



Redbooks

Migrating Applications to WebSphere Application Server V5 for z/OS

Sabine Holl	sabine_holl@at.ibm.com
Alex Louwe Kooijmans	nl53347@nl.ibm.com
Kevin J. Senior	kev_senior@it.ibm.com



Agenda

- Migration Overview
- Infrastructure Migration
- Application Migration
 - ▶ From WAS V3.5
 - ▶ From WAS V4.01
- ▶ Development environment migration



Migration overview

- Development environment
- Actual application code
- Application deployment procedures
- Test runtime environment
- Production runtime environment



Runtime migration

- Monoplex migration
 - No co-existence
 - Co-existence
- Sysplex migration



Monoplex with no co-existence

z/Series - S/390

LPAR 1	LPAR 2
z/OS	z/OS
WAS v4.01	WAS v4.01
Production	Test

Current LPAR configuration



Monoplex with no co-existence

z/Series - S/390

LPAR 1	LPAR 2	LPAR 3
z/OS	z/OS	z/OS
WAS v4.01	WAS v4.01	WAS v5
Production	Test	Test

Migration configuration



Monoplex with no co-existence

z/Series - S/390

LPAR 1	LPAR 2	LPAR 3
z/OS	z/OS	z/OS
WAS v5		WAS v5
Production		Test

Final production configuration



Monoplex with co-existence

z/Series - S/390

LPAR 1	LPAR 2
z/OS	z/OS
WAS v4.01	WAS v4.01 Test
Production	WAS v5 Test

Initial migration configuration



Monoplex with co-existence

z/Series - S/390

LPAR 1	LPAR 2
z/OS	z/OS
WAS v5	WAS v5
Production	Test

Final production configuration



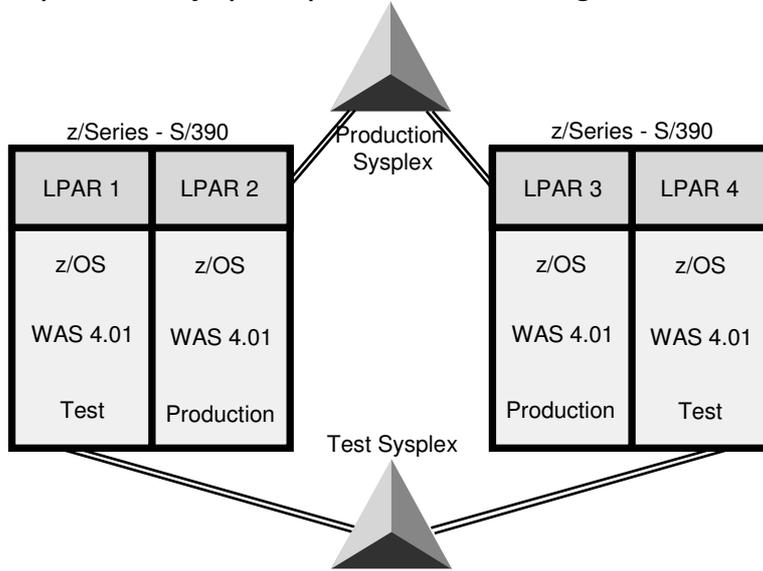
Sysplex migration

Things to consider with WAS V4 and WAS V5 co-existence

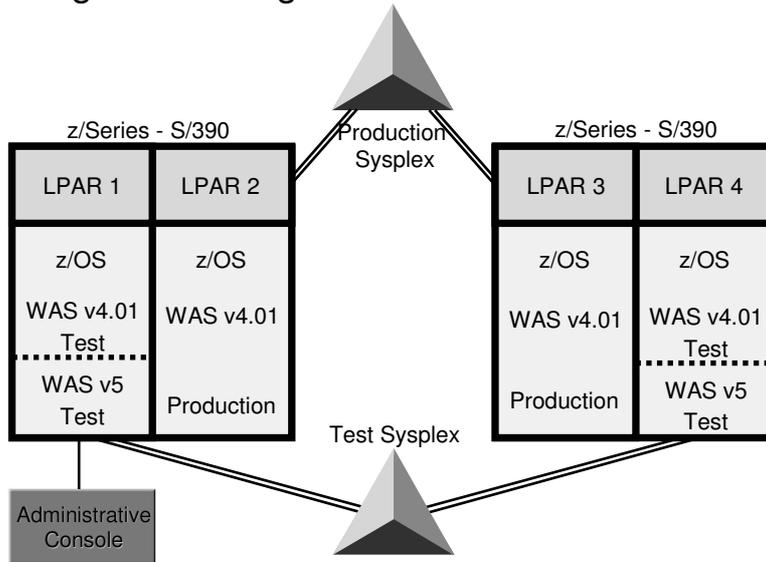
- Use STEPLIBs for WebSphere for z/OS V5
- Make sure TCP/IP ports used in WebSphere for z/OS V5 are unique and do not conflict with WebSphere for z/OS and OS/390 V4.01.
- Make sure WLM application environment names are unique
- Establish unique RACF profiles for each of the WAS environments
- Plan server names and procedure names for WebSphere for z/OS V5
- PLAN - PLAN - PLAN is basically the name of the game!



