WBSR85

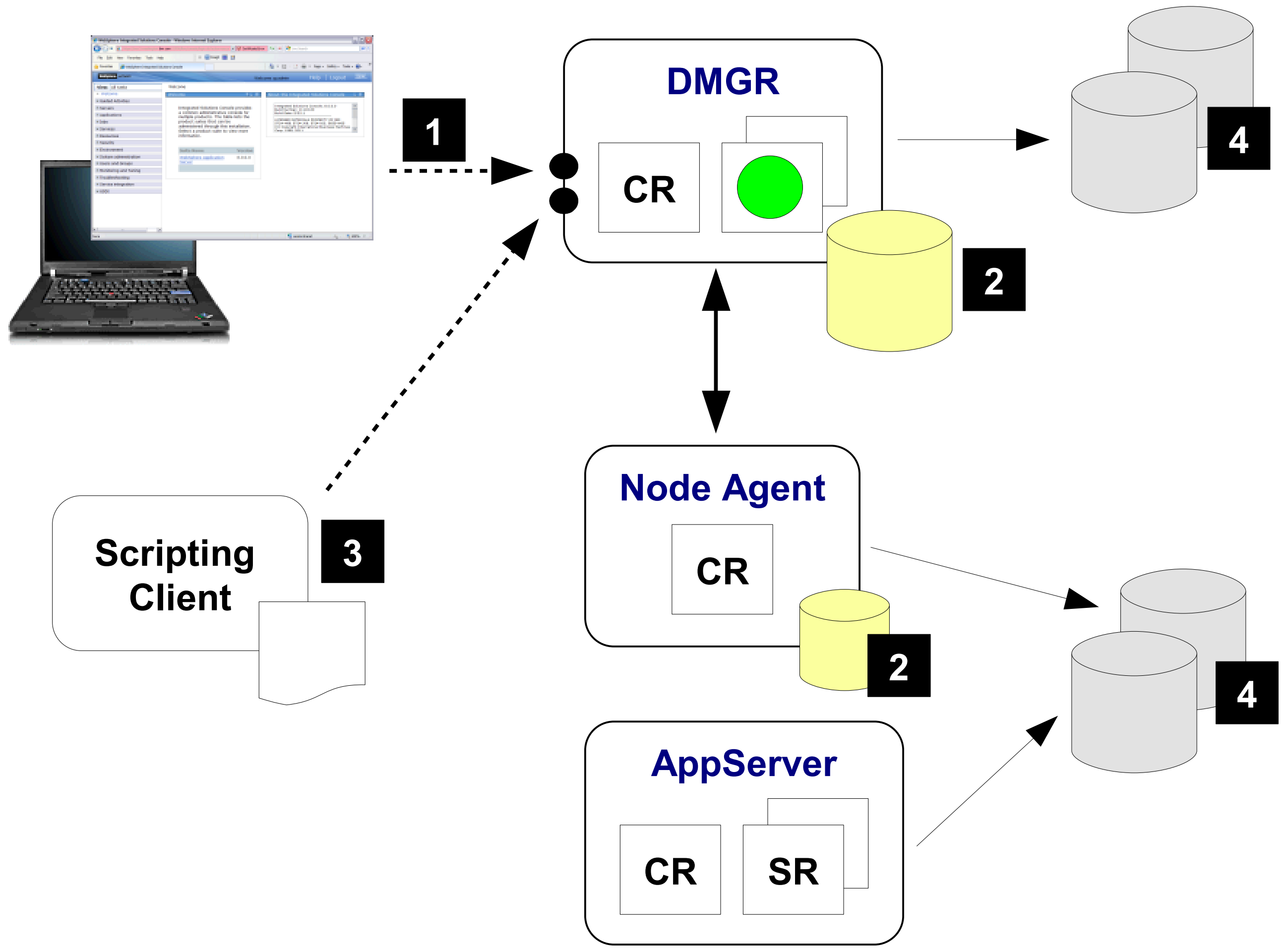
WebSphere Application Server z/OS V8.5

Unit 2 - Administration Model

This page intentionally left blank

High-Level Conceptual Picture

This provides the framework of our focus areas this unit:



