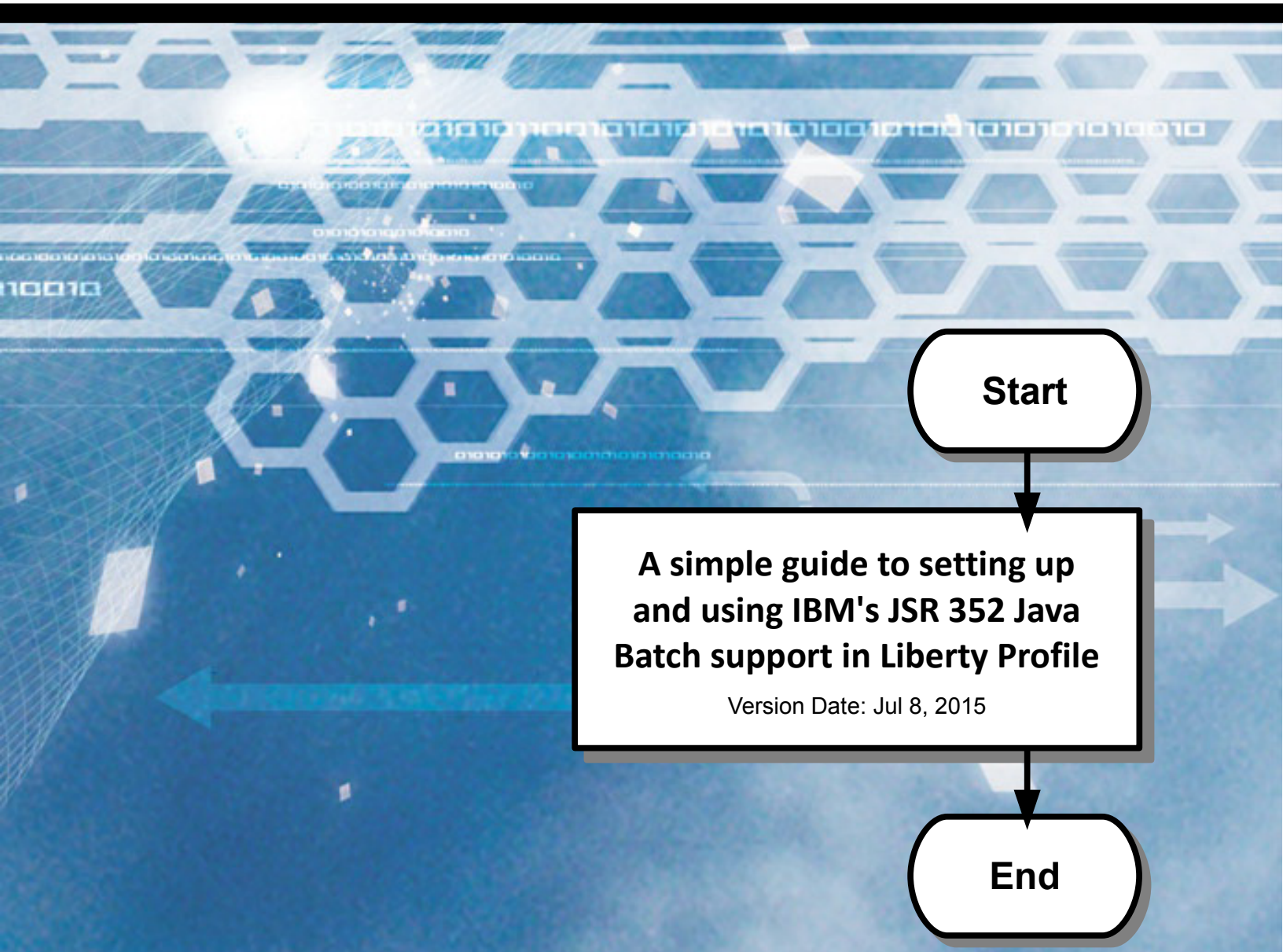
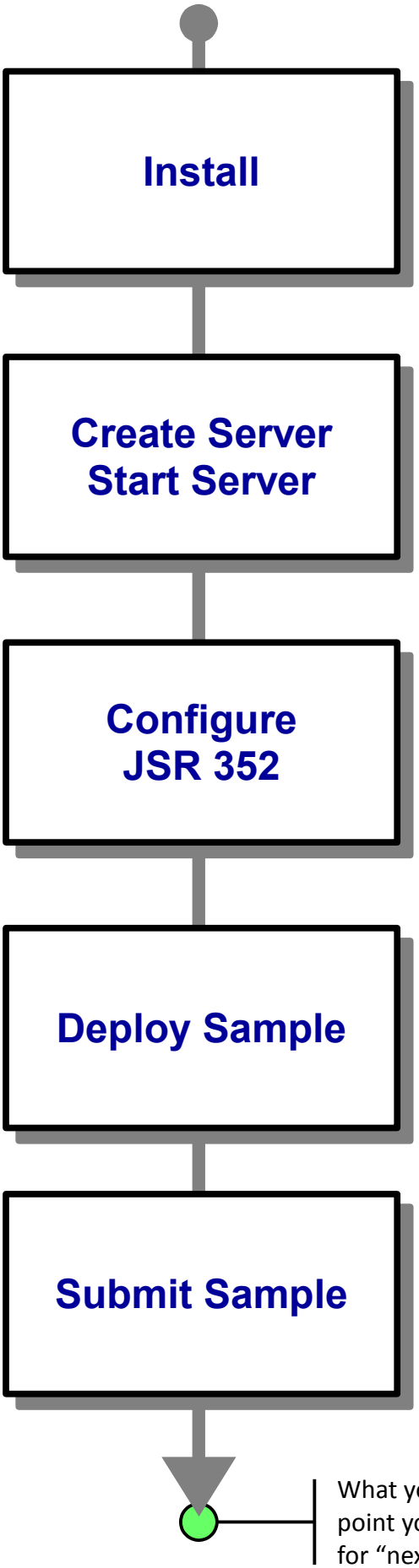
IBM JSR 352 Java Batch