Initial parallel sysplex production configuration

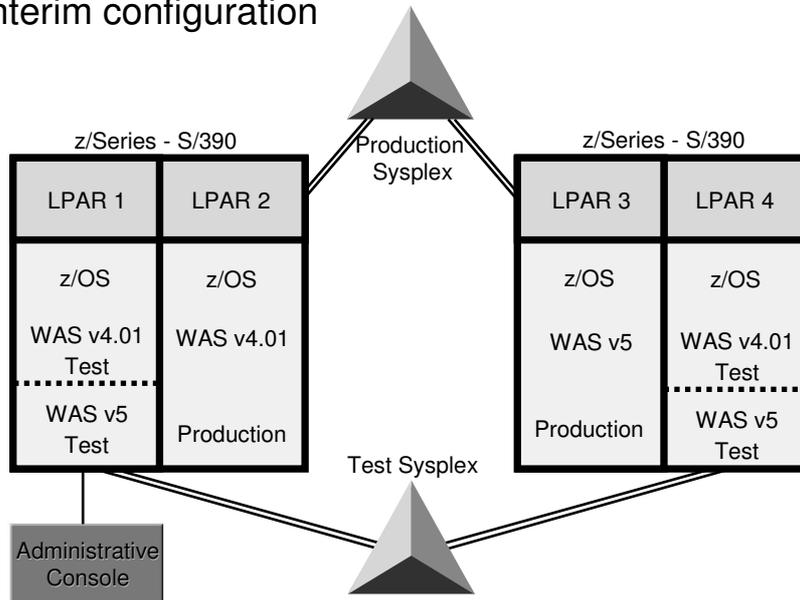


Initial migration configuration

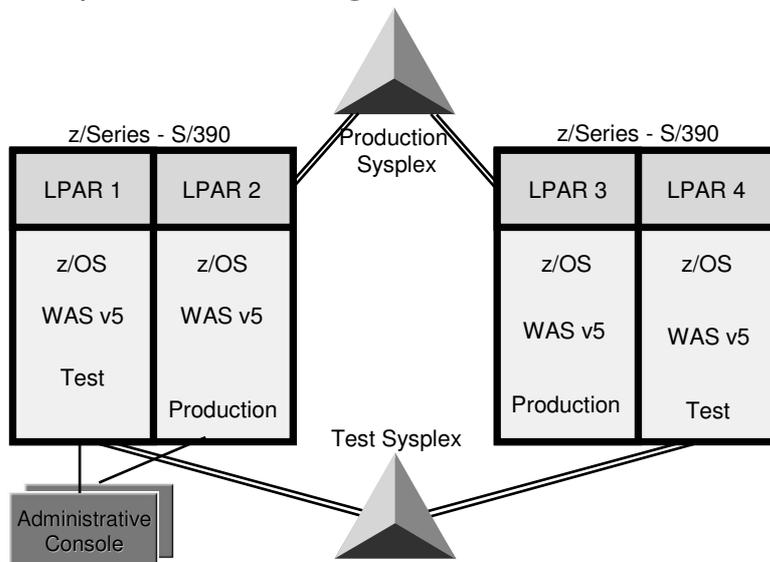




Interim configuration



Final production configuration





Application migration from WAS V3.5

Migration Issues
Packaging/Assembling
Web applications
Constructing Enterprise Applications
Deployment procedures
Security
ASCII/EBCDIC
Connectors
JDBC/SQLJ, Datasources
Plug-In
Migration tools