1 **Administrative Console**

2 **Configuration File Systems for each Node**

3 **WSADMIN scripting interface**

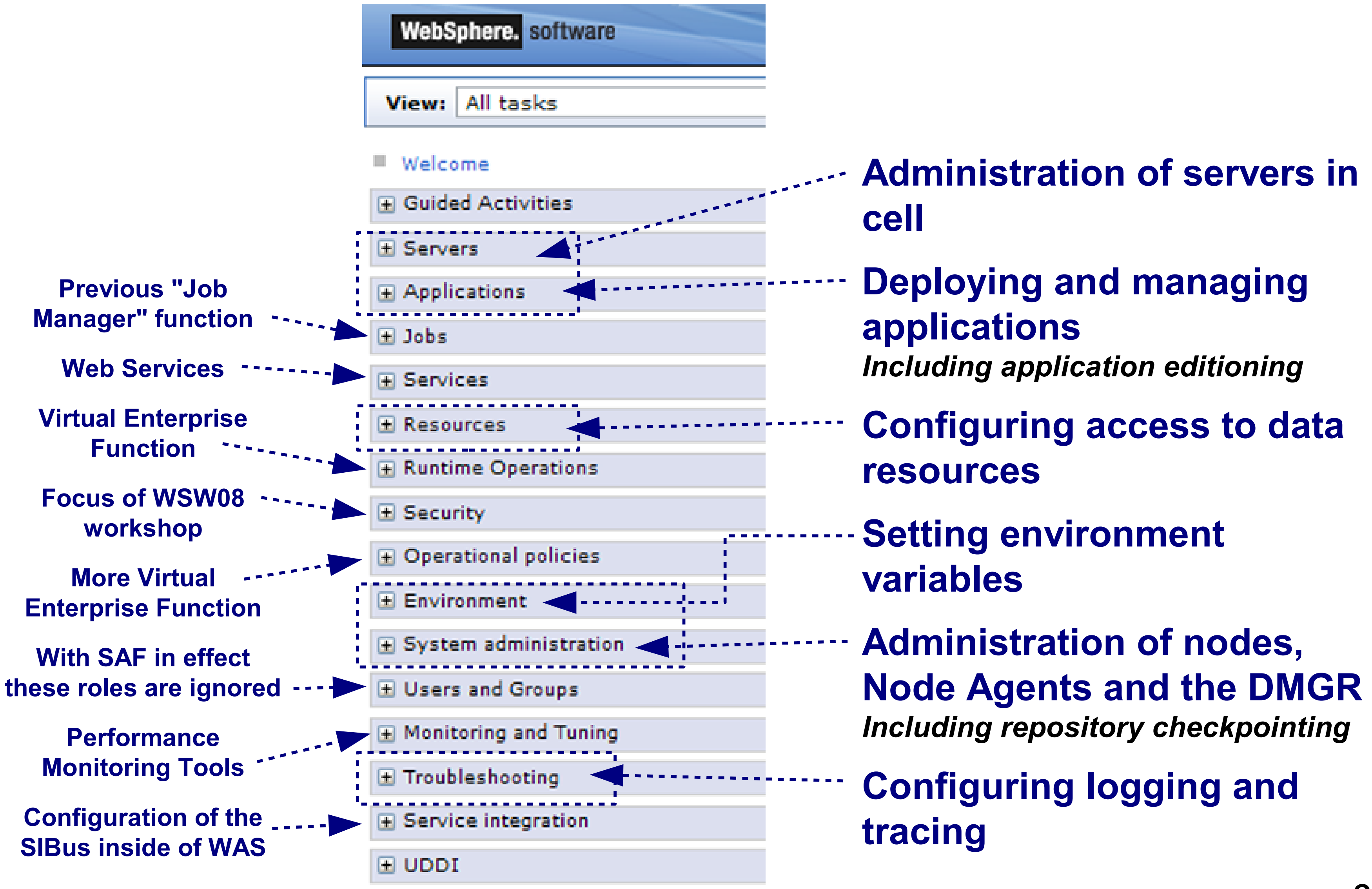
4 **Logging and tracing and HPEL**

These are the topics we'll cover in this unit

Admin Console

Left-Side Navigator Bar for V8.5 Admin Console

At this level the Admin Console is common across all platforms. The areas of focus for us in this workshop are indicated below:


