rules:**

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Start</th>
<th>Service</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEF</td>
<td>CBCLASS</td>
<td>Z8*</td>
<td>CBFAST</td>
<td>Z8BAT</td>
</tr>
<tr>
<td>2</td>
<td>TC</td>
<td>TCOMP</td>
<td>CBSLOW</td>
<td>Z8BATCI</td>
</tr>
<tr>
<td>2</td>
<td>TC</td>
<td>TCIVT</td>
<td>CBCCLASS</td>
<td>Z8XDIVT</td>
</tr>
<tr>
<td>2</td>
<td>TC</td>
<td>TCBATCH</td>
<td>CBCCLASS</td>
<td>Z8BATCH</td>
</tr>
</tbody>
</table>
2 Classification Mechanisms for Batch Jobs:

(1) “Job Class” can be specified by a job. Job Scheduler dispatches jobs to Endpoint Servers, based on:

- Availability of Servers to accept new work.
- Number of jobs running within the maximum threshold for the “Job Class”
- “Maximum Execution Time” can also be assigned based on the Job Class.

Example:
Jobs in Job Class “Compute” managed by Job Scheduler
- Max. Execution Time
- Max. # Running at once.

Assigned Trans. Class

(2) z/OS Workload Manager (WLM) dispatches CPU cycles & I/O to jobs according to:

- **Service class goals** assigned to prioritize the:
  - “Importance” and
  - Response Time objectives or “Velocity” goals.
- Service Class assigned based on Cluster Name & **Transaction Class** assigned by the Scheduler.
- Dynamically adjusted based on:
  - Availability of CPU (& other) resources
  - Other work in the sysplex.

Example:

```xml
<job
  name="SimpleCI"
  class="Compute"
>
</job>
```

Classification Rules

Job name= Compute
TrClass= TCOMP

Batch job Servant
managed by WLM
in the CBSLOW
service class
Compute Grid Job Usage Recording

Job Usage Accounting with SMF records and DB2 JOBUSAGE tables:

SMF 120.9 UserData Section (WAS V.8 only)
- Job identifier
- Job submitter
- Accounting info

SMF 120.20 record contents:
- Job identifier
- Job submitter
- Final Job state
- Server
- Node
- Accounting information
- Job start time
- Last update time
- General CPU usage
- zAAP or zIIP CPU use

DB2 JOBUSAGE records:
- Job identifier
- Job submitter
- Final Job state
- Server
- Node
- Accounting info.
- Job start time
- Last update time
- General CPU usage

Record Metrics added to Joblogs:
- Skipped record count, Retry count,
- Records/Second, Processing time
- Written to Joblog & Available in JobStepContext object.

Generate reports & determine usage for Charge-back & Capacity Planning.

August 11, 2011
1. Batch Containers should run **everywhere**

2. **Portable Batch applications** across platforms and J2EE vendors

3. Location of the data dictates the placement of the batch application

4. Centrally managed by your enterprise scheduler

5. Integrating with existing: Disaster Recovery, Auditing, Logging, Archiving
Java Batch + CICS, or COBOL? Yes ...

New Batch container for CICS and COBOL interoperability function allows Java batch programs to call CICS and COBOL directly ...

Job declaration, or xJCL file

Job Scheduler, or Dispatcher

Batch Container, or Endpoint server

CICS TS 4.1 (with Java batch support pack installed)

IBM-written "glue" code that provides call interface between Java and native

Re-use CICS & COBOL assets with a Java batch job
Batch job in Compute Grid sends HTTP request to CICS to start the transaction program.

CICS matches it against the path in CN11URIresource to handle the request.
Integration with CICS “CN11” SupportPac

• SupportPac enables Compute Grid to schedule jobsteps in CICS
  • Provides Java interfaces to process input and output files in VSAM or DB2
    – Increased availability for CICS during batch processing, Automatic checkpoints, Restarts
    – CICS TS V. 4.1 required
  • Compute Grid provides
    – General job dispatching, management, execute control, monitoring
    – Higher throughput: Process jobs in parallel across multiple CICS regions
    – Locking of data: Updates are synchronised at Checkpoints for I/O resources
    – Failure/Recovery scenarios
      • If batch jobstep fails, Rollback updates, Restore last checkpoint and Retry jobstep.
  • Configuration:
    – Configure supporting CICS, DB2 and VSAM resources (Samplejobs provided)
    – Customize `endpoint-config.xml` & `CN11PROF` describing CICS to WCG
    – Initialize the CICS-to-Compute Grid Connection
      • Run the “CN11” CICS transaction to register the SupportPac samples with WCG.
    – `CN11SampleJCL.xml` sample job runs a CICS transaction that updates a VSAM file.
The Compute Grid COBOL Container

"COBOL Container" provides the JNI services:

Important Points:
- Create and destroy COBOL container multiple times in the servant address space
- COBOL container's LE enclave is separate from the address space's LE enclave (clean environment)
- JDBC T2 connection can be shared between Java and the COBOL program (maintaining transactional context using RRSAF)
Essentials of the New COBOL Support

Included as part of WebSphere Compute Grid V8.
- Compatible with WAS z/OS V.7 or V.8

The server must run in 31-bit mode since COBOL programs are 31-bit.
And run with a workload profile of ISOLATE (to insure OUTDD back from
COBOL works with DISPLAY)

Compiler and Link Edit Options:
• Must be a Dynamic Link Library (DLL)
• Must specify OUTDD(WCGILOUT) so output may go back to Java batch

Major pieces of this:
• Runtime support (a few JAR and native files) shipped with WCG V.8
• Development tooling support (JAR files)
• Call Stub Generator utility
• Your Java code, that calls the COBOL module.
Enabling the function...