Web applications (WAR files)

- Visually inspect code
- Import in WSAD V5 and check warnings, error messages and deprecations
- Use CACT tool (Class API Check Tool)
http://www7b.software.ibm.com/wsd/library/techarticles/0208_cocasse/0208_cocasse.html
- HTTP sessions



ASCII/EBCDIC considerations

- WAS V5 default JVM encoding of ISO-8850-1 (US ASCII)
- Applications accessing EBCDIC encoded files need to change codepage:

```
private String readHFS(String file) {
String theString = "file not read properly";
String fileEncoding = null;
try {
// file.encoding is the default code page of the JVM
fileEncoding = System.getProperty("file.encoding");
System.out.println("Default file.encoding: " + fileEncoding);
// input: CP1047 to ASCII
FileInputStream fis = new FileInputStream(file);
BufferedReader r =
new BufferedReader(new InputStreamReader(fis, "CP1047"));
theString = r.readLine();
System.out.println("<<< String: " + theString);
}
catch (IOException IOEx)
{ IOEx.printStackTrace();}
catch (Exception ex)
{ ex.printStackTrace();}
return theString + ". " + "The default encoding for the JVM is: " + fileEncoding;
}
private void writeHFS(String file) {
try {
// output : ASCII to Cp1047
FileOutputStream fos = new FileOutputStream(file);
Writer w = new BufferedWriter(new OutputStreamWriter(fos, "CP1047"));
w.write(writeData);
w.flush();
w.close();
}
catch (IOException IOEx)
{ IOEx.printStackTrace();}
catch (Exception ex)
{ ex.printStackTrace();}
}
```



Connectors / JDBC / SQLJ

- CCF connectors are no longer supported -> use WSADIE V5 to migrate to J2C
- WAS 3.5 Applications using JDBC 2.0 are still working if coded as following:

```
import java.sql.*;
import javax.naming.*;
InitialContext ctx = new InitialContext();
javax.sql.DataSource ds =(DataSource)ctx.lookup("jdbc/myDS");
java.sql.Connection conn = ds.getConnection();
```

- WAS V3.5 early support of connection manager and pooling is no longer supported – code such as following needs to be changed

```
import java.sql.*;
import javax.naming.*;
import java.util.Hashtable;
import com.ibm.ejs.dbm.jdbcext.*; // Early connection pooling code
Hashtable parms = new Hashtable();
parms.put
( Context.INITIAL_CONTEXT_FACTORY
, "com.ibm.ejs.ns.jndi.CNInitialContextFactory"
);
InitialContext ctx = new InitialContext(parms);
javax.sql.DataSource ds =(DataSource)ctx.lookup("jdbc/myDS");
java.sql.Connection conn = ds.getConnection();
```



Plug-in

- WAS V5 requires use of HTTP transport handler
- Plug-Ins available for IBM HTTP Server 5.3 on z/OS and into web servers on distributed platform
- On z/OS add following directives:
 - `ServerInit /webserver_install_root/bin/hs390WAS50Plugin_http.so:init_exit fully_qualified_path`
 - `ServerTerm /webserver_install_root/bin/hs390WAS50Plugin_http.so:term_exit`
- For each webapp add a services directive:
 - `Service /webapp_contextroot/* /webserver_install_root/bin/hs390WAS50Plugin_http.so:service_exit`
- Deploy application into WAS V5
- Run **install_root/AppServer/bin/GenPluginCfg.sh** to create plugin-cfg.xml file
- plugin-cfg.xml is EBCDIC encoded
- Manually edit plugin-cfg.xml if necessary
 - Change localhost to with defined hostname
 - Include portnumbers where HTTP server is listening for incoming requests to the VirtualHost in the VirtualHostGroup
- In case of initialization problems check **.../AppServer/logs/native.log.x.y**
 - X ...date
 - Y ... Sequence number



Migration tools

- AAT
 - ▶ Assemble JSP/servlet components developed by third party tools and/or **javac** to produce EAR files
 - ▶ Flexibility in assembling applications from various sources: WAR,RAR,EJB JARs and JARs
 - ▶ Available options:
 - Import existing module (JAR, RAR or WAR files)
 - Creating new module while creating new applications
 - Copying code artifacts, such as servlets, from one module to another
 - ▶ Download AAT from **.../AppServer/lib/setup.exe**
- WSAD V5
 - Includes complete development environment for transition to J2EE 1.3 applications