WebSphere. software

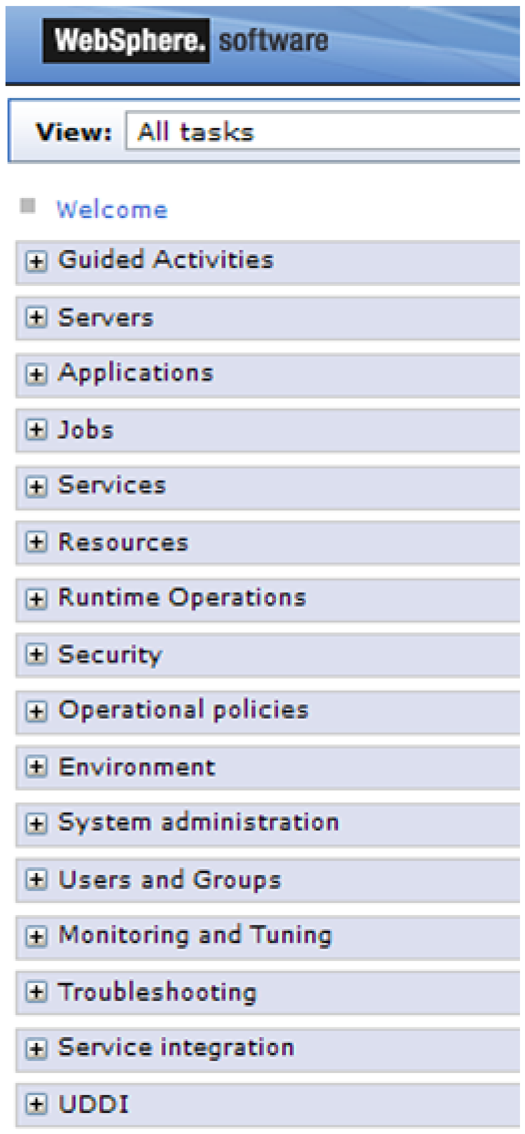
View: All tasks

- Welcome
- Guided Activities
- Servers — Administration of servers in cell
- Applications — Deploying and managing applications *Including application editioning*
- Jobs — Previous "Job Manager" function
- Services — Web Services
- Resources — Configuring access to data resources
- Runtime Operations — Virtual Enterprise Function
- Security — Focus of WSW08 workshop
- Operational policies — More Virtual Enterprise Function
- Environment — Setting environment variables
- System administration — Administration of nodes, Node Agents and the DMGR *Including repository checkpointing*
- Users and Groups — With SAF in effect these roles are ignored
- Monitoring and Tuning — Performance Monitoring Tools
- Troubleshooting — Configuring logging and tracing
- Service integration — Configuration of the SIBus inside of WAS
- UDDI

Commonality …

Degree of Commonality Across Platforms

Is actually fairly high …



🟩🟩🟩🟩🟨

Servers and Clustering
Very common until you get to things like server short names and the Multi-JVM model

🟩🟩🟩🟩🟩

Applications
Identical across platforms

🟩🟩🟩🟩🟩

Web Services
Identical across platforms

🟩🟩🟩🟩🟨

Security
Fairly common with the exception of definitions to allow use of SAF

🟩🟩🟩🟩🟩

Environment
Interface is identical. Some of the variables you enter may be z/OS-specific.

