Advanced
Technology
Group

IBM

CICS Support for
IBM WebSphere Application Server Liberty

Steve Fowlkes – fowlkes@us.ibm.com

Leigh Compton – lcompton@us.ibm.com

Topic Abstract

- There is never been a better time to take advantage of Java in CICS
- This topic is intended to cover
  - CICS Support for the Liberty profile in CICS TS
- Debugging and Tuning covered in a different topic

Notes

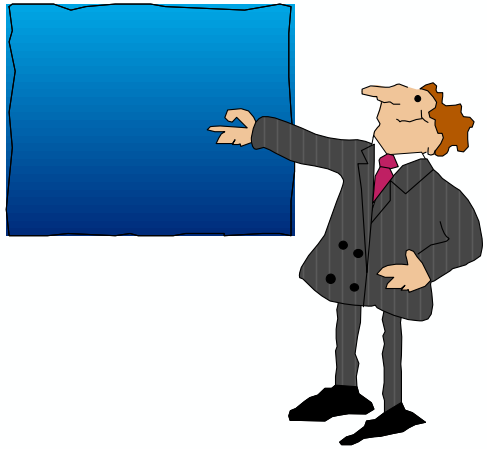
CICS TS has evolved to become the world's most powerful mixed language application server. Applications can share core programming contexts regardless of the language its components are written in. This session will discuss how developers can create incredible mixed language applications, that include Java EE and Node.js capabilities.

Advanced Technology Group

IBM

Agenda

- CICS Support for the Liberty Profile
  - Application scenarios
  - Application deployment
  - Core technologies
  - Up & running



Advanced
Technology
Group

IBM

Java / Jakarta in CICS

What is Websphere Liberty?

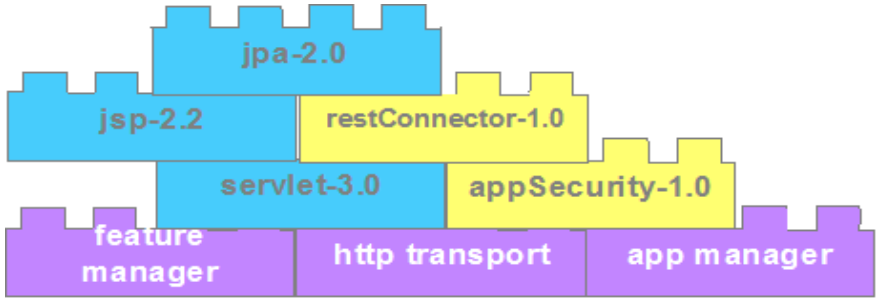
A lightweight, dynamic, composable JEE server runtime

- Lightweight
  - Server install is only about 55 MB
  - Extremely fast server starts – typically under 5 seconds
- Dynamic
  - Available features are user selected and can change at runtime
  - Restarts are not required for server configuration changes
- Composable
  - Features are implemented as loosely coupled components with lazily resolved optional and mandatory dependencies
  - The availability of features and components determines what Liberty *can do and what is available to applications*
  - Focuses on having an easily configurable opt-in customization model, giving you full control over your configuration

Composability – Based on features

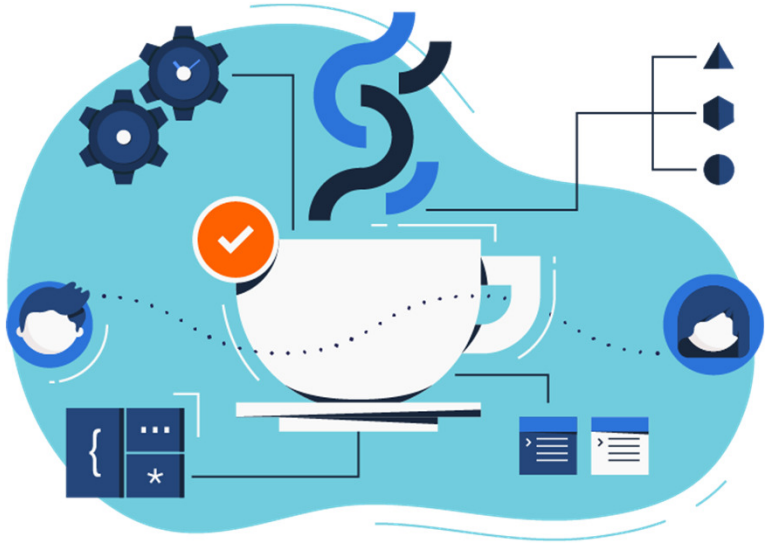
Aside from its kernel, everything in Liberty is delivered as composable features

```
<server description="composabilityIsKey">

    <featureManager>
        <feature>appSecurity-1.0</feature>
        <feature>jsp-2.2</feature>
        <feature>restConnector-1.0</feature>
        <feature>jpa-2.0</feature>
    </featureManager>

</server>
```



Java EE 8 Full Platform application support

- CICS TS supports Java applications that are written to the Java Enterprise Edition 8 (Java EE 8) Full Platform specification
  - Using the embedded version of IBM WebSphere Liberty
- Java applications that are hosted in CICS TS are integrated with CICS tasks by default
- A simple and powerful mechanism of modernizing CICS applications by using Java EE 8 features and capabilities
- IBM 64-bit SDK for z/OS, Java Technology Edition, Version 8 – latest fix pack recommended SR7 FP6



Support for Jakarta EE 9.1

- The CICS Liberty JVM server now supports the Jakarta Enterprise Edition (EE) 9.1

- The Jakarta technologies and specifications are an evolution of Java EE 8
  - Allows developers and applications to easily transition from Java EE to Jakarta EE

- The promise of Jakarta EE is a community-driven open source model
  - More frequent releases than Java EE
  - Evolving more quickly to address the needs of modern applications

Jakarta EE 9.1

- Most obvious change is the Namespace change from javax to jakarta
  - **import** javax.servlet.http.HttpServlet;
  - **import** jakarta.servlet.http.HttpServlet;
- For convenience, a transformation tool is offered that operates on an application and produces a 'jakarta' version (jar)
  - https://projects.eclipse.org/projects/technology.transformer
  - Used internally in build process
- Be aware: there are some feature ***renames*** - not just feature version increments
  - Historically, each new release of EE platform corresponded with a **version update** to each of the Liberty Features
    - E.g. EE7 -> EE8: jaxrs-2.0 -> jaxrs-2.1
  - However for Jakarta 9, features have been both **renamed and incremented major version number** (short names are trademarked)
    - E.g. EE8 -> EE9: jaxrs-2.1 -> restfulWS-3.0
    - E.g. EE8 -> EE9: jsp-2.3 -> pages-3.0
  - Full details of Jakarta feature updates can be found here
    - https://openliberty.io/docs/latest/jakarta-ee9-feature-updates.html

Jakarta EE 9.1

- CICS has reworked the internal 'cicsts:' Liberty features to auto-provision and match the Liberty server level of Java EE / Jakarta EE
- By product of this is the deprecation of the *com.ibm.cics.jvmserver.wlp.wab jvm profile option*
  - *-Dcom.ibm.cics.jvmserver.wlp.wab=true|false*
- wab-1.0 is a Java EE 6/7 feature
- The option was added to remove it from server.xml (to allow Java EE 8 / Jakarta EE 8 toleration)
  - This option is no longer required
- Coding this value will now have no effect, a deprecation message is emitted by CICS on start-up if the value is present in the JVM profile

Support for Java 11

Java 11

- IBM Semeru Runtime Certified Edition for z/OS, Version 11 fix pack 11.0.15.0 minimum

Java 11 - restrictions and changes:

- Liberty 'safKeyringjce' replaces 'safKeyring' to utilise IBM Semeru SAF security providers
- IBM Semeru, does not provide JAXB or JAF in the base runtime anymore, those have moved to Jakarta EE
- CICS offers two new JVM profile options to add/remove JAXB and JAF function into the CICS runtime
    - JAXB_REGISTRATION= {TRUE | FALSE}
    - JAF_REGISTRATION= {TRUE | FALSE}
- The Liberty JDBC 4.3 feature requires Java 11 (APAR coming to support it post GA)
- CICS Explorer support for Java 11 initially only through Eclipse Marketplace

Java support in CICS (includes Jakarta)

- CICS provides the tools and runtime to develop and run Java applications in a CICS JVM
  - Java applications can interact with CICS services and applications written in other languages
  - You can develop applications using the IBM CICS SDK for Java, Maven modules, or Gradle modules

- The CICS JVM server
  - Eligible Java workloads can run on specialty engine processors
    - reducing the cost of transactions
  - Different types of work such as threadsafe Java programs and web services
  - Application life cycle can be managed in the OSGi framework
    - no need to cycle the JVM server
  - Java applications that are packaged using OSGi can be ported more easily between CICS and other platforms
  - Java EE applications can be deployed into the Liberty JVM server

IBM CICS SDK for Java – The CICS Explorer

- The IBM CICS SDK for Java is included with the CICS Explorer
  - Provides support for developing and deploying applications that comply with the OSGi Service Platform specification

- The IBM CICS SDK for Java EE, Jakarta EE and Liberty is included as an option with the CICS Explorer
  - Supports packaging of Liberty applications into CICS bundles that can be deployed to CICS

- The OSGi Service Platform provides a mechanism for developing applications using a component model
  - Deploy applications into a framework as OSGi bundles
    - an OSGi bundle is the unit of deployment for an application component
      - contains version control information, dependencies, application code

- The IBM CICS SDK for Java allows development of Java applications for any supported release of CICS
  - The SDK includes the Java CICS library (JCICS) to access CICS services
  - Examples to get started with developing applications for CICS

Using Maven or Gradle

- You can use popular build tools such as Maven and Gradle to create your own scripts for building and deploying CICS Java programs
  - An alternative to the IBM CICS SDK for Java
  - Easy management of dependencies
    - Java developers can add the required versions of the Java CICS APIs and the CICS annotation processor to the Java dependencies
  - More freedom when choosing the development environment
    - Maven and Gradle support most Java IDEs
      - such as Eclipse, IntelliJ IDEA, and Visual Studio Code
  - Better integration into a build toolchain
    - Maven and Gradle integrate smoothly with other automation tools such as Jenkins and Travis CI

- The following artifacts are available on Maven Central
  - The Java CICS class library (JCICS)
    - Provides the EXEC CICS API support for Java applications in CICS TS
  - The CICS annotations library and the CICS annotation processor
    - Provides support that enables CICS programs to invoke Java applications in a Liberty JVM server
  - A bill of materials (BOM)
    - Defines the versions of the other artifacts to ensure that they are at the same CICS TS level

Plug-ins for Gradle and Maven to automate building CICS bundles

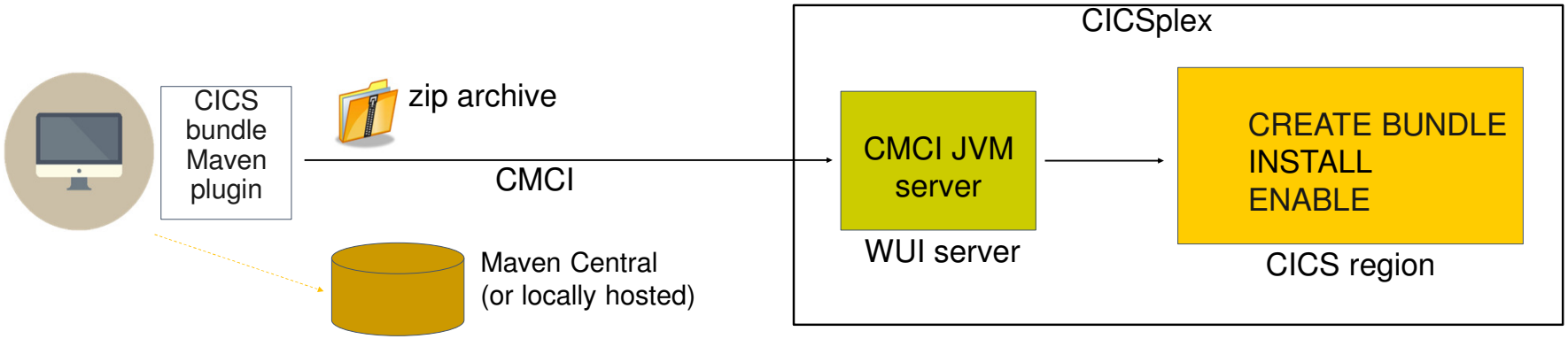
- Use Gradle or Maven to build CICS bundles that provide a convenient packaging mechanism for Java applications, and a wide range of CICS resources
- Push and lifecycle bundles containing Java (and other artifacts) as part of the build process

- Maven plug-in `cics-bundle-maven-plugin`
- Gradle plug-in `com.ibm.cics.bundle`