Quick Start Guide





- Install Liberty Profile at the 8.5.5.6 level or higher
- Use *installUtility* to install batch feature
- More on page 3

- Use *server* to create a Liberty Profile server
- Make a few changes to *server.xml*
- Start server and make sure it works okay
- More on page 4

- Update *server.xml* with a few things
- More on page 5

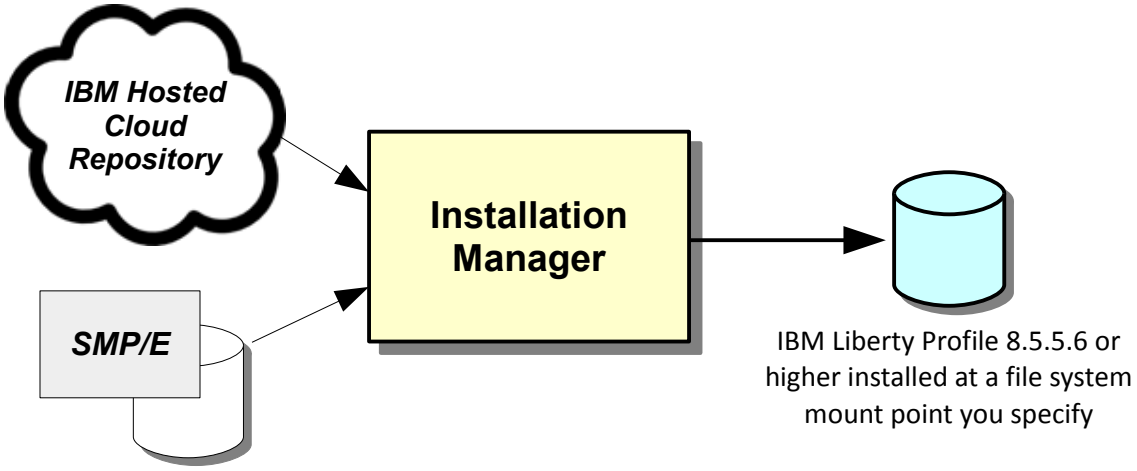
- Get sample from GitHub
- Drop sample into */dropins* directory
- More on page 6