🟩🟩🟩🟩🟨

Resources
Very common until you get to the definition of local adapters and native libraries

🟩🟩🟩🟩🟨

System Administration
Very common with the exception of the "z/OS location server" under "Node Groups"

🟩🟩🟩🟩🟩

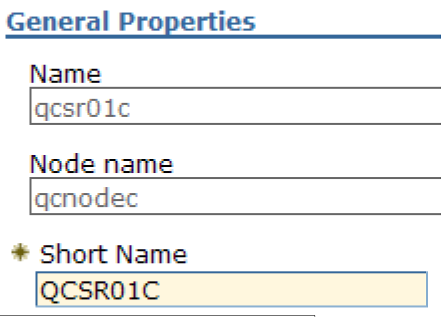
SIBus
Identical across platforms

Platform specifics …

Examples of Platform Specifics Surfacing

A brief sampling of where some z/OS platform specifics surface in the Administration Console:

Under a given application server:

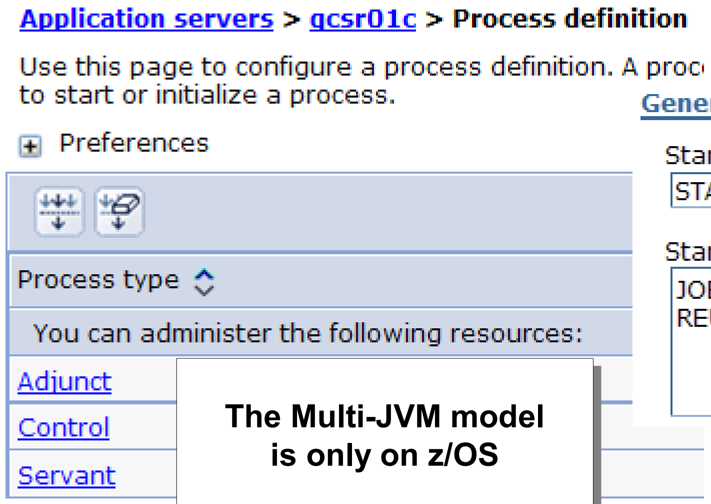

General Properties

Name
qcsr01c

Node name
qcnodec

Short Name
QCSR01C

Short names are exclusive to z/OS


Application servers > qcsr01c > Process definition

Use this page to configure a process definition. A proc
to start or initialize a process.

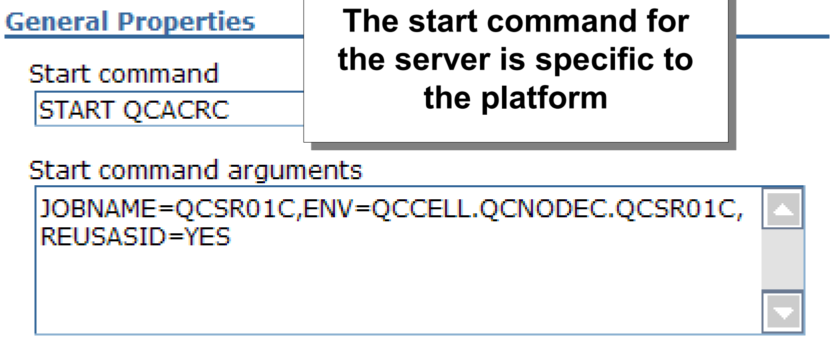
Preferences

Process type

You can administer the following resources:

Adjunct
Control
Servant

The Multi-JVM model is only on z/OS

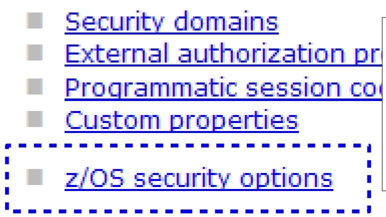

General Properties

Start command
START QCACRC

Start command arguments
JOBNAME=QCSR01C,ENV=QCCELL.QCNODEC.QCSR01C,
REUSASID=YES

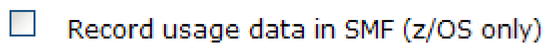
The start command for the server is specific to the platform

Under Global Security:


- Security domains
- External authorization pr
- Programmatic session co
- Custom properties
- z/OS security options

A section on z/OS-specific security settings and properties

Under the integrated Java Batch configuration:


☐ Record usage data in SMF (z/OS only)

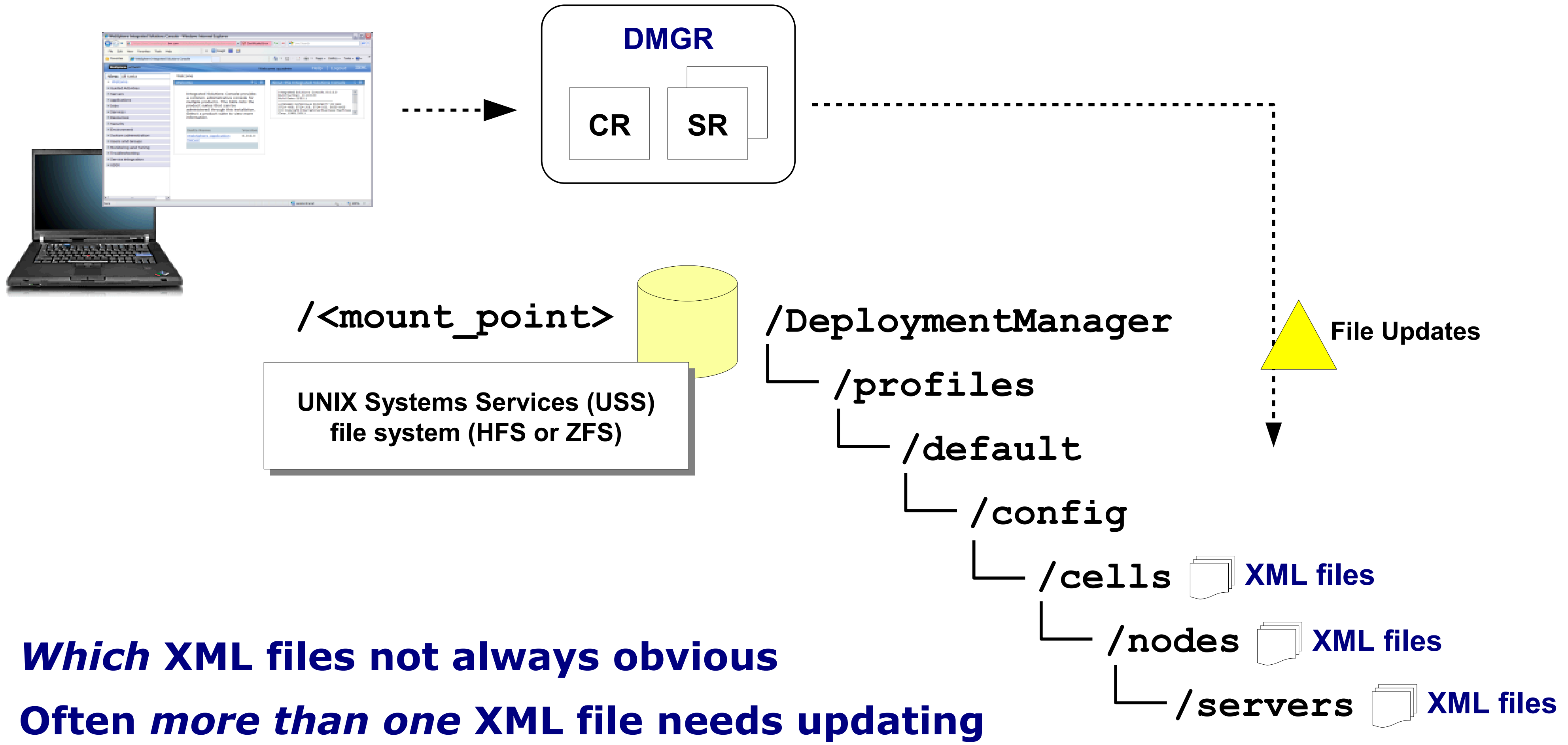
Exploit z/OS SMF if you wish

Point here is that while the Admin Console has a great deal of commonality, you can find differences the closer to the platform you get

Updating XML …

Administrative Application - Smart XML Updater

It does other things, but a large portion of the Administrative Application's function is to know how to translate mouse clicks into XML updates:



Which **XML files not always obvious**

Often *more than one* XML file needs updating

You could try to do this yourself ...