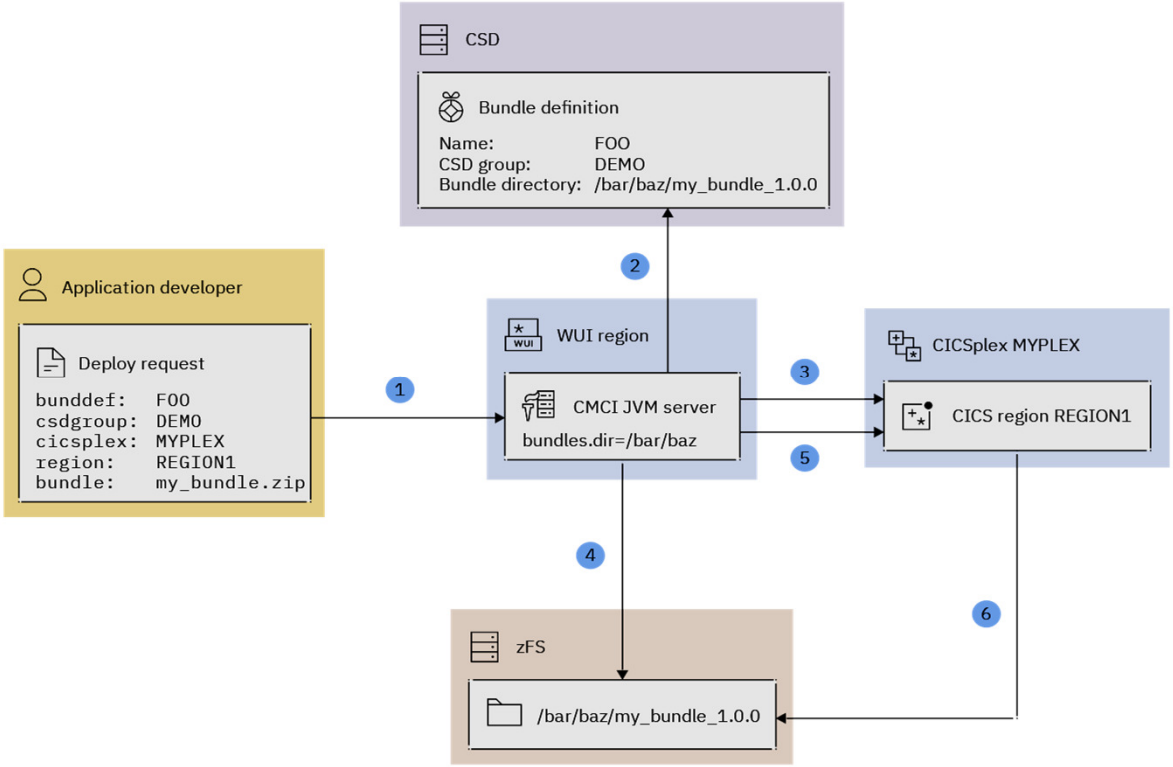
- Build CICS bundles as part of Gradle or Maven build tools, ready to be installed into CICS TS
- Can *push* and *lifecycle* bundles as part of the build process using the new CICS bundle deployment API
- Open source projects

Plug-ins for Gradle and Maven to automate building CICS bundles

- *Enables developers to deploy and lifecycle CICS bundles as part of their build process*
- Significantly reduces the time to rebuild and deploy the application
- Provided by the CICS management client interface (CMCI) JVM server in a CICSPlex SM environment
  - Single Region in CICS TS 6.1
- Can use with the Gradle and Maven plug-ins to deploy & lifecycle CICS bundles
- Deployment API receives metadata and the compressed CICS bundle
  - including details of the CICSplex, the CICS region, and a CICS bundle definition
- CICS bundle is installed and enabled into the specified CICS region
  - with any existing CICS bundle disabled and discarded where required



How the API works



1 The application developer publishes the application bundle via the CICS bundle deployment API

Validate
2 The CMCI JVM server finds the BUNDLE definition in the target region's CSD and checks that the BUNDLE definition's bundle directory (BUNDLEDIR) attribute value is within the API's configured bundles directory

Uninstall
3 The CMCI JVM server checks whether any previously installed bundle with the same name as the BUNDLE definition specified exists in the target region. Such a bundle is disabled and discarded as required

4 The CMCI JVM server deletes any previous bundle with the same name and version from the bundles directory on zFS. Then, it unpacks the published bundle to the bundles directory

Install
5 The CMCI JVM server initiates a CSD install of the BUNDLE definition

6 The target region reads the bundle from zFS and installs it

New Java API classes – JCICSX

- Improved API that is easier to use, supporting:
- **Linking** with a channel
- Putting and getting from **containers**
- Getting some **task information** such as task number
- **Syncpoint** and rollback

- Less boilerplate
- More easily understandable code
- Fewer mistakes
- Mockable and stubbable

- Examples:

Easier to link and use channels and containers:

```
return task.createProgramLinkerWithChannel(PROG_NAME,CHANNEL)
       .setSyncOnReturn(false)
       .setStringInput(INPUT_CONTAINER,INPUTSTRING)
       .link()
       .getCHARContainer(OUTPUT_CONTAINER).get();
```

Easier to use Mocking with frameworks like Mockito:

```
task = Mockito.mock(CICSContext.class);
accountsChannel = Mockito.mock(Channel.class);
Mockito.when(task.getChannel("ACCOUNTS"))
       .thenReturn(accountsChannel)
accountContainer = Mockito.mock(BITContainer.class);
Mockito.when(accountsChannel.getBITContainer("CURRENT"))
       .thenReturn(accountContainer);
```

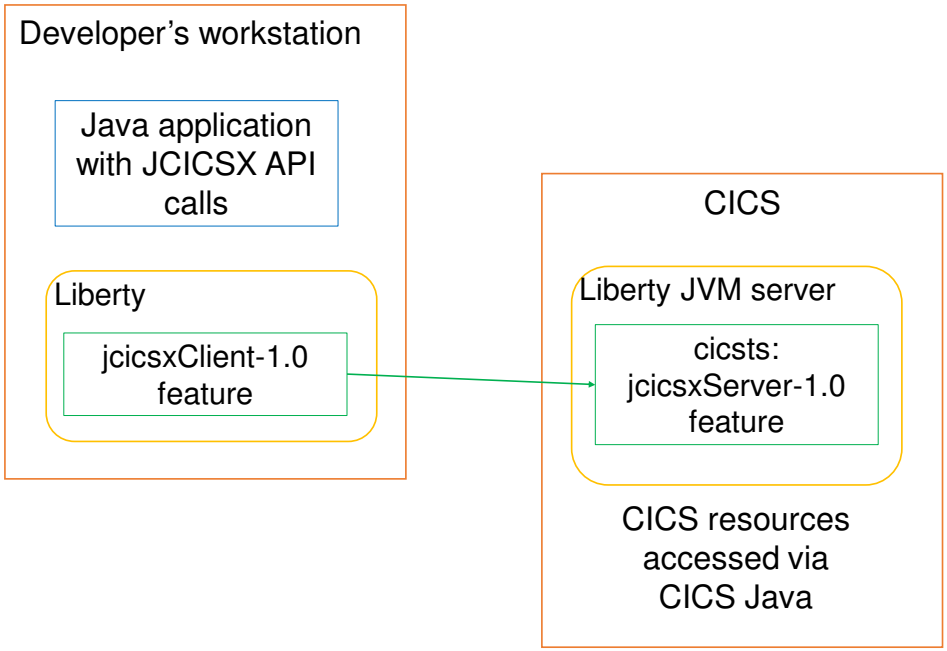
New remoteable Java API classes – JCICSX

- The new API classes enable **remote development**
- Developers can run CICS Java application code on **their own workstations**
  - making calls to CICS using the new API
- Calls executed on a **real CICS region**
  - then application code will continue on the developer's workstation

➢ Faster development cycles

➢ Use of technologies such as line-by-line **debugging** and **hot code replace** when developing CICS applications

➢ Access to CICS APIs during development via the JCICS development feature for Java (jcicsxClient-1.0)



Bundle status & config file polling

CICS bundle status wired to Liberty application status

- CICS bundle with Web application bundle part remains in `ENABLING` state until applications are started in Liberty
- (Also available in CICS TS V5.5 APAR PH08321)

- Enables:
  - More robust application deployments
  - System policy rules for bundle status to be used for automation
  - Liberty Admin Center for recycling apps
  - MBean config file updates

    ```
    <config updateTrigger="mbean"/>
    ```

    ➢ *Reduces need for continual polling of config files - server.xml & installedApps.xml*

Liberty Product Extensions – V5.6

What is a Liberty Product Extension?

- A collection of one or more user-features designed to extend the Liberty application server
- Typically placed into the Liberty install directories for use by all derived servers
- In CICS this does not work well because the WLP_INSTALL_DIR location is not writeable

How

- Develop and deploy your Product Extension to a specific zFS directory
- Install to Liberty via `LIBERTY_PRODUCT_EXTENSIONS` option

```
LIBERTY_PRODUCT_EXTENSIONS=MyExtension;/u/dir1
```

Liberty SPI Commands – V5.6

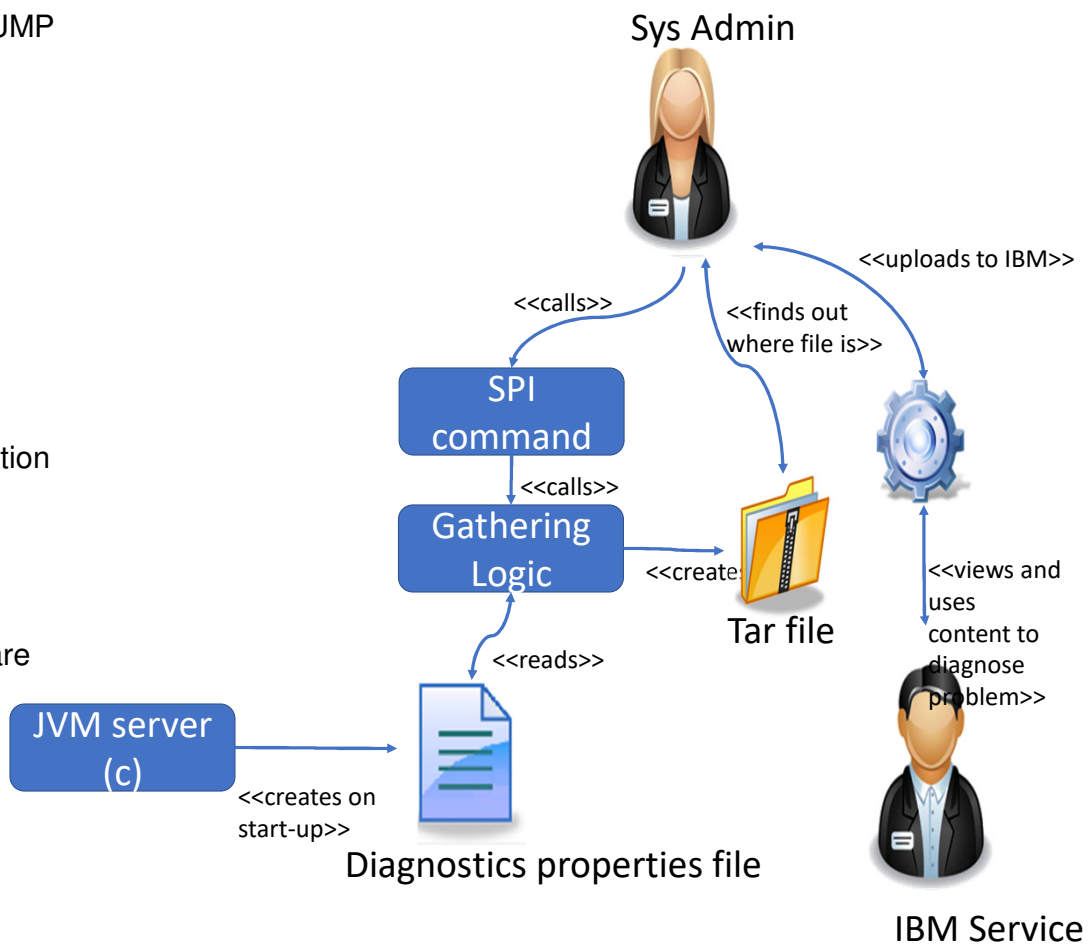
- Liberty HTTP endpoints can be resumed/paused using
  - SET JVMENDPOINT command
    - allows ports to be enabled or disabled
    - Allows Web applications to be taken off-line without terminating JVM
    - ServerEndpointControl MBean

- INQUIRE JVMENDPOINT command
    - Returns details of all HTTP and JMS MDB ports used in Liberty JVM servers

- INQUIRE JVMSERVER
    - Returns JVM profile, stdout/stderr/jvmlog/jvmtrace, WORK_DIR, and JAVAHOME