- Point browser at server HTTP port
- Invoke sample with URI we provide here
- See results of batch job in *messages.log*
- More on page 7

What you achieved above was fairly basic. At this point you may wish to explore further. See page 8 for "next steps" in exploring IBM JSR 352

Installation

Step 1 Use IBM Installation Manager to install Liberty Profile* at the 8.5.5.6 level or higher



Step 2 Use the `installUtility` function of Liberty Profile to install the batch feature

The *installUtility* function adds features to the base install of Liberty Profile. It gets the feature from either the IBM web-based repository, or a local repository location you point to. Find the *installUtility* function in the */bin* directory of your Liberty Profile install path.

If you have Internet access, then:

```
installUtility install batch-1.0 --acceptLicense
```

*If you do **not** have Internet access, then see the WP102544 "Step-by-Step" guide for instructions on how to do this. The section is found under "Miscellaneous" with title "If your z/OS LPAR does not have internet access."*

This involves downloading a ZIP from IBM Fix Central that contains all the features. This is then uploaded to your z/OS system and unzipped. The *installUtility* is then pointed to the local repository.

Validate the feature is there by listing the installed features:

```
productInfo featureInfo
```

Review the returned listing of features and insure `batch-1.0` is included.

With Liberty Profile installed and the *batch-1.0* feature included, you are ready to move on to the next step … creating the server.

* The JSR 352 function is provided with Liberty Profile on all supported platforms, including z/OS. This Quick Start is assuming z/OS. The instructions are generally applicable to other platforms as well.

Server Creation and Start

Step 1

Use the *server* function of Liberty Profile to create the server

This is a common procedure for creating a Liberty Profile server on a UNIX-based environment, which includes z/OS:

Open a Telnet (or OMVS) session to your system and change to the Liberty Profile /bin directory:

```
cd /<install_path>/bin
```

Export an environment variable for the location of a 64-bit Java SDK:

```
export JAVA_HOME=/<path_to_SDK>
```

Export an environment variable for the location of your Liberty Profile server:

```
export WLP_USER_DIR=/<path_to_user_director>
```

Only needed if the profile you are creating will reside *outside* the installation directory, as is often the case on z/OS.

Create the server:

```
server create <server_name>
```

Step 2

Perform minimal updates to *server.xml* to make server accessible

As a security precaution, the default *server.xml* will not have a *host=* attribute provided, which restricts usage to *localhost*. For z/OS, where a local browser isn't available, you need to add a virtual host to make the server accessible from a remote browser.

Edit server.xml and update with one line:

```
<httpEndpoint id="defaultHttpEndpoint"
              host="*"
              httpPort="9080"
              httpsPort="9443" />
```

Save the file

Step 3

Start the server and verify access

From Telnet (or OMVS) session:

```
server start <server_name>
```

In the *messages.log* file you should see (among other messages):
*CWWKO0219I: TCP Channel defaultHttpEndpoint has been started and is now listening for requests on host * (IPv4) port 9080.*
CWWKF0011I: The server server1 is ready to run a smarter planet.

From browser:

```
http://<host>:9080/
```

You should receive the "Welcome to Liberty" page from the started server.