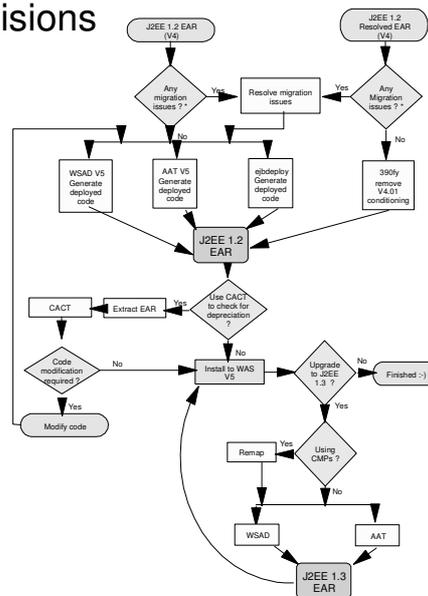


Application migration from WAS V4 - Overview

- Finding the existing EAR file
- Resolving any migration issues
- Generating a new (J2EE 1.2) ear
- Installing to WAS V5
- Optionally migrating ear to J2EE 1.3 (recommended)
- Re-install to WAS V5



Migration decisions





Migration Overview

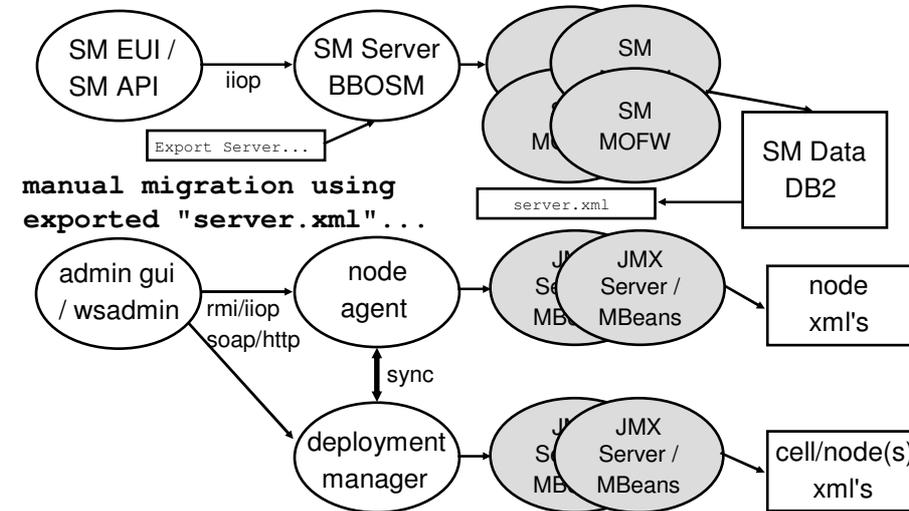
- Migration tools
- Configuration migration
- Application migration
- Installation script migration



Migration Tools

WSAD	Comprehensive development environment
AAT	Create/edit applications modules, verify archive files generate deployment code, alternative to WSAD
EJBDEPLOY	Commandline tool to generate deploy code
390fy	WAS V4 tools, migration option removes V4 conditioning to produce ear file to be deployed in WAS V5
CACT	Class API Checker Tool analyzes compiled JAVA servlets and EJBs and provides information on deprecated code