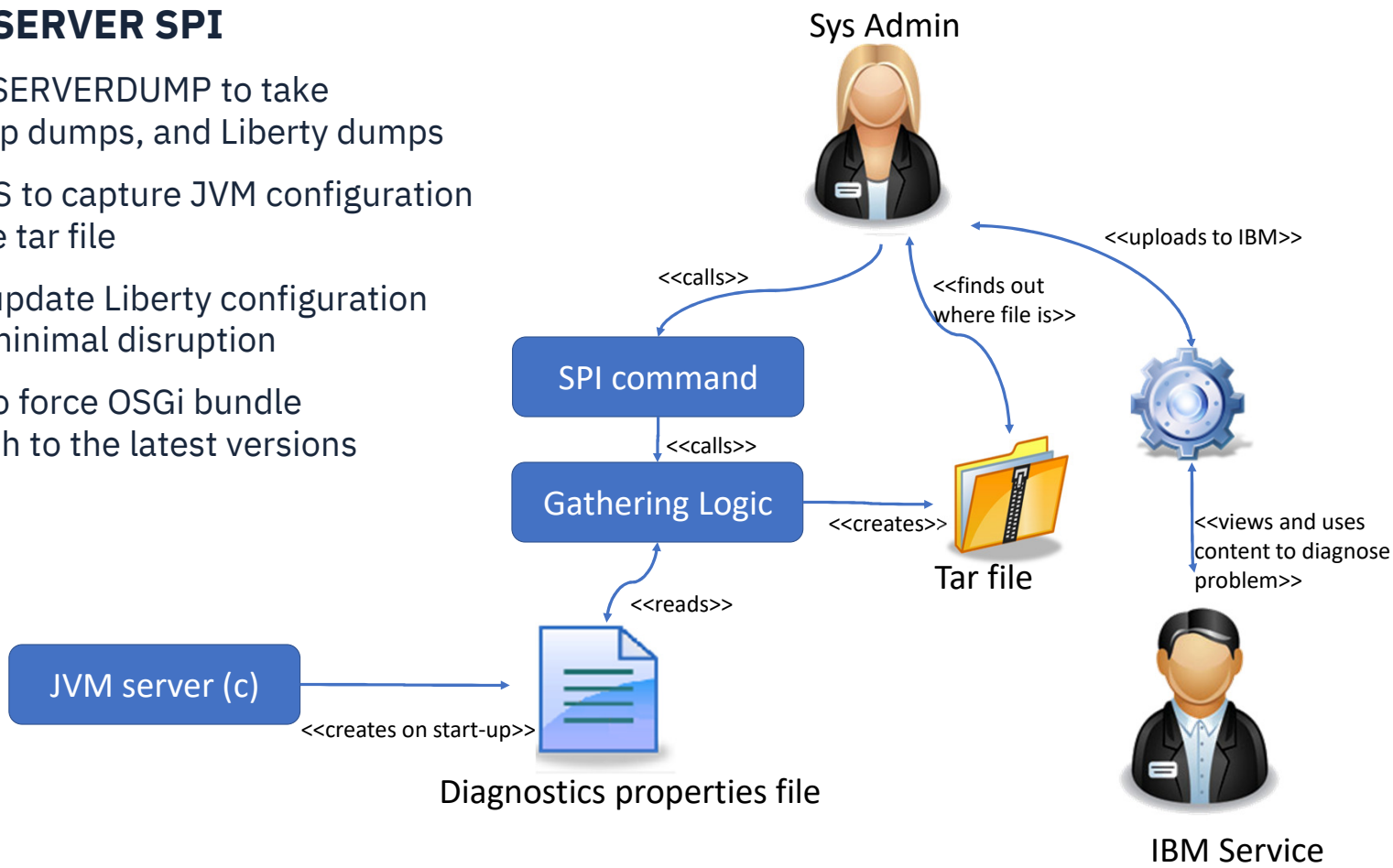
Enhanced administration commands for JVM server

- PERFORM JVMSERVER JVM DUMP / LIBERTY SERVERDUMP
  - takes Java and Liberty dumps
- PERFORM JVMSERVER GATHER DIAGNOSTICS
  - captures configuration, logs, traces, dumps, and output files
  - combined into a single archive
- PERFORM JVMSERVER LIBERTY REFRESH APPLICATION/CONFIGURATION
  - refreshes a Liberty application or the Liberty server configuration
- PERFORM JVMSERVER OSGI REFRESHPKGS
  - provides a mechanism for refreshing bundle dependencies
  - ensures latest version of packages and dependent libraries are used
  - can be disruptive and stall workloads



CICS JVMSERVER SPI enhancements

- New **PERFORM JVMSERVER SPI**
- JVM DUMP / LIBERTY SERVERDUMP to take javacore, heap and snap dumps, and Liberty dumps
- GATHER DIAGNOSTICS to capture JVM configuration and output into a single tar file
- LIBERTY REFRESH to update Liberty configuration and applications with minimal disruption
- OSGI REFRESHPKGS to force OSGi bundle dependencies to refresh to the latest versions



CICS JVM profiles – includes

Include & share common configuration when cloning JVM servers across CICS regions

- For example unique ports, database configuration or log settings

```
%INCLUDE=<file_path>
```

Append to variables are built up over multiple lines

```
OSGI_BUNDLES=&CLONEDIR;/mybundle.jar
+OSGI_BUNDLES=/newpath/mybundle2.jar
```

... is equivalent to ...

```
OSGI_BUNDLES=&USSHOME;/&JVMSERVER;/bundles/mybundle.jar;/newpath/mybundle2.jar
```

Including server.xml snippets

- Inject Liberty configuration into server.xml

- A new JVM profile option LIBERTY_INCLUDE_XML is provided
  - to enable Liberty to load shared configuration
  - making it easier to administer, clone, and control Liberty JVM servers

- You can now use the LIBERTY_INCLUDE_XML property in JVM profiles
  - specify files that CICS will add <include> tags for

- In JVM profile
  `LIBERTY_INCLUDE_XML=<file>`

Management – JVM server log

Extended CICS JVM server message

```
LOG_LEVEL=INFO | WARNING | ERROR | NONE
```

- New dfhjvmlog zFS file for CICS JVM server information, warnings, and errors
- Can be redirected to MVS JES DD
- For example
  - a value of NONE suppresses all output
  - a value of WARNING gives log entries of warning level and above
  - the default value is INFO

More than one Liberty JVM server per CICS region

Multiple secure Liberty servers in a CICS region

- Provides improved application isolation or scalability without increasing number of regions
- Each Liberty server can have its own configuration and lifecycle – ideal for developers

Wait for Liberty angel process * (also in V5.4 APAR PI92676)

`-Dcom.ibm.ws.zos.core.angelRequired=true`

- More robust CICS start-up and IPL procedures
- Ensure that a Liberty JVM server will connect to a Liberty angel process
  - before reaching the ENABLED state
- Integrates with named Liberty angel process `-Dcom.ibm.ws.zos.core.angelName`

JSON Web Token

Liberty JWT feature

- Programmatically parse, build, and verify JWT tokens in Java applications
- Provides for authentication using digitally signed web tokens
- Also available on CICS TS V5.3 and 5.4 with APAR PI91554

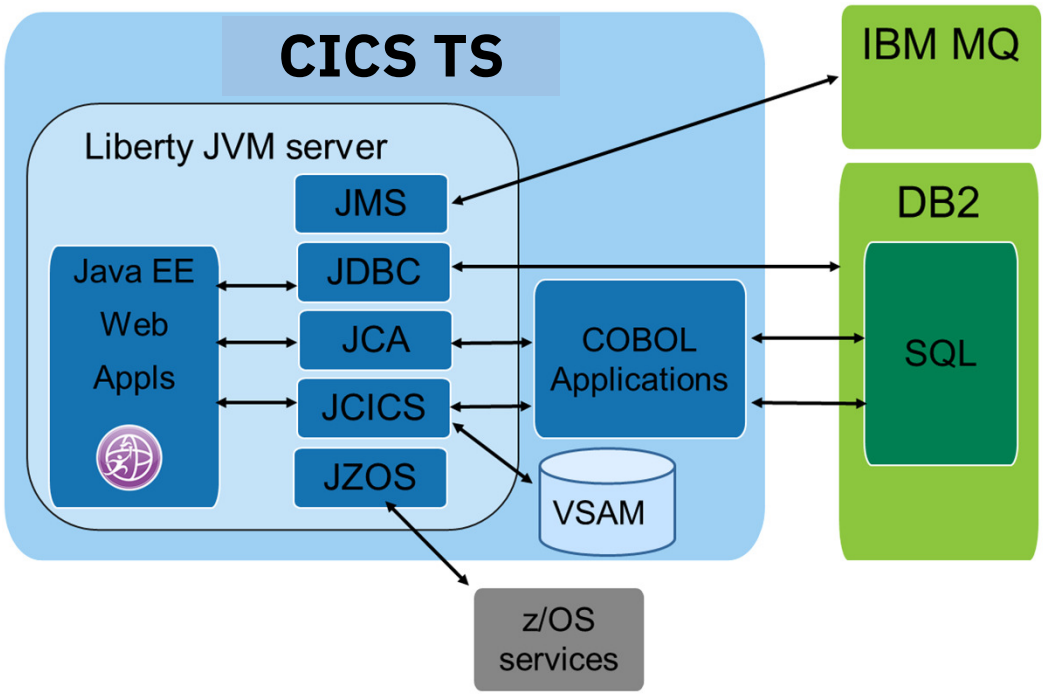
OpenID Connect Client feature

- Configure Liberty server to authenticate a request using a JWT token without writing any code
- Supports identity mapping
  - Map Subject in JWT to local registry user
  - Map distributed identity to SAF registry user via RACMAP

Liberty Admin Center

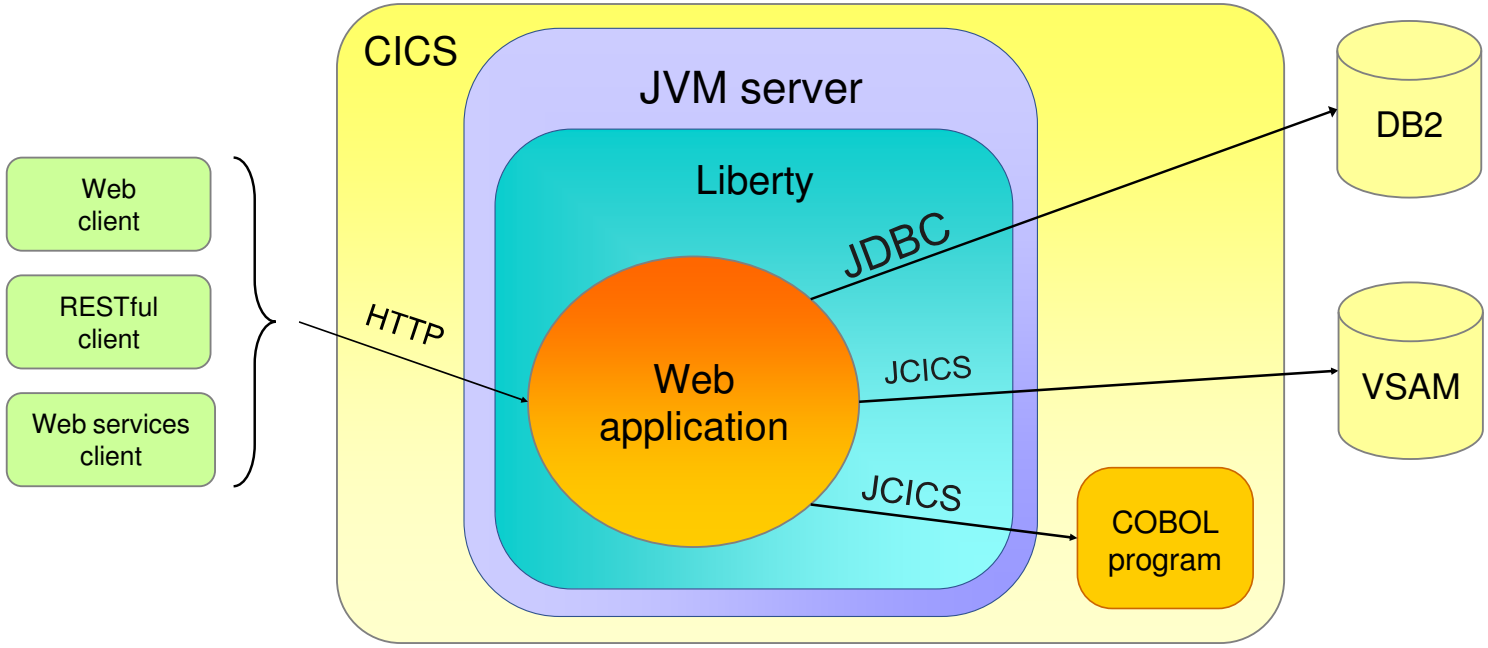
- The Liberty Admin Center is supported in CICS
- It is a site built into Liberty that allows you to
  - View and configure server.xml and related files
  - Examine applications running in the server
  - View live statistics about heap, CPU, and threads
- Available on CICS TS 5.5 with PH08321

Java in CICS



- ✓ **Java** supported in an integrated Liberty JVM server
- ✓ **JMS** support for MQ in client mode
- ✓ **JDBC** and **SQLJ** for Db2 data sources and other relational databases
- ✓ **JCICS** to provide access to CICS API including linking to other CICS programs
- ✓ **JCA** local ECI adapter supports porting of CICS TG ECI applications into CICS
- ✓ **JZOS** provides access to z/OS services such as console, files

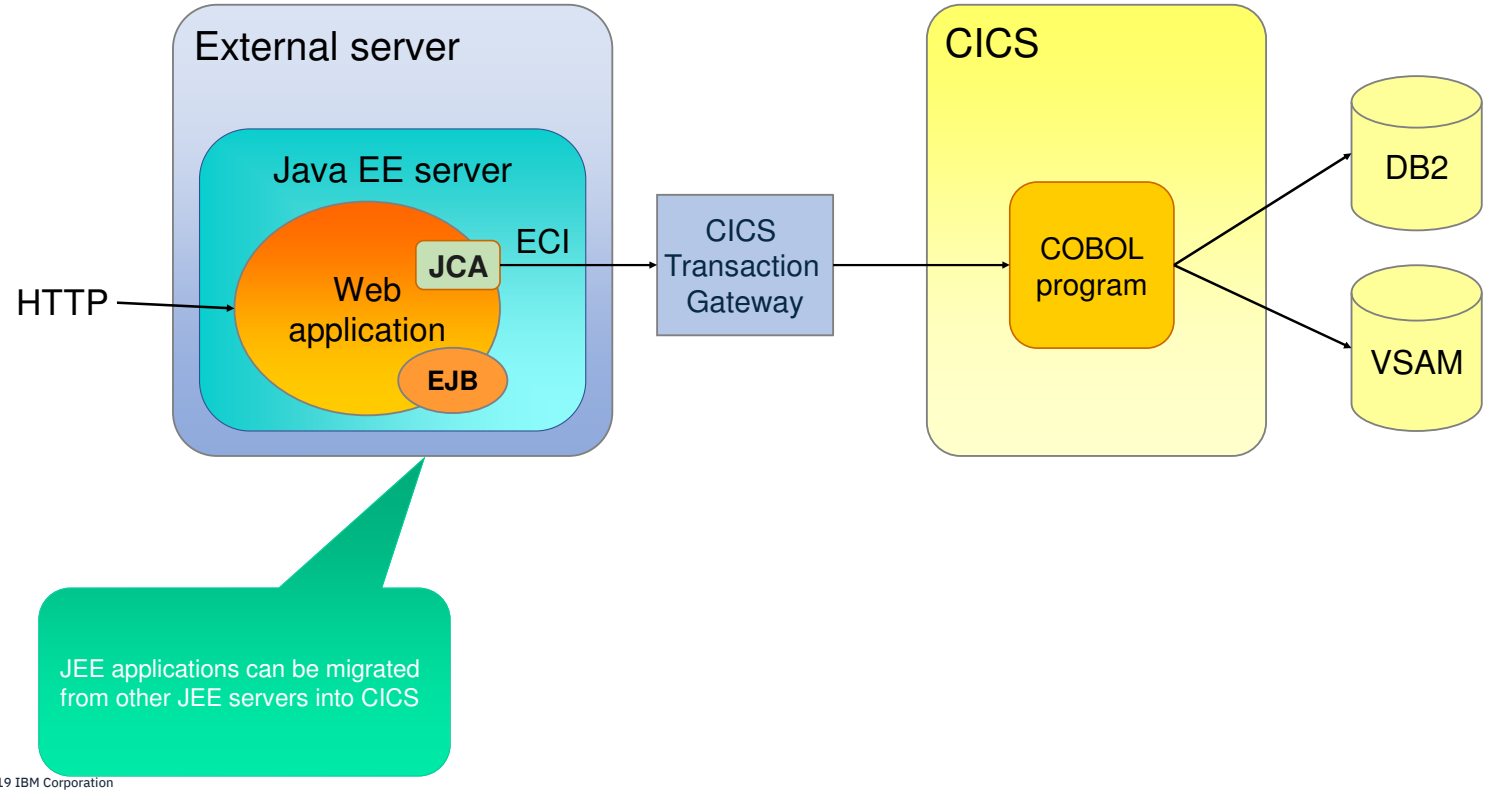
Why use Liberty in CICS?