With the Liberty Profile server created and started, you are ready to move to the next step … configuring the *batch-1.0* feature

JSR 352 Configuration

Step 1 Edit the *server.xml* file:

Edit server.xml and do the following:*

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">

    <!-- Enable features -->
    <featureManager>
        <feature>jsp-2.2</feature>
        <feature>servlet-3.1</feature>
        <feature>batch-1.0</feature>
    </featureManager>

    <httpEndpoint id="defaultHttpEndpoint"
                  host="*"
                  httpPort="9080"
                  httpsPort="9443" />

</server>
```

Remove *jsp-2.2*

Add *servlet-3.1*
Add *batch-1.0*

Save the file

Step 2 Look for messages in the *messages.log* file:

Browse the messages.log file and look for the following:

```
CWWKF0012I: The server installed the following features:
                              [jdbc-4.1, servlet-3.1, jndi-1.0, batch-1.0].
```

For this Quick Start we are illustrating the most basic JSR 352 operations, which requires just *batch-1.0*.

The IBM JSR 352 implementation includes quite a few more sophisticated things, such as a REST interface, a command line tool (batchManager), job logging, and multiple JVM support. Those require the *batchManagement-1.0* feature as well as additional configuration. See page 8 for guidance on where to go to learn more about those functions.

With the *batch-1.0* feature enabled, you are ready to deploy and submit the sample JSR 352 application.

* If on a non-z/OS platform, these instructions assume you downloaded and installed "Web Profile." The "kernel" and "javaee7" downloads would appear slightly different from what's shown above.

Deploy Sample JSR 352 Application

Step 1 Get the "SleepyBatchlet" sample* application from GitHub:

Go to the following GitHub URL:

```
https://github.com/WASdev/sample.batch.sleepybatchlet
```

Download the following file:

SleepyBatchletSample-1.0.war

Right mouse-click and "Save link as," or click on the link on the "Raw" button to download.

Step 2 Upload to the server

Upload the WAR file to the server in binary and place in the /dropins directory:

Make sure file permissions allow at least READ by the Liberty Profile server ID

Look for the following messages in the messages.log file:

```
CWWKZ0018I: Starting application SleepyBatchletSample-1.0.
SRVE0169I: Loading Web Module: SleepyBatchletSample-1.0.
SRVE0250I: Web Module SleepyBatchletSample-1.0 has been bound to default_host.
CWWKT0016I: Web application available (default_host):
                                    http://<host>:9080/SleepyBatchletSample-1.0/
CWWKZ0001I: Application SleepyBatchletSample-1.0 started in 0.347 seconds.
```

With the sample application deployed and started, you are ready to submit the application and see it run ...

* The JSR 352 specification defines two job step types: "batchlet" and "chunk." A *batchlet* step type is invoked and performs whatever is coded in the Java class. A *chunk* step is the more traditional iterative batch loop model. The SleepyBatchlet sample is a *batchlet* step type.

Submit Sample and See Results

Step 1 Start the job from a browser

From a browser, issue the following URL:

```
http://<host>:<port>/SleepyBatchletSample-1.0/sleepybatchlet?action=start
```

Look for the following on the browser:

```
Job started!

JobInstance: instanceId=0, jobName=sleepy-batchlet

JobExecution: executionId=0, jobName=sleepy-batchlet, batchStatus=STARTING,
createTime=Fri May 22 14:40:49 EDT 2015, startTime=null, endTime=null,
lastUpdatedTime=Fri May 22 14:40:49 EDT 2015, jobParameters={}
```

Step 2 See further evidence of the batch job running

Look in the messages.log file for the following messages (see notes that follow):