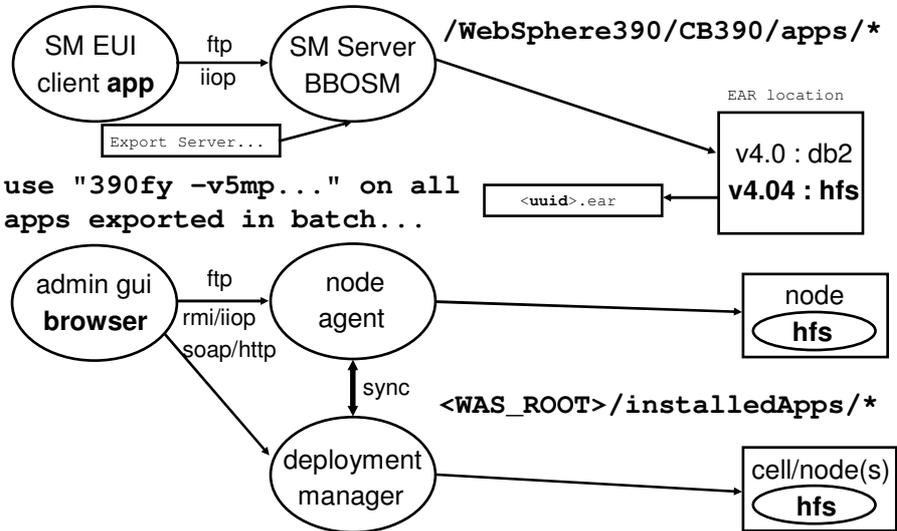
Configuration migration



Configuration migration

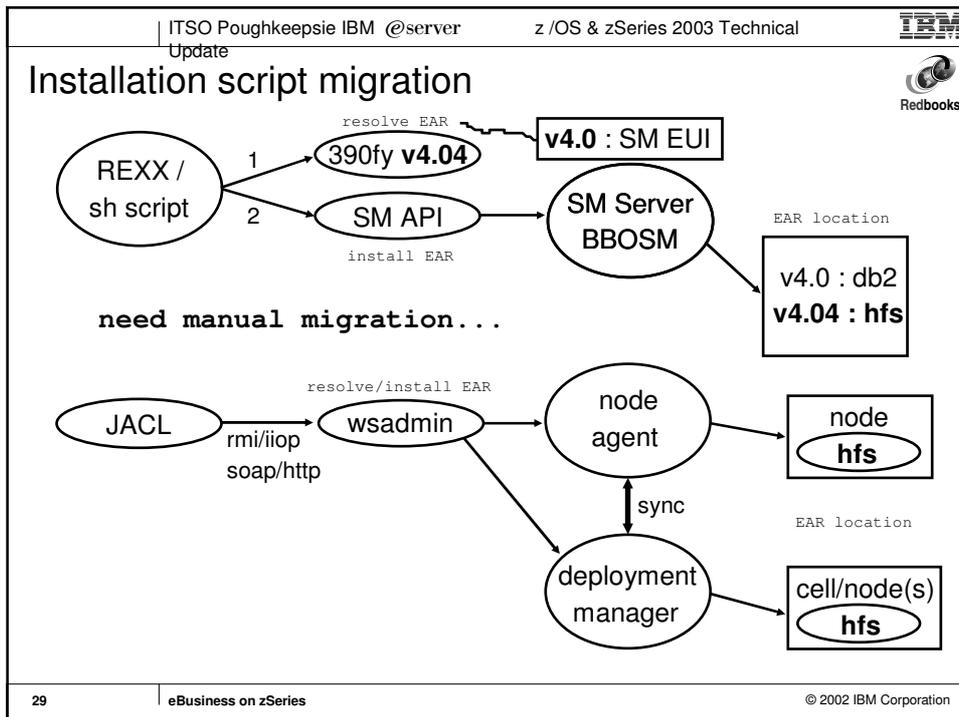
- v4.0 configuration setup
 - ▶ SMEUI / SMAPI is used to change the current config settings
 - ▶ SM Server processes the request via MOFW (IBM specific - managed object framework CORBA) objects
 - ▶ SM configuration data is stored in DB2 for all servers in various SM managed DB2 tables
- v4.0 => v5.0 configuration **migration** [manual]
 - ▶ use "Export server..." task to specify <export_dir> within HFS
 - ▶ use <export_dir>/<server_name>/server.xml & transfer config
- v5.0 configuration setup
 - ▶ admin gui / wsadmin used to change the current config settings
 - ▶ each server process contains "node agents" (aka MBean agent/server) which processes config change requests
 - ▶ SM configuration data is stored in HFS via various xml files
 /WebSphere/V5R0M0/AppServer/config/cells/SY1/...*.xml