- **Porting web** application to z/OS
  - JEE7 Web profile and JCA local ECI - 'lift and shift' porting from other JEE servers
- New **integration logic** for existing CICS services
  - Restful services or SOAP Web services interfacing existing CICS components
- **Java business logic** in CICS
  - Access to DB2 data – JDBC, EJBs, JPA or VSAM/JCICS

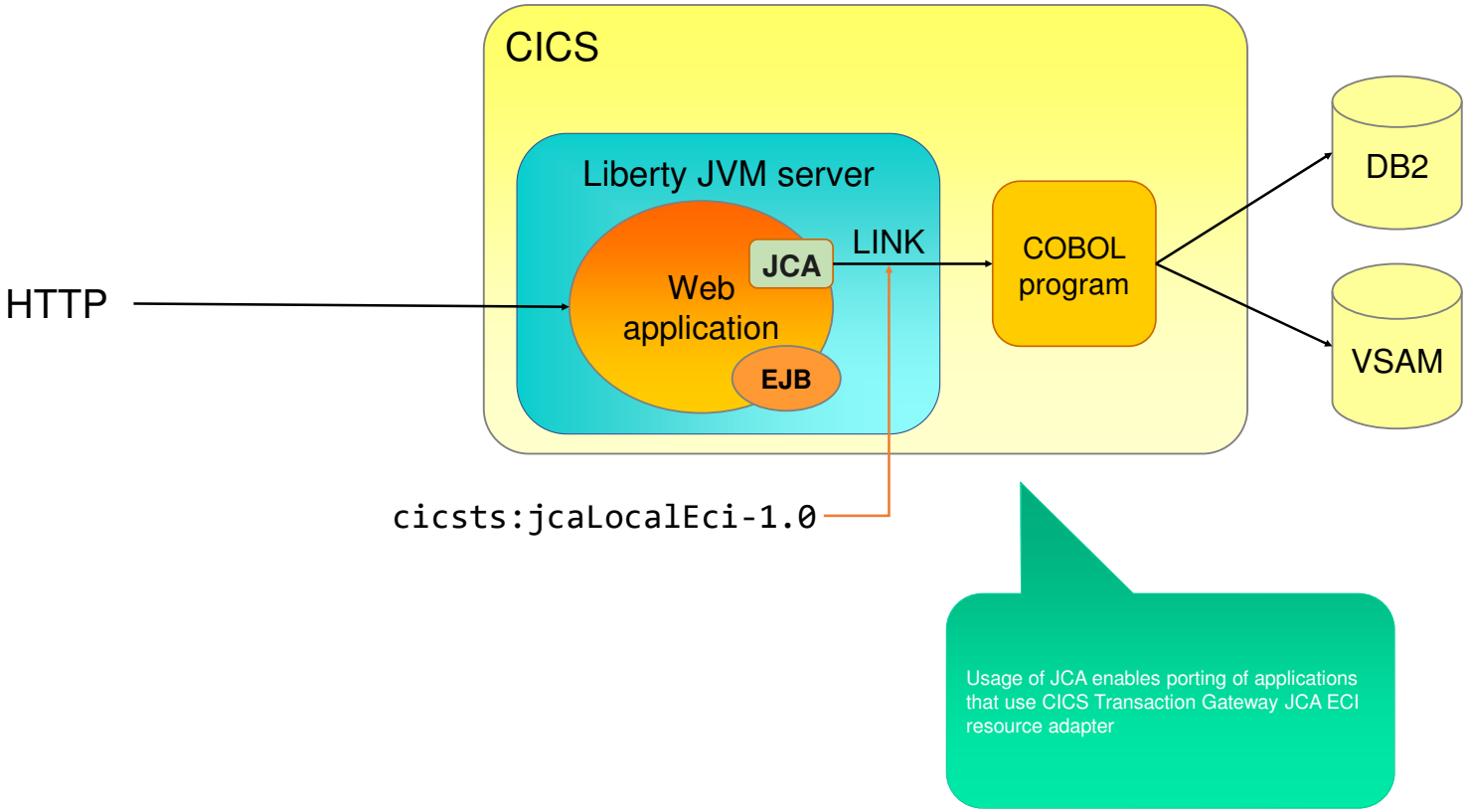
Liberty in CICS scenarios - 1

Porting Web applications to CICS



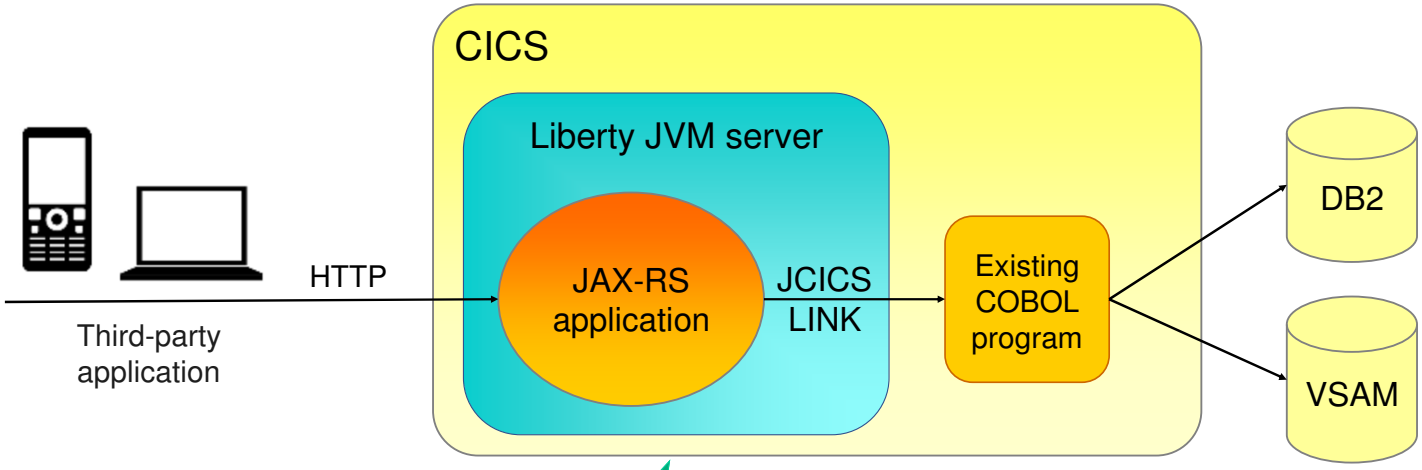
Liberty in CICS scenarios - 1

Porting Web applications to CICS



Liberty in CICS scenarios - 2

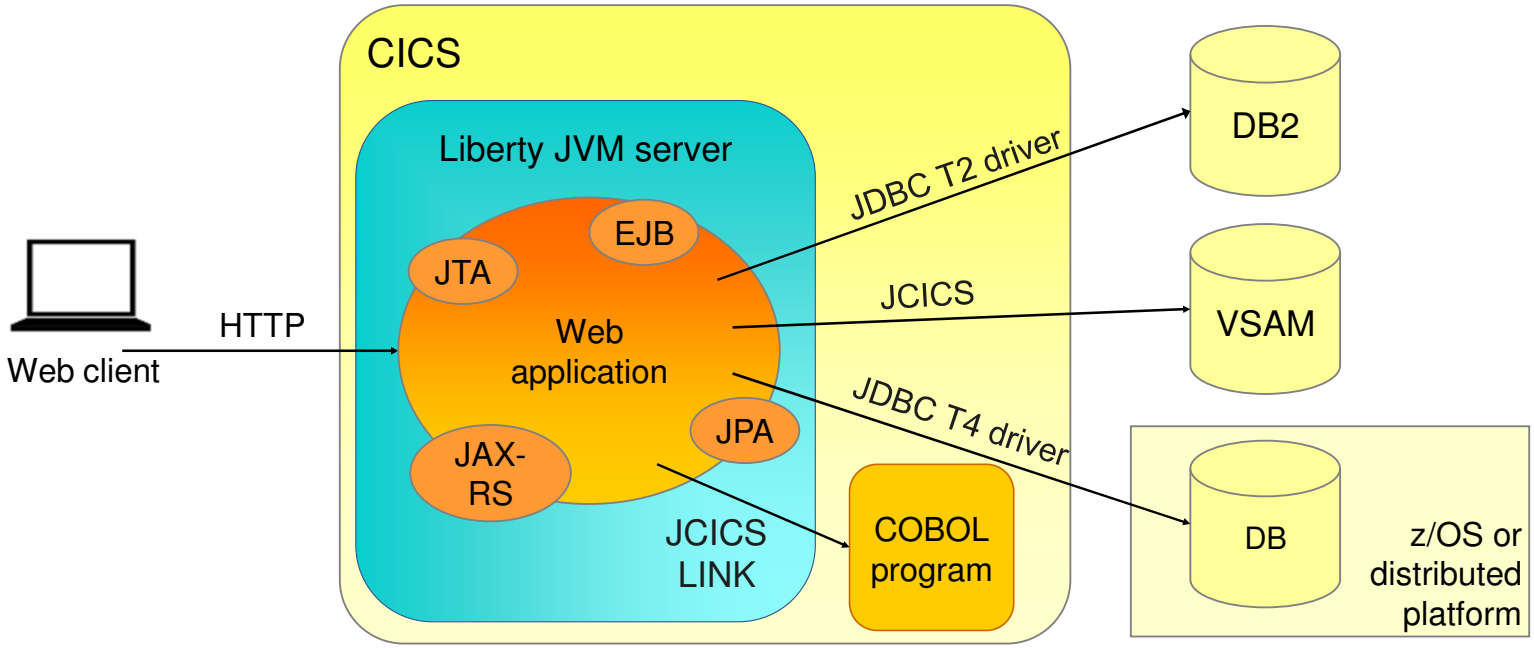
Integration with existing CICS services



RESTful (JAXRS) or Web service (JAXWS) applications can be easily built to provide new service interfaces to existing business process applications

Liberty in CICS scenarios - 3

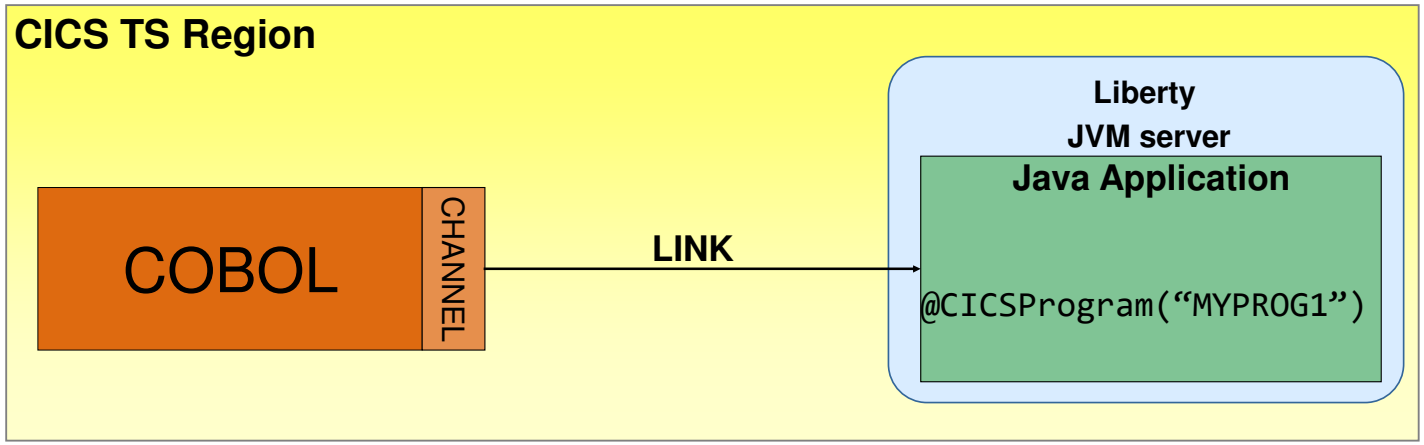
Java business logic in CICS



JEE offers ability to use wide variety of frameworks including EJB, JTA, JDBC, JPA, ManagedBeans to create new business applications which integrate with existing CICS applications or relational databases.