```
CWWKY0005I: The batch In-Memory persistence service is activated.
CWWKY0008I: The batch feature is using persistence type In-Memory.   (1)
SleepyBatchlet: process: entry
SleepyBatchlet: process: sleep for: 15   (2)
SleepyBatchlet: process: [0] sleeping for a second...
SleepyBatchlet: process: [1] sleeping for a second...
SleepyBatchlet: process: [2] sleeping for a second...
SleepyBatchlet: process: [3] sleeping for a second...
SleepyBatchlet: process: [4] sleeping for a second...
SleepyBatchlet: process: [5] sleeping for a second...
SleepyBatchlet: process: [6] sleeping for a second...
SleepyBatchlet: process: [7] sleeping for a second...   (3)
SleepyBatchlet: process: [8] sleeping for a second...
SleepyBatchlet: process: [9] sleeping for a second...
SleepyBatchlet: process: [10] sleeping for a second...
SleepyBatchlet: process: [11] sleeping for a second...
SleepyBatchlet: process: [12] sleeping for a second...
SleepyBatchlet: process: [13] sleeping for a second...
SleepyBatchlet: process: [14] sleeping for a second...
SleepyBatchlet: process: exit. exitStatus:
SleepyBatchlet:i=15;stopRequested=false   (4)
```

Notes:

1. The JSR 352 specification calls for a JobRepository to store information about batch jobs. In the absence of any configured database (such as Derby, or DB2), the IBM JSR 352 implementation will default to an in-memory JobRepository.
2. By default the sample batchlet will sleep for 15 seconds.
3. Once every second the sample batchlet writes a println() message indicating its status
4. The sample batchlet ended on its own, without being stopped by a stop request.

Congratulations! You have set up and used the IBM JSR 352 implementation inside Liberty Profile. You successfully deployed and submitted a "batchlet" job step application.

Next Steps in Exploring IBM JSR 352

Step 1 See the WP102544 Techdoc Page

From a browser, issue the following URL:

`http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102544`

Review the "IBM JSR 352 Java Batch Overview" presentation

This will give you a good overview of the JSR 352 specification and extensions to the specification IBM has made in their implementation to make Java batch more usable and manageable.

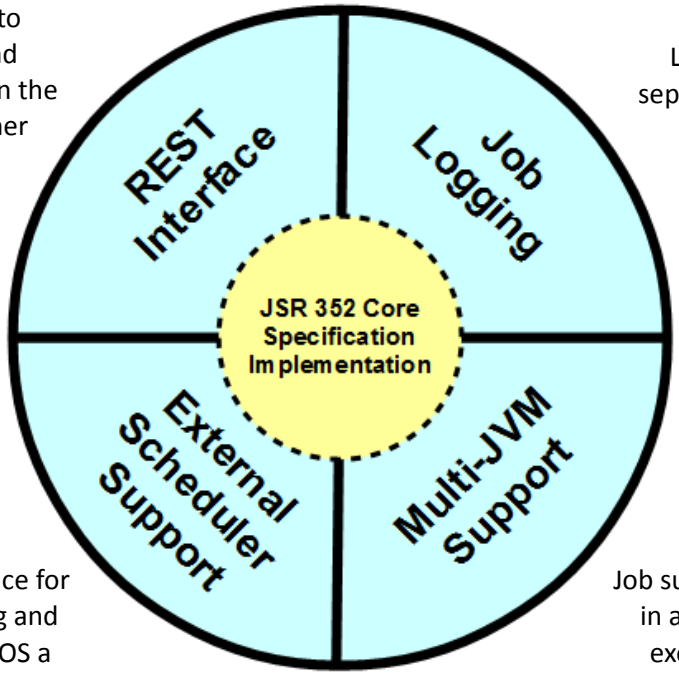
Review the "Step-by-Step Implementation Guide" document

This is a detailed guide for configuring and using many of the additional features, such as Derby or DB2 as a JobRepository, the REST interface, the batchManager command line utility, the batchManagerZos native command line utility for z/OS, and the multiple JVM support.

JSR 352 and IBM extensions to the standard:

A RESTful interface to the JSR 352 JobOperator, this provides functions to submit, monitor and manage jobs running in the IBM JSR 352 container

Logging of job activity separate from the runtime.



A command-line interface for submitting, monitoring and managing jobs. On z/OS a native implementation is provided that uses the cross-memory WOLA function

Job submission and dispatching in a JVM separate from job execution in other JVMs. Provides greater flexibility for JSR 352 runtime topology design and operations.

End of Document