Application migration



Application migration

- v4.0 application installation
 - ▶ SM EUI and/or SM API is used to deploy applications
 - ▶ SM Server processes the **resolved** ear file and installs it into:
 - DB2 for certain application information + [v4.0] EAR file as BLOB
 - [v4.04] HFS as an EAR file [/WebSphere390/CB390/apps/*]
- v4.0 => v5.0 application **migration**
 - ▶ use "Export server..." task to specify <export_dir> within HFS
 - ▶ [optional] change the <UUID>.ear name format to regular name
 - ▶ run "**390fy -v5mp <export_dir>/<server_name>**" to migrate all
- v5.0 application installation
 - ▶ admin gui / wsadmin can be used to deploy each migrated ear [will need to provide server specific info during install for v5.0]
 - ▶ appropriate deployment process ("node agent" and/or "dmgr") will take the migrated (**resolved**) ear file and install it into:
 - HFS as a directory that matches the EAR file name (ear unpacked)
 - .../AppServer/config/cells/SY1/applications/*.ear



- ITSO Poughkeepsie IBM @server z/OS & zSeries 2003 Technical Update
- ## Installation script migration
- v4.0 application installation
 - ▶ REXX exec written to expose SM API commands to deploy application and/or change system configurations
 - ▶ REXX exec can both resolve + install your applications all in one:
 - [v4.0 : NO]** EAR must first be resolved using SMEUI
 - [v4.04 : YES]** EAR can be applied **390fy** for resolve & installed via SMAPI
 - v4.0 => v5.0 application **migration** [manual]
 - ▶ No way to migrate scripts
 - [Note:** even on WAS Distributed - need to throw away wscp scripts]
 - ▶ Learn how to write JACL script and use wsadmin commands
 - v5.0 application installation
 - ▶ JACL script along with wsadmin can be used to deploy each migrated ear file
 - ▶ JACL script written for WAS Distributed (even wsadmin with different host/port on Windows can be used) can be used on zOS
- 30 | eBusiness on zSeries | © 2002 IBM Corporation

Migrating a J2EE 1.2 ear file to 1.3 using WSAD

- May save time that would otherwise be spent reworking CMPs
- Migration procedure
 - Import the 1.2 ear file into WSAD
 - On Enterprise application perspective select Migrate -> J2EE migration wizard
 - Explicitly request that CMPs are migrated and local client views are generated
 - Once migration wizard has completed check and correct compiler errors
 - EJB QL statements in the ejb.xml may have to be rewritten due to the EJB 2.0 specs
 - Fig. 7-12
- Check section Chapter 8. Migrating from EJB 1.0 to EJB1.1or toEJB2.0 of the migrate.pdf in the WSAD filesystem readme folder

JDBC/SQLJ

- db2sqljdbc.properties
 - Required by JDBC driver
 - Location defined in DB2SQLJPROPERTIES
 - Set at node level – related to specific DB2 subsystem !
- Loading external modules
 - WAS V4: specify location of DB2 DLLs in LIBPATH of server region
 - WAS V5: loading of external modules done by runtime classloader defined in Native Library Path during definition of resource
- Accessing DB2 Version 6 data
 - WAS V5 supports connection to DB2 V7
 - Use DDF (distributed data facility) of DB2 to access DB2 V6 data
 - See IBM Techdoc WP100217: WAS/390 V4 DB2 V7 Conundrum

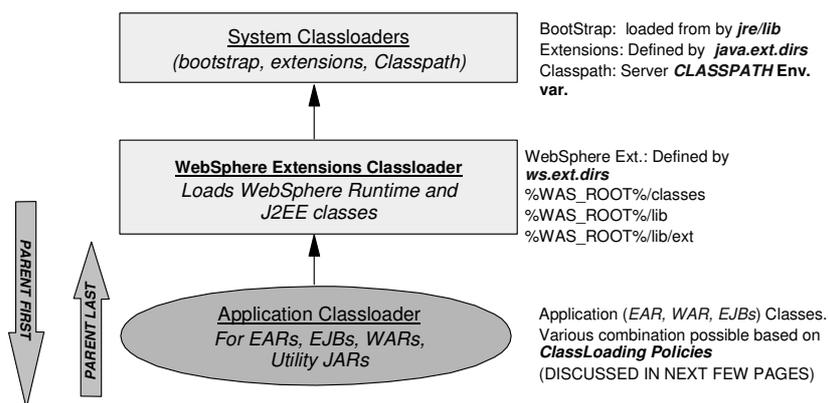


Classloader

- Classloaders hierarchy in WebSphere v5
- Application classloader policy
- WAR classloader policy
- Classloader Mode
- Application Classloader Policy – SINGLE
- Application Classloader Policy – MULTIPLE
- v4 vs. v5 Classloading Policies – Comparison
- Reloading Classes in Version 5
- Application Specific Libraries