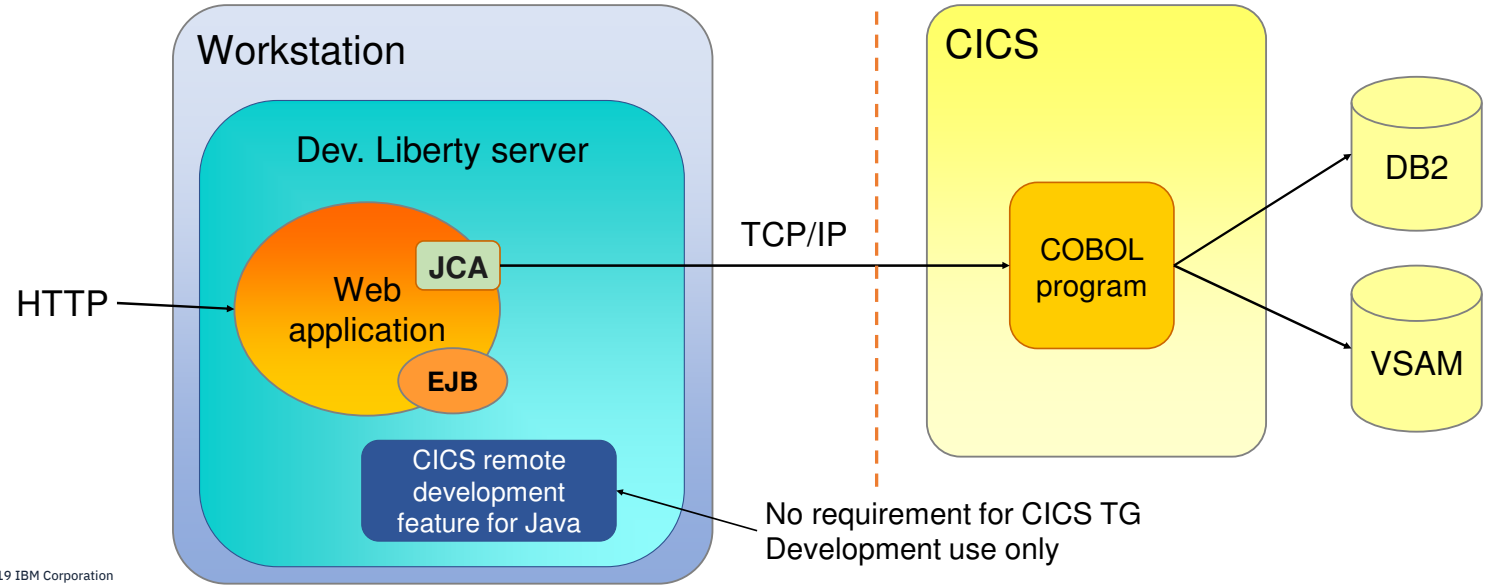
LINK to Liberty from non-Java programs

- Exchange data between mixed-language applications
  - using our standard mechanism of channels and containers
  - with this update non-Java programs can perform a LINK to a CICS Liberty application
    - from COBOL and PL/I, for example
  - it is now possible for non-Java programs to START a Java EE application in a CICS Liberty JVM server

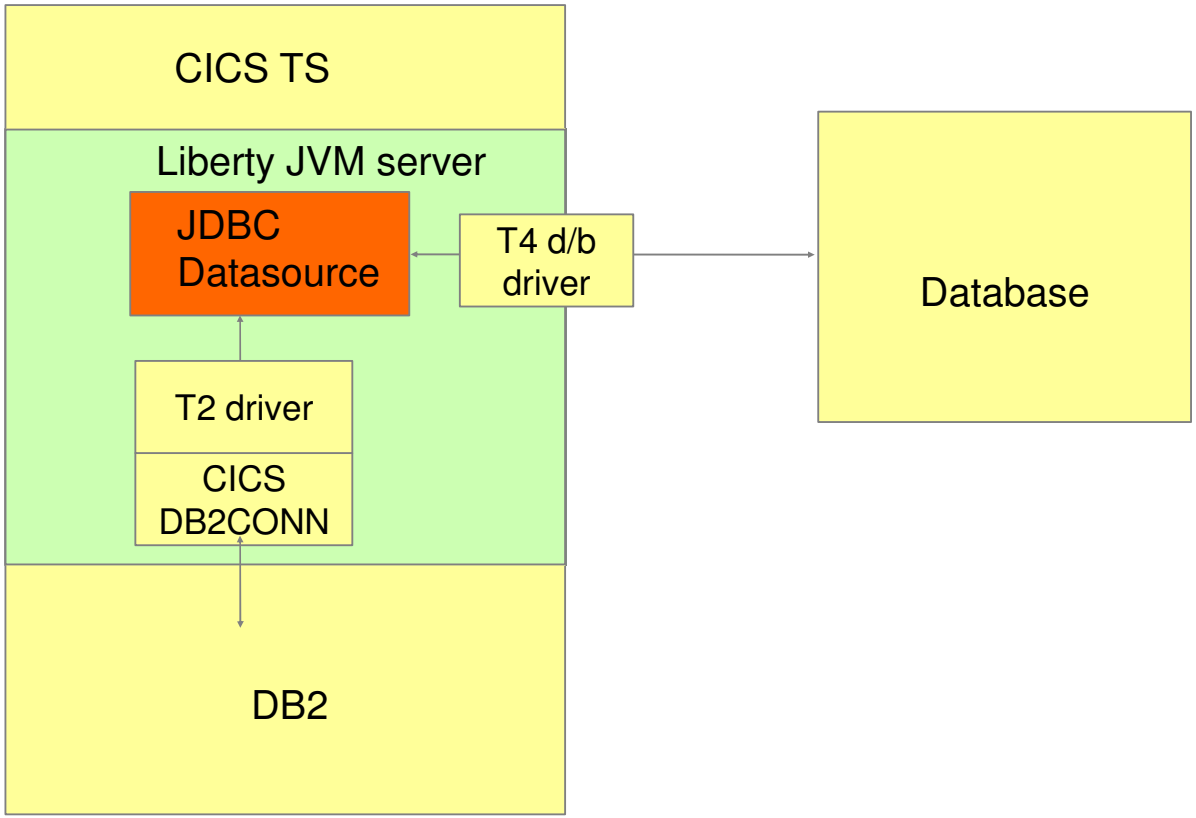


CICS TS Remote Development Feature for Java

- Provides a JCA resource adapter for use in Liberty to call a program in CICS
- The resource adapter connects to CICS using TCP/IP
- The feature is for development use only
- Available from the Liberty Repository for CICS TS
  - https://developer.ibm.com/wasdev/downloads/#asset/features-com.ibm.cics.wlp.jcaRemoteEci-1.0



Liberty JVM server – JDBC usage



CICS Liberty

- Can be used for
  Web interactions to include
  REST requests and web services

Application or application part is packaged as a WAR file (Web Archive) or EAR (Enterprise Archive)

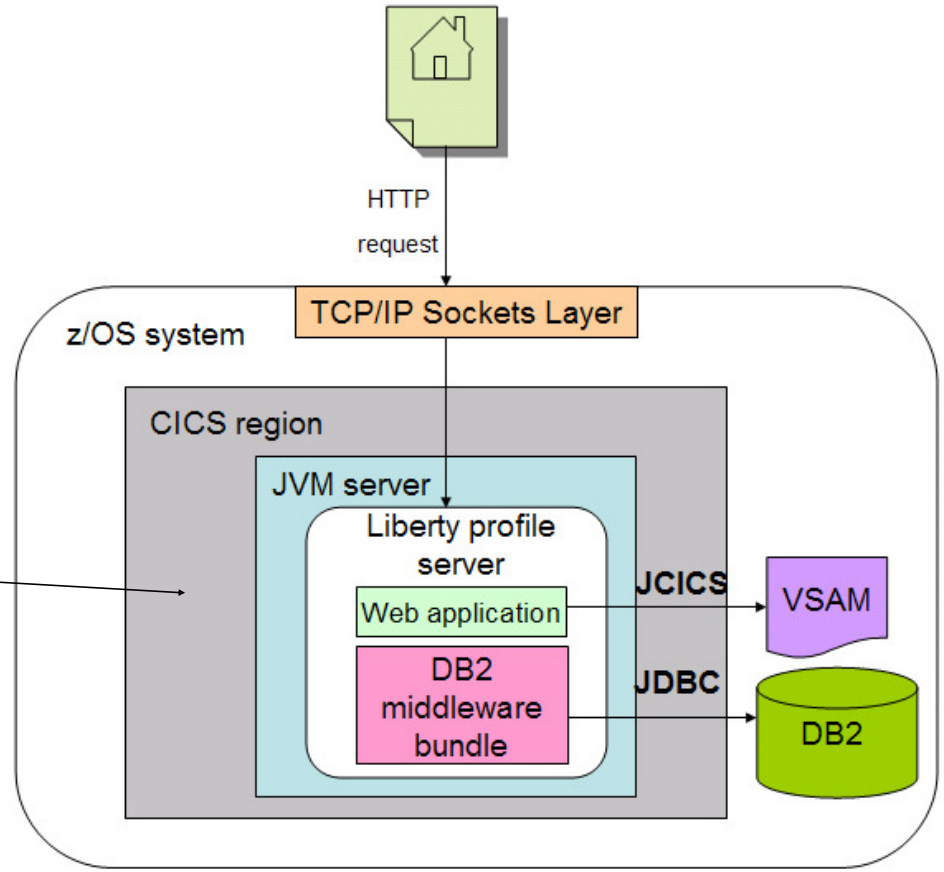


Diagram taken from CICS Knowledge Center

Advanced
Technology
Group

IBM

Deployment

Application deployment – Liberty JVM server

- Liberty dropins directory
  - For development/testing
- CICS Bundle resource
  - Web application(WAR), JEE archive (EAR), OSGi Application(EBA) (deprecated)
  - CICS-managed application deployment
- Liberty application definition
  - For manual deployment via server.xml
- Liberty shared bundle repository or global library
  - For shared components

Deployment – 1… Dropins

- To use the Drop-in directory (turned off by default):
  - Update server.xml to enable dropins

```
<applicationMonitor dropins="/test/applicationX/cicsjava/dropins"
    dropinsEnabled="true"
    pollingRate="5s" updateTrigger="polled"/>
```

  - FTP the WAR/EAR file in binary mode to drop-ins zFS directory
    - Directory is automatically created when JVM server is created
    - In default configuration:
      - /$WORK_DIR/applid/jvmserver/wlp/usr/servers/defaultServer/dropins

    - Liberty detects the deployed WAR file and installs/updates it
    - CICS is **not** aware of it, no CICS bundle life cycle
    - Not integrated with CICS security
    - Useful for development regions

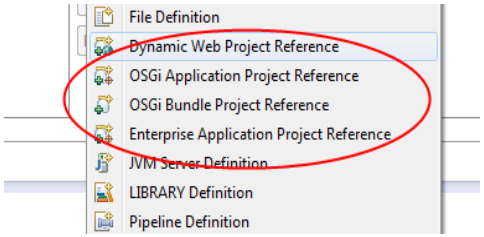
Deployment – 2… CICS Bundles

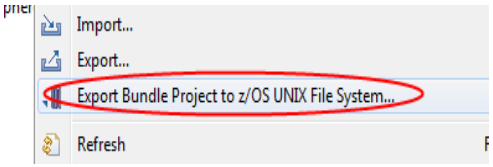
- Create CICS bundle in CICS Explorer

- Add EAR/WAR reference

- Export to zFS (or use build toolkit)
  - Maven / Gradle

- Define & install CICS BUNDLE resource with BUNDLEDIR attribute referencing zFS location
  - CICS Bundle disable/enable will cycle application





Deployment – 3. server.xml definitions

- Define applications in server.xml or an embedded file

```
<application type="war" id="LibertyWorld" name="LibertyWorld"
     location="/u/cicsjava/deploy/liberty.security.helloworld.war">
     <classloader commonLibraryRef="mqlib" />
     <application-bnd>
         <security-role name="testing">
              <user name="WAKELIN" />
         </security-role>
       </application-bnd>
</application>

<library id="mqlib">
  <fileset dir="/mqm/V7R1M0/java/lib" includes="*.jar" scanInterval="5s" />
</library>
```

Locally defined java library

testing role must be defined in application's web.xml

Userid to be given permissions,

Note: CICS Bundle defined applications are automatically added to installedApps.xml when CICS bundle enabled

Deployment – 4. Shared repositories

- Shared bundle repository – available to all OSGi bundle
  - i.e. bundles deployed in EBAs

```
<fileset dir="/mqm/V7R1M0/java/lib/OSGi" id="mqosgilib"
   includes="com.ibm.mq.osgi.java_7.1.0.4.jar"/>
<bundleRepository filesetRef="mqosgilib"/>
```

- Global library
  - available only to standalone Web applications (WARs / EARs)

```
<fileset dir="/mqm/V7R1M0/java/lib" id="mqlib" includes="*.jar"/>
<library filesetRef="mqlib" id="global"/>
```