Provide WAS server access to COBOL DLLs -- LIBPATH & STEPLIB

Create the container

```java
ILContainer ilc = ILContainerFactory.getFactory().create();
```

Name the COBOL module and procedure within the module, and where you pass parameters.

Create the procedure

```java
ILProcedure ilp = ILProcedureFactory.getFactory().create(...);
```

Call Stub Generator makes doing this much easier.

Invoke the procedure

```java
x = ilc.invokeProcedure(ilp)
```

Compile the COBOL modules and put DLL in USS or PDSE

Compiled DLL

In USS or in PDSE

```
IDENTIFICATION DIVISION.
PROGRAM-ID. ADDER RECURSIVE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
DATA DIVISION.
LINKAGE SECTION.
01 adder-input.
   05 int-a PIC 9(8).
   05 int-b PIC 9(8).
01 adder-return.
   05 int-c PIC 9(8).
```

The "procedure" is the value provided for PROGRAM-ID.

The DLL module name is also part of the invocation procedure.

August 11, 2011

Batch Update
The Call Stub Generator (CSG)

Java utility inspects the COBOL source & generates call stubs & data bindings:

COBOL Source

```
LINKAGE SECTION.
  01 adder-input.
    05 int-a                PIC 9(8).
    05 int-b                PIC 9(8).
  01 adder-return.
    05 int-c                PIC 9(8).
PROCEDURE DIVISION USING adder-input
  RETURNING adder-return.
```

CSG uses RAD function to create stubs/bindings in RAD

Data Bindings

Two ways to invoke the CSG:

Command line & ANT task in RAD:

Command Prompt

```
java -jar 
  lib/COBOLCallStubGenerator.jar 
  testcases/adder.cbl 
  -configFile csg.properties 
  -callStubPackage com.ibm.mycobol
```

IBM Rational Application Developer

ANT Task

Call Stub and Data Bindings

August 11, 2011
Documentation for COBOL Container

Call Stub Generator User's Guide
• Reference for file properties & details on setup and usage

COBOL Container Programming Guide
• Detail on programming to the call stubs and bindings

WP101909 Techdoc
“WebSphere Compute Grid COBOL Container”
• Technical Executive Color Flyer
• Architectural Comparison Document
• Contains the two documents shown above
• Available on www.ibm.com/support/techdocs
Migration from Compute Grid V 6.1.1 or FP

Notes:
- WCG V. 6.1 not supported on WAS V. 8.0.
- WCG V. 8.0 not supported on WAS V. 6.1.

Migrate the nodes in the following order:
A) Deployment manager.
B) Migrate the databases.
C) Schedulers and endpoints nodes one at a time.

Migrate the Deployment Manager:
1. Run the backup script `migrateWCGConfigTo8.py --backup`
2. Stop the deployment manager.
3. Unaugment Dmgr: `manageprofiles.sh -unagument`
4. Uninstall XD Compute Grid V. 6.1.1.3 or the Batch FeP
5. Install XD Compute Grid V. 8.0. (Installation Manager)
6. Augment the deployment manager. `manageprofiles.sh -agument`
7. Migrate the Database(s) `MigrateLRSCHEDTablesToV8.ddl`
8. Start the deployment manager.
9. Run the restore script `migrateWCGConfigTo8.py --restore`

Migrate the Scheduler and Endpoint nodes:
10. Stop the server and node.
11. Unaugment the profile `manageprofiles.sh -unagument`
12. Uninstall XD Compute Grid V. 6.1.1.3 or the Batch FeP
13. Install XD Compute Grid V. 8.0. (Installation Manager)
14. Augment the scheduler node `manageprofiles.sh -agument`
15. Run the restore script `migrateWCGConfigTo8.py --restore`
16. Start the server.
Information Center & other Resources


Download docs: http://www.ibm.com/support/docview.wss?uid=swg27021566&wv=1
Techdocs: PRS4644, PRS4467, WP101783 & WP101909
PRS4686 - WAS z/OS Version 8 Configuration Spreadsheets
WP101936 - Migrating to WebSphere XD Compute Grid v8 on z/OS

August 11, 2011
Batch Update
<table>
<thead>
<tr>
<th>Day</th>
<th>Time</th>
<th>#</th>
<th>Title</th>
<th>Speaker</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wednesday</td>
<td>3:00</td>
<td>9483</td>
<td>Using IBM's New Cross-Platform Installer on z/OS</td>
<td>Mierzejewski</td>
<td>Oceanic 5</td>
</tr>
<tr>
<td>Thursday</td>
<td>8:00</td>
<td>9482</td>
<td>WAS Version 8 – Overview</td>
<td>Follis</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>9:30</td>
<td>9486</td>
<td>WAS Version 8 – Batch Update</td>
<td>Hutchinson</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>11:00</td>
<td>9485</td>
<td>WAS Version 8 – New z/OS Exploitation/Differentiation Features</td>
<td>Follis</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>1:30</td>
<td>9484</td>
<td>WAS Version 8 – High Availability Enhancements</td>
<td>Follis</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>3:00</td>
<td>9488</td>
<td>WAS - Back to Basics Part 1</td>
<td>Loos</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Thursday</td>
<td>4:30</td>
<td>9489</td>
<td>WAS - Back to Basics Part 2</td>
<td>Stephen</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Friday</td>
<td>8:00</td>
<td>9490</td>
<td>WAS for z/OS - Level 2 Update</td>
<td>Stephen</td>
<td>Europe 2</td>
</tr>
<tr>
<td>Friday</td>
<td>9:30</td>
<td>9487</td>
<td>WAS for z/OS – PotPourri</td>
<td>Follis, Hutchinson, Loos, Stephen, etc.</td>
<td>Europe 2</td>
</tr>
</tbody>
</table>