Classloaders Hierarchy in WebSphere v5



Note: WAS 5.0 default of PARENT_FIRST is the opposite of most WAS v4.0x behavior, except for v4.0x J2EE Application Mode which did default to PARENT_FIRST



Application Classloader policy

- Determines how the applications share classloader
- At the Application Server level - choose SINGLE or MULTIPLE
 - SINGLE means the EJB, RAR modules and dependent JARs for all the EARs are loaded by one classloader called the Application classloader
 - MULTIPLE means the EJB, RAR modules and dependent JARs for each EAR are loaded by its own classloader
 - Whether the WAR is loaded by this Application classloader is dictated by the WAR classloader policy



WAR Classloader policy

- Determines how the WAR modules are loaded per Application
- At the Enterprise Application level, choose APPLICATION or MODULE
 - APPLICATION: all the Web modules in the application EAR use the Application classloader (dictated by the Application Classloader Policy)
 - MODULE: every WAR uses its own classloader, different than the Application Classloader
 - Selection can be made at application install time



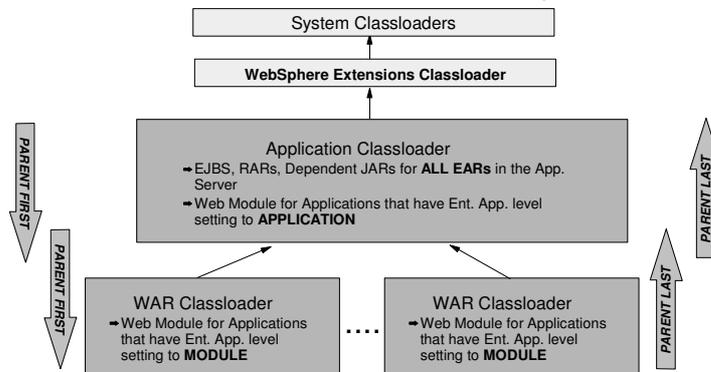
Classloader Mode

- Set for Application Classloader policy and WAR Classloader policy
- PARENT_FIRST - default
 - Search the immediate parent first and then its policy would determine if that got search first or its parent
- PARENT_LAST
 - Tries to find and load the class from its own classloader and if the class was not found, it delegates to its immediate parent classloader and then the immediate parent's classloader policy would take control



Application Classloader Policy - SINGLE

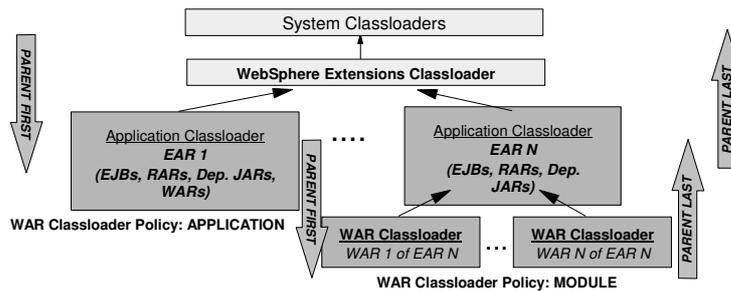
- Application Classloader policy of **SINGLE** and WAR classloader policy of **APPLICATION** or **MODULE**
 - ▶ Pros: Each Application EJBs, RARs, Dep. JARs can reference other classes in other Applications
 - ▶ Cons: Cannot start and stop individual Applications without affecting others





Application Classloader Policy - MULTIPLE

- Application Classloader policy of **MULTIPLE**
- WAR classloader policy of **APPLICATION** or **MODULE**
- Pros:
 - ▶ Can restart each application without affecting others
 - ▶ Classes within each EAR can reference all the classes with the same EAR even if in different modules of the EAR (classes in WAR may or maynot be referenced, depending on Ent. Application level policy)
- Cons: Classes within one EAR cannot reference classes in another EAR



v4 vs. v5 Classloading Policies - Comparison

	v4	v5
	4.0 Classloader Policy	v5 Application Classloader Policy
	v5 WAR Classloader Policy	
SERVER	SINGLE	APPLICATION
COMPATIBILITY	SINGLE	MODULE
APPLICATION	MULTIPLE	APPLICATION
MODULE	N/A	N/A
J2EE Application Mode	MULTIPLE	MODULE