New Parms in JVMProfile for Liberty

Some of the parameters

- Supplied sample is DFHWLP
- New Liberty options in JVMProfile for Liberty:
  - **WLP_INSTALL_DIR=&USSHOME;/wlp**
  - WLP_OUTPUT_DIR= default is WORKDIR/$APPLID/$JVMSERVER/wlp/usr
  - WLP_USER_DIR= default is $APPLID/$JVMSERVER/wlp/user/servers
  - -Dcom.ibm.cics.jvmserver.override.ccsid=
    use if other than LOCALCCSID in SIT is to be used (caution)
  - **-Dcom.ibm.cics.jvmserver.wlp.autoconfigure=** (meaning changed)
    true = CICS should build server.xml if it does not exist (V5.3)
  - -Dcom.ibm.cics.jvmserver.wlp.server.host= sets host name
  - **-Dcom.ibm.cics.jvmserver.wlp.server.http.port= port to listen on**
  - -Dcom.ibm.cics.jvmserver.wlp.server.https.port = SSL port to listen on
  - -Dcom.ibm.cics.jvmserver.wlp.server.name= Name for Liberty profile
  - -Dcom.ibm.cics.jvmserver.wlp.optimize.static.resources= (V5.2)

Advanced
Technology
Group

IBM

Security

Protecting Liberty apps with CICS security

- Default Tranid is CJSA
- URIMAP provides CICS authorization via Transaction Security
- URIMAP allows context switch to a 'user' transaction
  - Transaction Security (URL mapped to transaction)
  - monitoring and audit purposes
  - “Transaction class” support
- Each 'Invocation' (think Servlet Request) on a Hybrid Thread is also a CICS Transaction (has a Tranid, Task Context etc)
- Can provide
  - A single CICS (UOW) and CICS Managed JDBC or JTA (Java Transaction API)
  - Full JCICS API Access
    - Including LINK and access to VSAM
  - WLM (CICS WLM, Performance Classes etc)
  - Monitoring / Statistics
  - CICS Transaction Tracking / Association Data

CICS Liberty Security

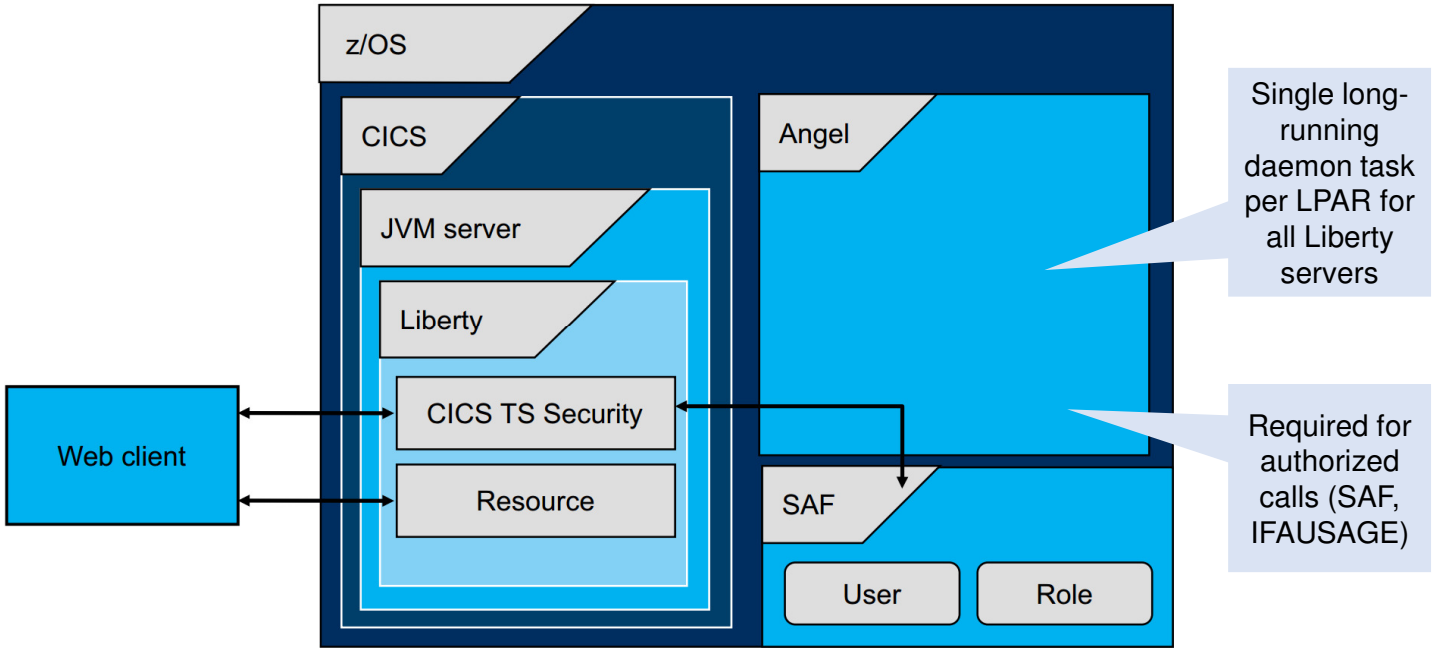
- CICS Liberty Security feature
  - zOSSecurity feature integrated with cicsts:security-1.0 feature
  - AppSecurity-2.0 Liberty feature fully supported

- Additional authentication options now supported in addition to basic auth
  - Form Login
  - SSL client authentication
  - Custom user registry
  - Basic registry
  - Trust Association Interceptor (TAI) / Java Authentication and Authorization Service (JAAS)

- New authorization options
  - JEE roles (defined in server.xml or RACF EJBROLE)

- CICS Transaction and Resource security remain supported
  - Authenticated userid is set as Task ACEE
  - Used for CICS Transaction and Resource security checks
  - Region user ID now used in monitoring (CMF), or task association, or GLUEs/TRUEs

Angel process



Angel Process

- WLP Angel process used to provide access to MVS authorized services
  - Password authentication
  - Role authorization (EJBROLEs)
- Start Angel process via:

  &USSHOME/wlp/templates/zos/procs/bbgzangl.jcl

  Query usage using MVS command

  - /MODIFY BBGZANGL,DISPLAY,SERVERS

SAF authentication setup

1) Permit the CICS region userid to access the Angel and its services
   BBG.ANGEL
   BBG.AUTHMOD.BBGZSAFM
   BBG.AUTHMOD.BBGZSAFM.PRODMGR
2) Setup an APPL class for Liberty (or reuse the CICS APPL)
3) Permit the CICS region to access authorization services
   BBG.SECPFX.<appl>
4) Permit the WSGUEST to access the angel APPL class
5) Permit each authenticated userid, to access the angel APPL class

Technology essentials – workload management

- Web server plug-in
  - Licensed for use with CICS TS and provided with WAS
  - Provides round-robin request distribution and session affinity management
    - http://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.3.0/com.ibm.cics.ts.java.doc/topics/dfhpj2_wsplugin.html

- IP load balancing
  - Port sharing/Sysplex Distributor allows you to use multiple JVM servers listening on shared IP endpoint
  - HTTP session state may need to be shared using session-database-1.0 feature

- CICSPlex SM – WLM
  - Dynamic DPL allows DPL requests to AORs

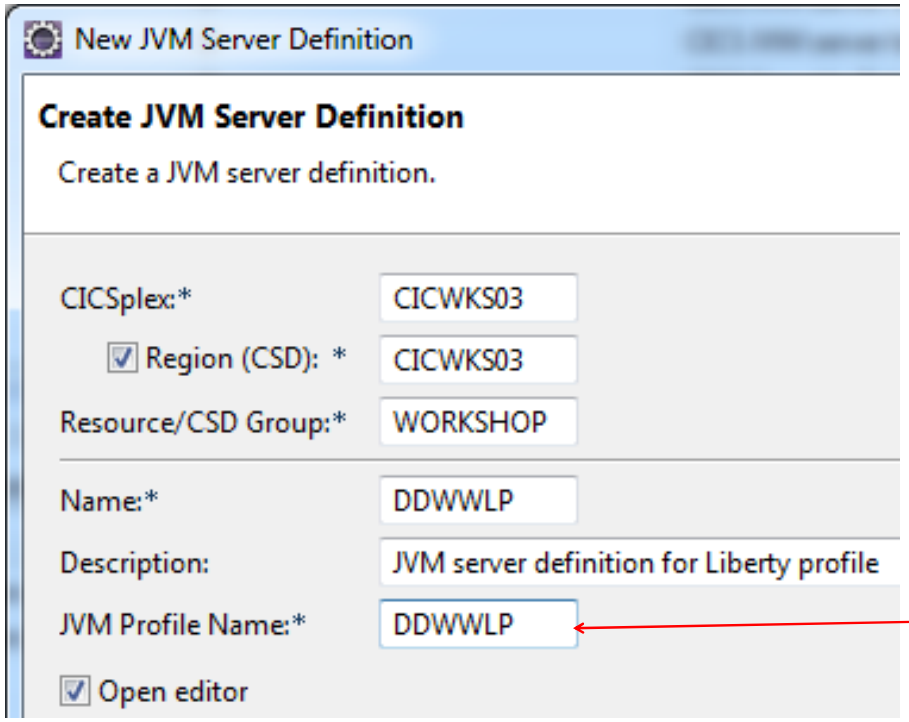
Advanced
Technology
Group

IBM

CICS – example servlet

Create a Liberty JVMSERVER

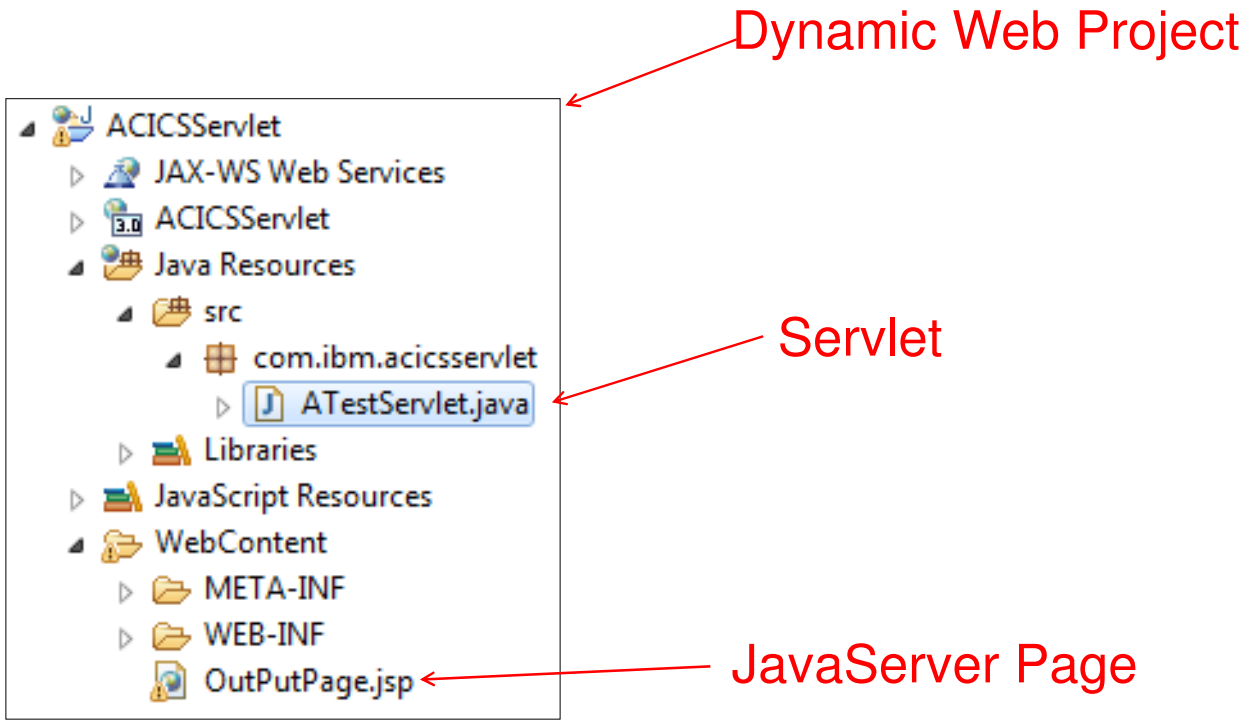
- Create and install a JVMSERVER resource in Explorer, CEDA, or CPSM