Reloading Classes in Version 5

- Web Modules can be reloaded independently
 - Stop/Restart the Web Module
 - Also, can configure Web Modules to restart automatically if contents changes
- EJB Modules
 - Cannot be stopped/restarted independently in Version 5
 - The entire application has to be restarted
- Best practice: stop/restart the Enterprise App
 - This will ensure that all the dependencies are also reloaded



Application Specific Libraries

- Application Specific Libraries can be defined
 - For libraries of code that needs to be shared across multiple applications within a server
 - Support for java and native libraries.
 - Environment ->Shared Libraries -> New
- On the Enterprise Application
 - Specify the shared libraries to be used
 - Enterprise Applications -> Application -> Libraries -> Add (add one of the defined Shared Libraries)



Client Container

- Overview
- Application Client - Client Container Interaction
- LaunchClient



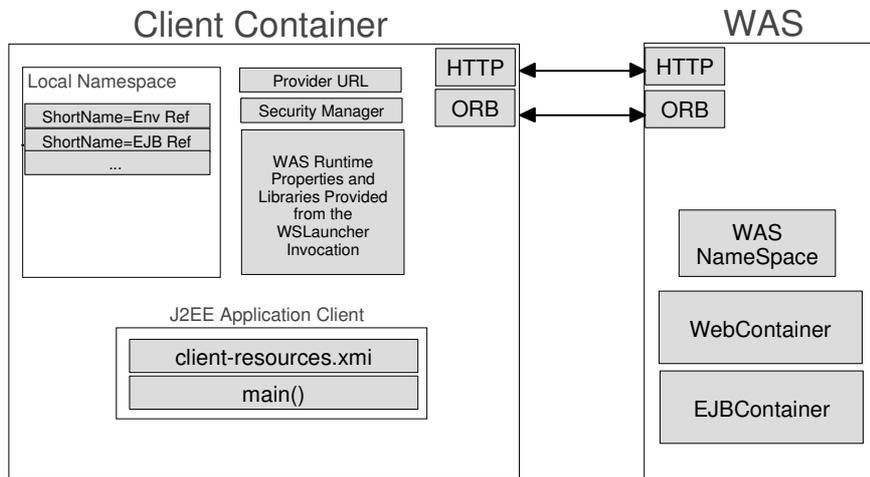
Client Container - Overview

- The Client Container is a runtime environment that provides a specific set of services and resources to J2EE Application Clients.
- J2EE application client can be built with WSAD
- Resources include:
 - ▶ JDBC Databases
 - ▶ URL's
 - ▶ JMS Message Queues
 - ▶ Java Mail
 - ▶ Environment Entries (Native Types)
 - ▶ EJB's
- Services include:
 - ▶ Security
 - ▶ Naming
 - ▶ Communications Protocols (RMI/IIOP, HTTP, etc.)

Note: J2EE Specification allows for the option of Application Clients to directly participate in Global Transactions. For WebSphere, Application Clients cannot directly participate in RRS Transactions

Client Container

Application Client - Client Container Interaction



LaunchClient

- Validate command-line arguments
- Create Application Client Meta Data
 - ▶ init the MOF/WCCM runtime environment
 - ▶ Determine the main class of the application client
 - ▶ MOF/WCCM parses the client-resources.xml file for resource/ejb bindings
- initialize the Name Space
 - ▶ Namespace is of type com.ibm.ws.naming.java.javanamespace
- Add Environment Entries
- Lookup and bind
 - ▶ EJB's
 - ▶ Resources
 - ▶ Environment References
- Add the ORB
- Find and Invoke application client's main class



Development environment migration

- IBM VisualAge for Java Enterprise Edition
 - Common Connector Framework (CCF)
 - Enterprise Access Builder (CCF)
- J2C standards evolved
 - First shipped as beta with VA Java and WSADIE V4
- WebSphere Studio Integration Edition V5
 - Fully supports J2C



Summary: Migration Issue

Servlet/JSPs

EJBs

CMPs

Resources, DB2, JMS, IMS and CICS

Connectors

ASCII/EBCDIC

JDBC/SQLJ

Transactions

Deployment Descriptors

Plugin

Packaging/Assembling

Security

Constructing Enterprise Applications

Migration Tools

JMS/WMQ