JVMProfile discussed previously

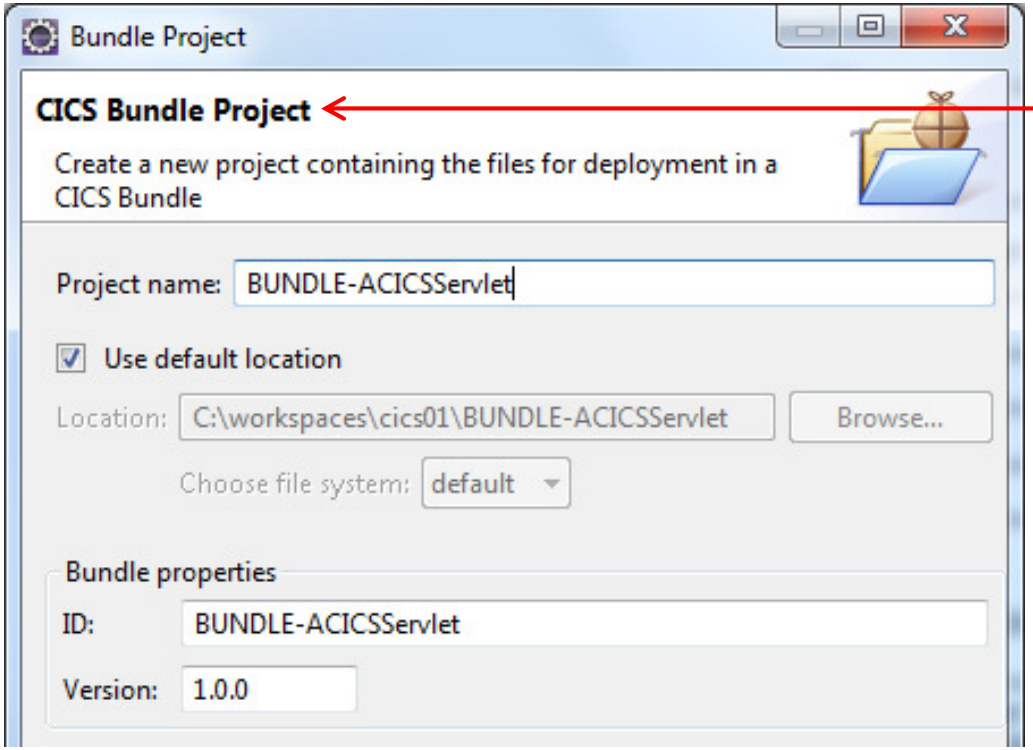
Create a Servlet

- Create your Servlet…



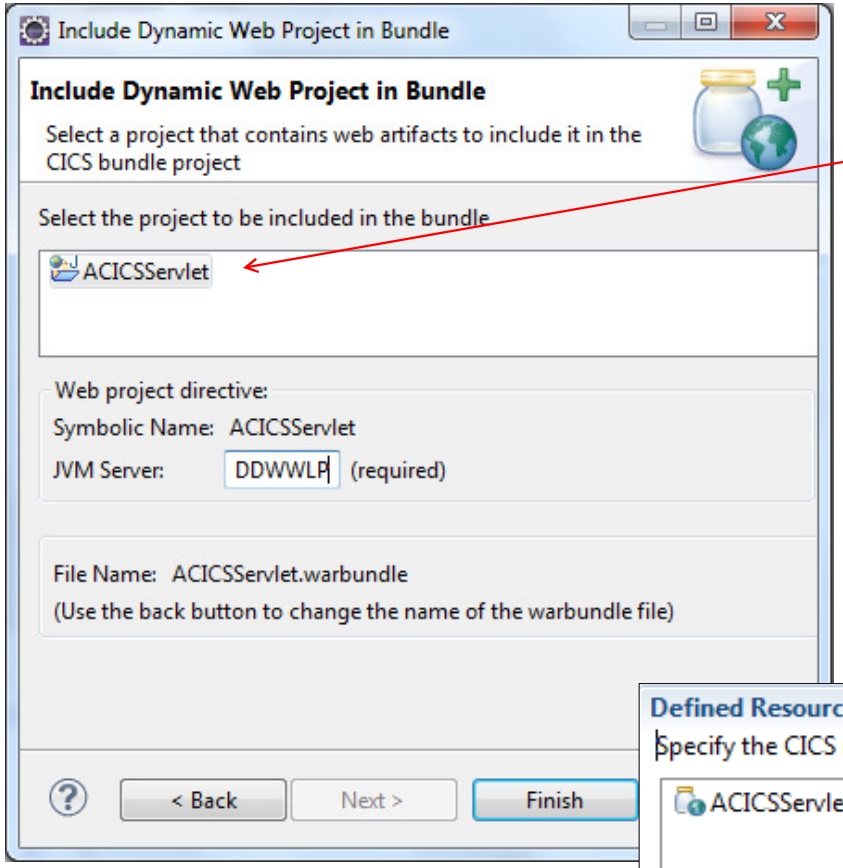
Create a CICS BUNDLE

- Create a CICS BUNDLE



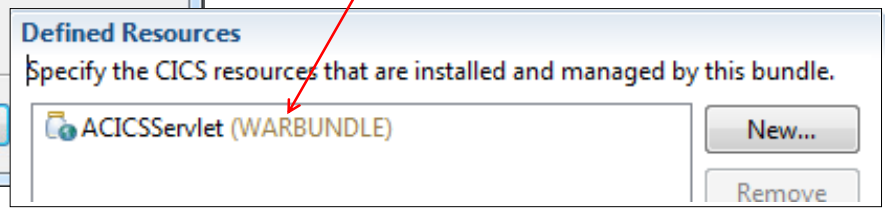
CICS BUNDLE Project

Put your Servlet into a CICS BUNDLE



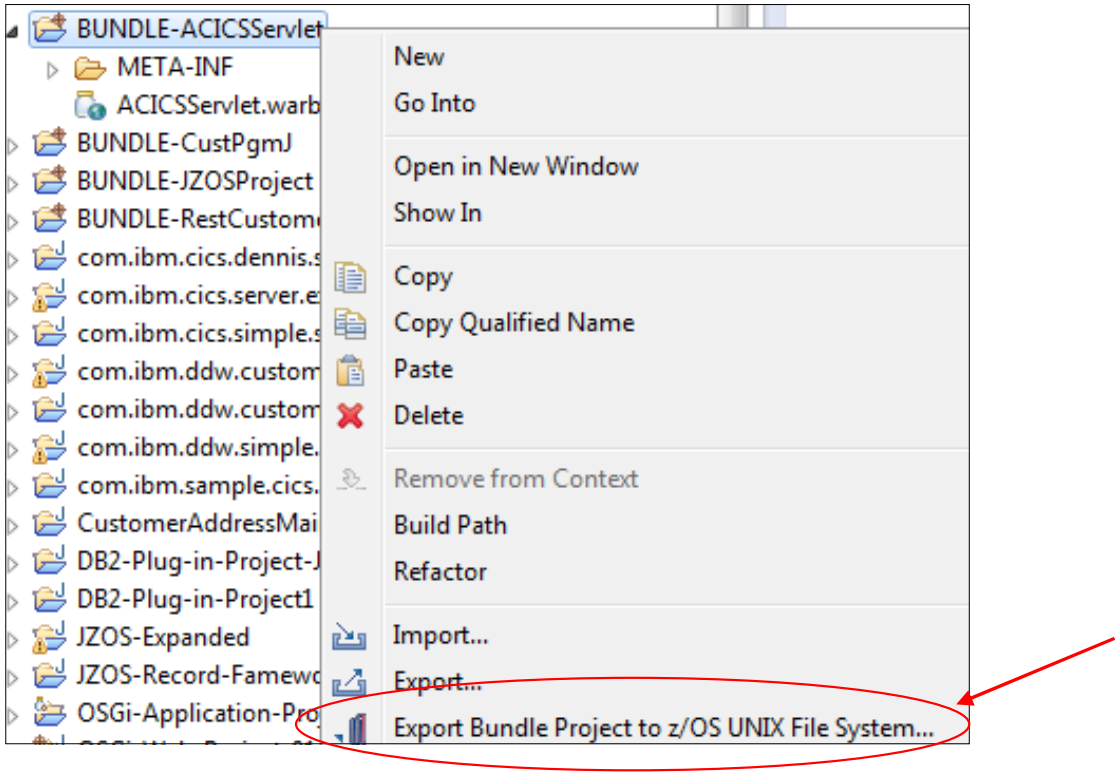
Add the Servlet to the CICS BUNDLE in the manifest editor

The Servlet was added to the CICS BUNDLE



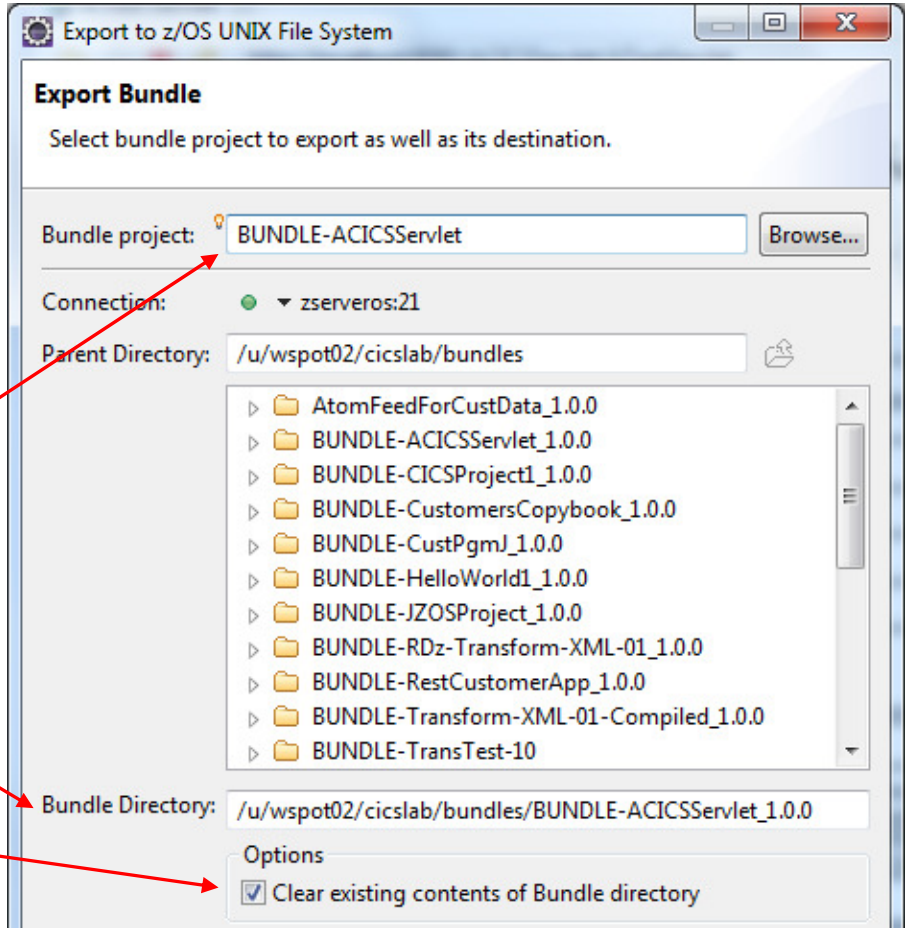
Deploying the Application to USS

- Export your CICS bundle project from your workstation to USS on z/OS



Place the application on z/OS USS

- Place your CICS bundle on USS on z/OS (the export wizard)



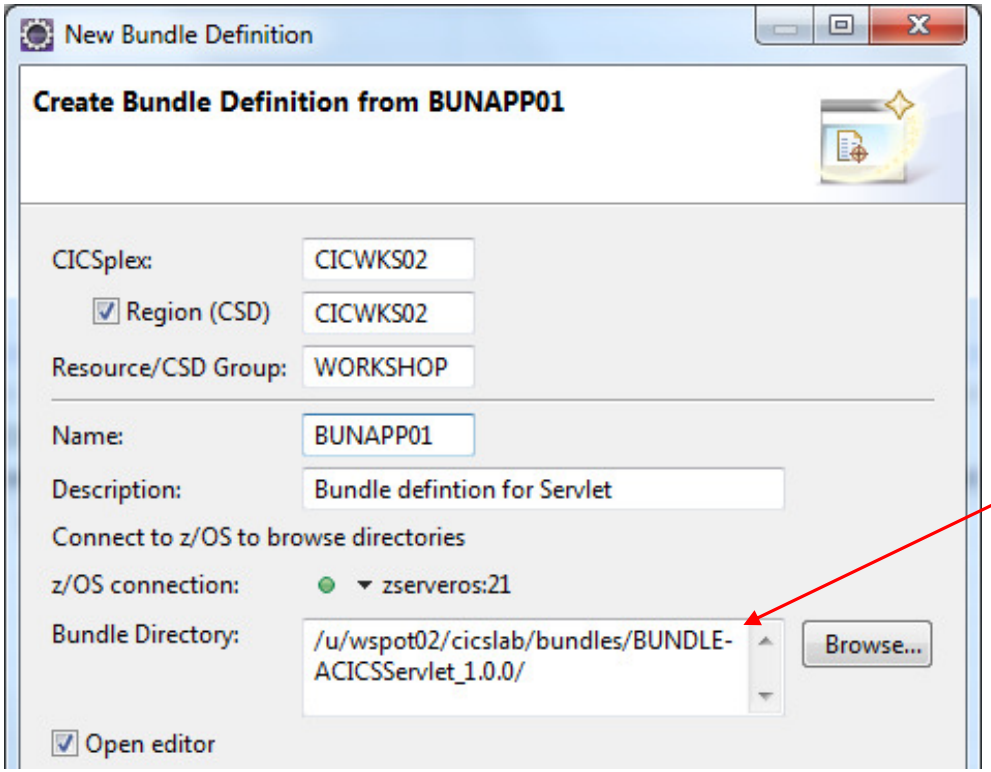
BUNDLE to be exported to z/OS

Full directory

Remove anything that was in the directory

Create and Install CICS Bundle Definition

- Create and install a CICS bundle definition pointing to your servlet on USS



The bundle directory on USS

Check CICS Bundle Definition in Region

- The servlet is installed and is ready…

Bundle Definitions | Bundles | Bundle Parts | OSGi Bundles | OSGi Services | Name:

CNX0211I Context: CICWKS02. Resource: BUNDLE. 1 records collected at Aug 25, 2013 8:18:27 PM

Region	Name	Bundle ID	Major Versi...	Minor Versi...	Micro Versi...	Status	Install Time	Enabled Co...
CICWKS02	BUNAPP01	BUNDLE-ACICSServlet	1	0	0	✓ ENABLED	Aug 25, 201...	1

Resource name

Name of BUNDLE

BUNDLE Version

The BUNDLE is Enabled

Test Servlet in a Browser

http://zserveros.demos.ibm.com:19002/ A Test Servlet

A Test Servlet

This output is from lab L70, it is a very simple servlet

The counter = 1 Submit

Servlet Request URL:	http://zserveros.demos.ibm.com:19002/ACICSServlet/ATestServlet
Servlet Method:	GET
Context Path:	/ACICSServlet
Servlet Path:	/ATestServlet
Query String:	
Body Content Length:	-1
Parameters:	-
	User-Agent: Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.1; Trident/6.0)
	Cookie: UnicaNIODID: N2XHTDo7ie9-WfgWQUf

CICS-Liberty – REST Support

```
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;
// some stuff missing
@WebServlet("/INVStockFundAjax")
public class INVStockFundAjax extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
              throws ServletException, IOException {
// do stuff here
    response.addHeader("Content-Type", "text/html");
    if (stockFundInfo == null) {
      response.getWriter().write("<b>Symbol not found</b>");
      response.setStatus(404);
    } else {
    response.getWriter().write(" write the symbol found stuff here ");
    response.setStatus(200);
  }
}
```

Also supports JSON4J

Also supports JAX-RS (see next page)

CICS-Liberty – REST Support

```
package com.ddw.sample.json.service;
import java.util.HashSet;
import java.util.Set;
import javax.ws.rs.core.Application;
public class SayHelloApplication extends Application {
    //List the JAX-RS classes that contain annotations
    public Set<Class<?>> getClasses() {
         System.out.println("SayHelloApplication: Instaciated");
         Set<Class<?>> classes = new HashSet<Class<?>>();
         classes.add(com.ddw.sample.json.service.SayHello.class);
         return classes;
    }
}
```

Also supports JSON4J

```
package com.ddw.sample.json.service;
import javax.ws.rs.GET;  import javax.ws.rs.POST;  import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
@Path("/hello")
public class SayHello {
    @GET
    public String helloGet() {
        return "{ \"HelloText\", \"Howdy Partner\" }";
    }
    @POST
    public Response helloPost() {
       return Response.ok("{ \"HelloText\", \"Hi, Neighbor\" }").status(201).build();
    }
}
```

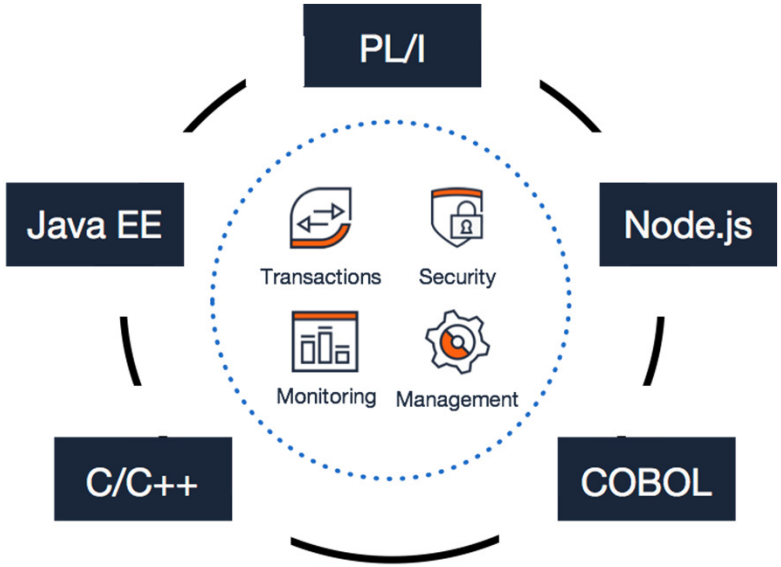
Advanced
Technology
Group

IBM

Summary

Unparalleled mixed-language application server

- **IBM CICS Transaction Server** has evolved to become the world's most powerful mixed language application server.
- Applications can share core programming contexts such as **transactionality**, **security**, **monitoring,** and **management**, regardless of the language its components are written in, and take full advantage of IBM Z.
- Developers can create incredible mixed-language applications, that include **Jakarta EE, Spring Boot, Eclipse MicroProfile, and Node.js** capabilities, together with traditional languages like COBOL, C/C++, and PL/I, with first-class interoperability.



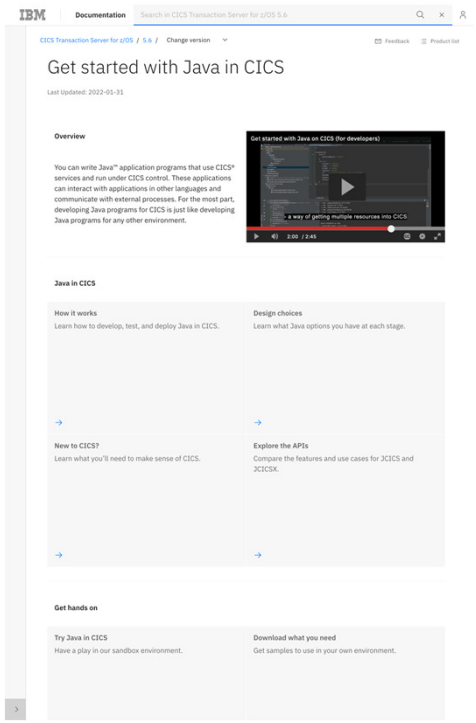
So… what’s the deal?

- Modernize the presentation interfaces of your CICS application
  - Replacing 3270 screens with web browser and RESTful clients
- Use Java standards-based development tools
  - To package, co-locate, and manage a web client with other existing CICS applications
- Already use Liberty profile applications in WebSphere Application Server and want to port them to run in CICS
- Already use Jetty or similar servlet engines in CICS and want to migrate to a web container that is based on the Liberty profile
- Want to use DataSource definitions to access DB2databases from Java
- Want to coordinate updates made to CICS recoverable resources
  - With updates made to a remote resource manager via a type 4 JDBC database driver, using the Java Transaction API (JTA)
- Want to develop services that follow REST principles using JAX-RS
- Want to develop applications through support of a standard, annotation-based model using JAX-WS

Enhanced Developer productivity

Improvements to Java getting started documentation

- To help Java developers to get started with applications in CICS, updated information is available
- Simple overview, CICS concepts and access to resources, such as samples, videos, and tutorials



New samples on GitHub

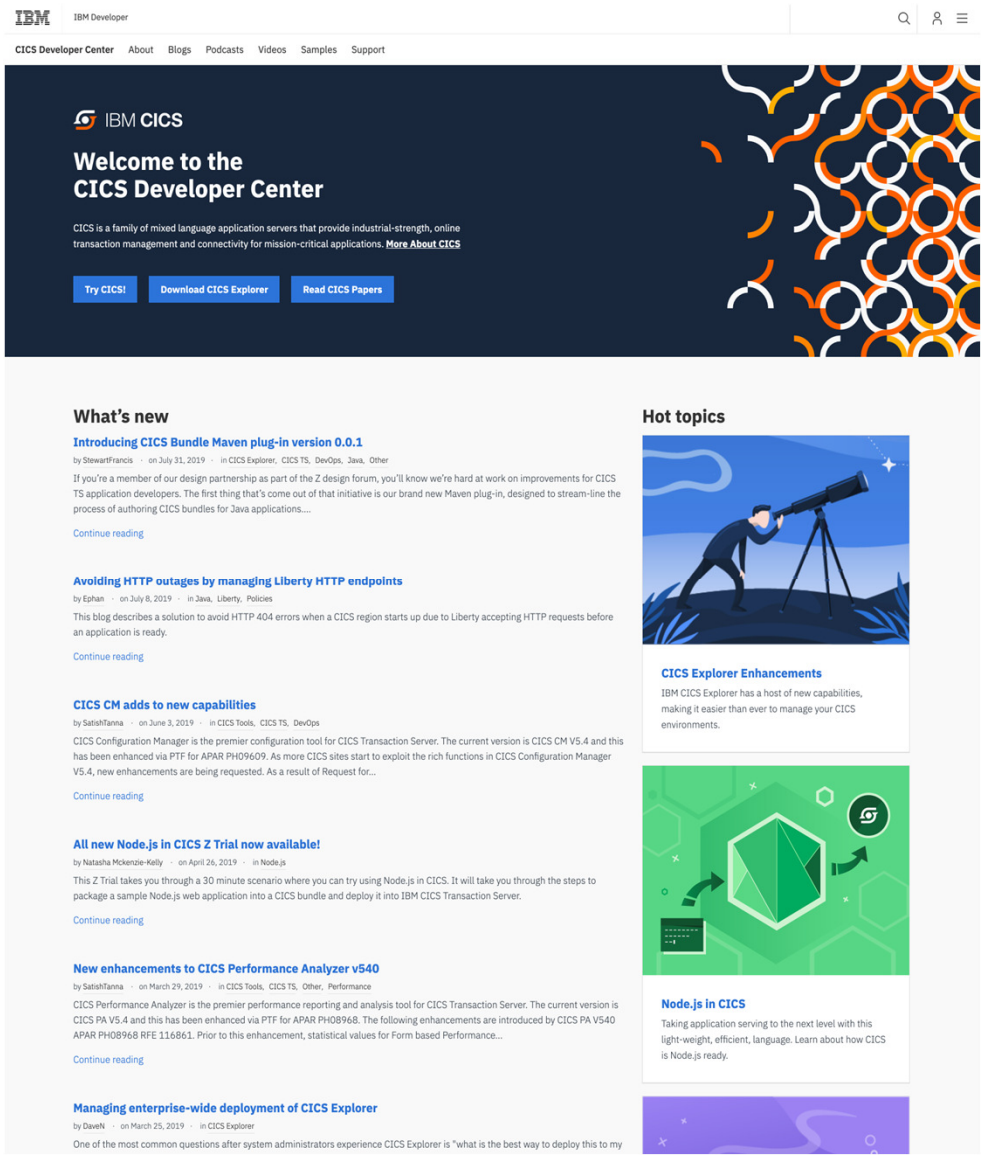
- Spring boot samples and tutorials
- How consume CICS events in Java
- JCICS samples include the higher level api

Get started with Java on CICS

- We have video series aimed at getting started with Java on CICS
  - Developing a RESTful Web application for Liberty in CICS
  - Architecting Java solutions for CICS
  - Extending a CICS web application using JCICS

Keeping up to date

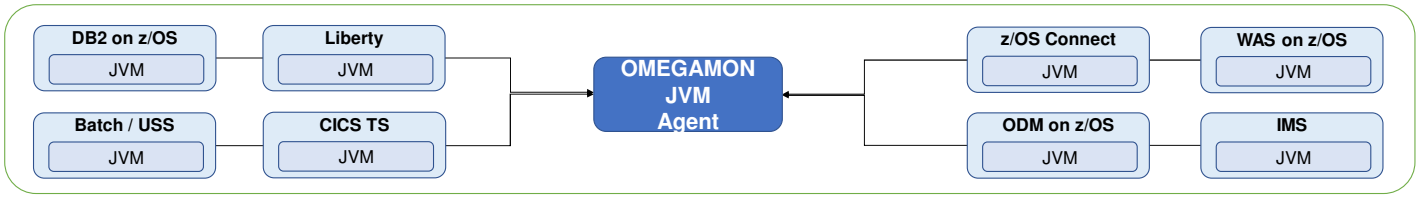
- For all the latest developments subscribe to the blogs at developer.ibm.com/cics



OMEGAMON Monitoring for JVM

IBM Tivoli OMEGAMON XE on z/OS Monitoring Feature for JVM V5.3.0 is a new feature providing resource level monitoring of *all* Java Virtual Machines (JVMs) on z/OS

- Enable users to view all active JVMs within a single screen, regardless of subsystem type
- Auto-discover all online JVMs within seconds, including subsystem type
- Identify problematic thread and locking issues, sub-optimal JVM garbage collection performance, CPU performance issues and offers drill-downs into detailed JVM environment information.
- Enable users to be alerted to problems within JVM performance, isolate the issue, and identify the root cause quickly.
- Workspaces are provided in both the enhanced 3270 user interface (3270UI) and Tivoli Enterprise Portal (TEP)



Lab Exercises (Java)

- L34 – Simple OSGi Application
- L70 – Servlet
- L72 – Servlet front-end to CICS-provided COBOL Catalog Sample application (uses JZOS to get the COMMAREA to COBOL)
- L92 – Simple JAX-RS ‘Hello World’
- L93 – JAX-RS, JSON4J, JZOS
- L95 – Simple JAX-WS ‘Hello World’
- L96 – JAX-WS and JAXB
- L97 – z/OS Connect EE

Advanced
Technology
Group

IBM

More on Security

- Can use the CICS Liberty security feature to
  - authenticate users
  - authorize access to web applications through Java EE roles
  - (provide integration with CICS transaction and resource security??)
- Can use CICS resource security to authorize users to
  - manage the lifecycle of the JVMSERVER
  - manage the lifecycle of Java web applications that are deployed in a CICS BUNDLE
- The default transaction for running web requests is CJSA
  - can configure CICS to use a different transaction by using a URIMAP (type JVMSERVER)
- The default user ID for running web requests is the CICS default userID
  - a URIMAP can specify a static userID
  - the web request can contain a userID in its security header (it takes precedence)

Authenticating users in Liberty

- You can configure CICS security for all web applications that run in Liberty
  - the web application will only authenticate users if it includes a security constraint
  - The security constraint is defined by an application developer in the deployment descriptor (web.xml) of the Dynamic Web Project or OSGi Application Project
  - The security constraint defines what is to be protected (URL) and by which roles
  - A <login-config> element defines the way a user gains access to web container and the method used for authentication
- Tasks that are authenticated in CICS using Liberty security
  - can use the userID derived from any of the Liberty application security mechanisms
    - to authorize transaction and resource security checks in CICS
- Any of the application security mechanisms supported by Liberty are supported in CICS
  - HTTP basic authentication, form login, SSL client certificate authentication, identity assertion using a custom login module, JACC, JASPIC, or a Trust Association Interceptor (TAI)

Authorizing users to run applications in a Liberty JVM server

- To authorize access to Java applications in Liberty
  - You can use Java application security roles
  - You can use CICS transaction and resource security
- Applications are secured by providing an authorization constraint (<auth_constraint>) element in the deployment descriptor (web.xml)
  - Use an <application-bnd> element in the <application> element of your server.xml
    - describes the user/group to role mappings directly in XML
  - Use <safAuthorization> in your server.xml
    - allows users/groups role membership to be mapped by SAF (typically using EJBROLEs)