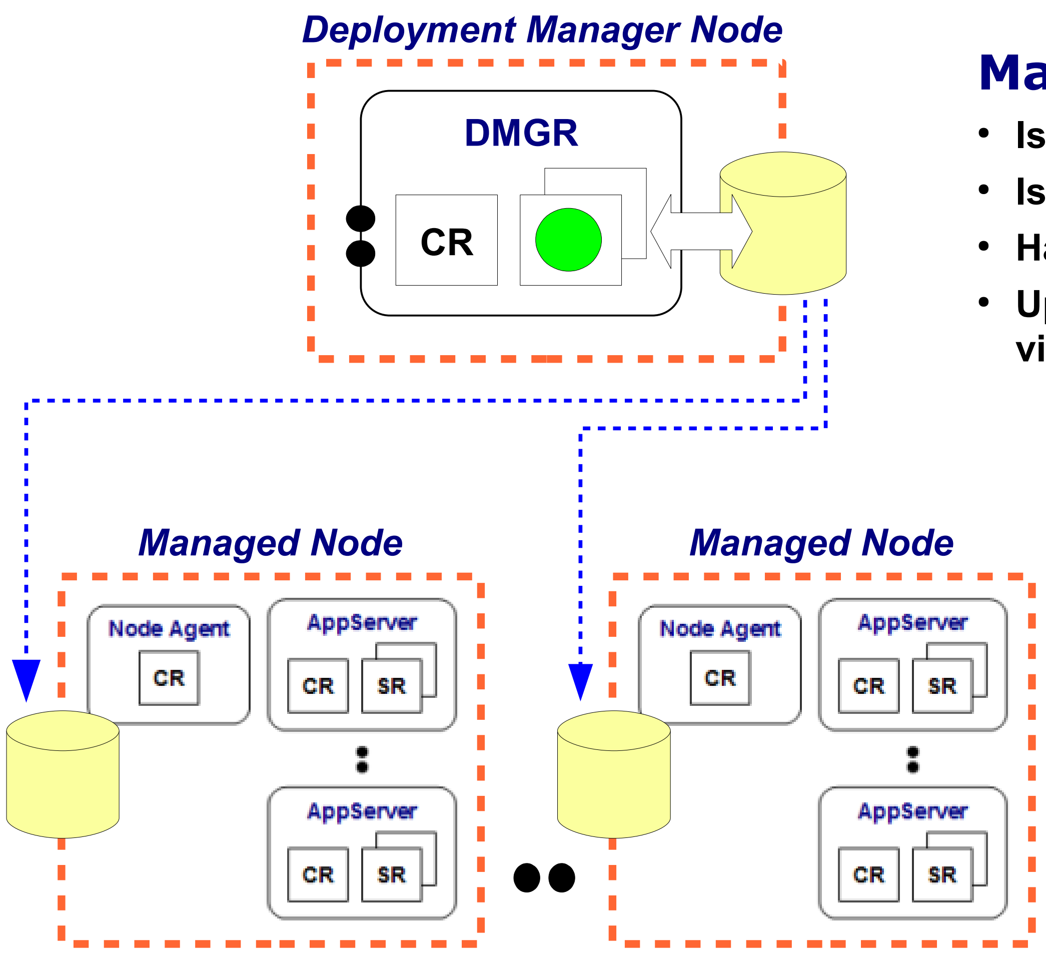
Better to allow administrative function to do it

Config file system ...

Configuration File System

Each Node Has a Configuration Structure

In a Network Deployment (ND) configuration, each node has its own configuration file structure. DMGR owns the "master" and nodes are subordinate to that:



Master Configuration

- Is maintained in a USS file system (HFS or ZFS)
- Is updated by the Administrative Application
- Has all the information about the whole cell
- Updates to master are propagated to each node via act of synchronization

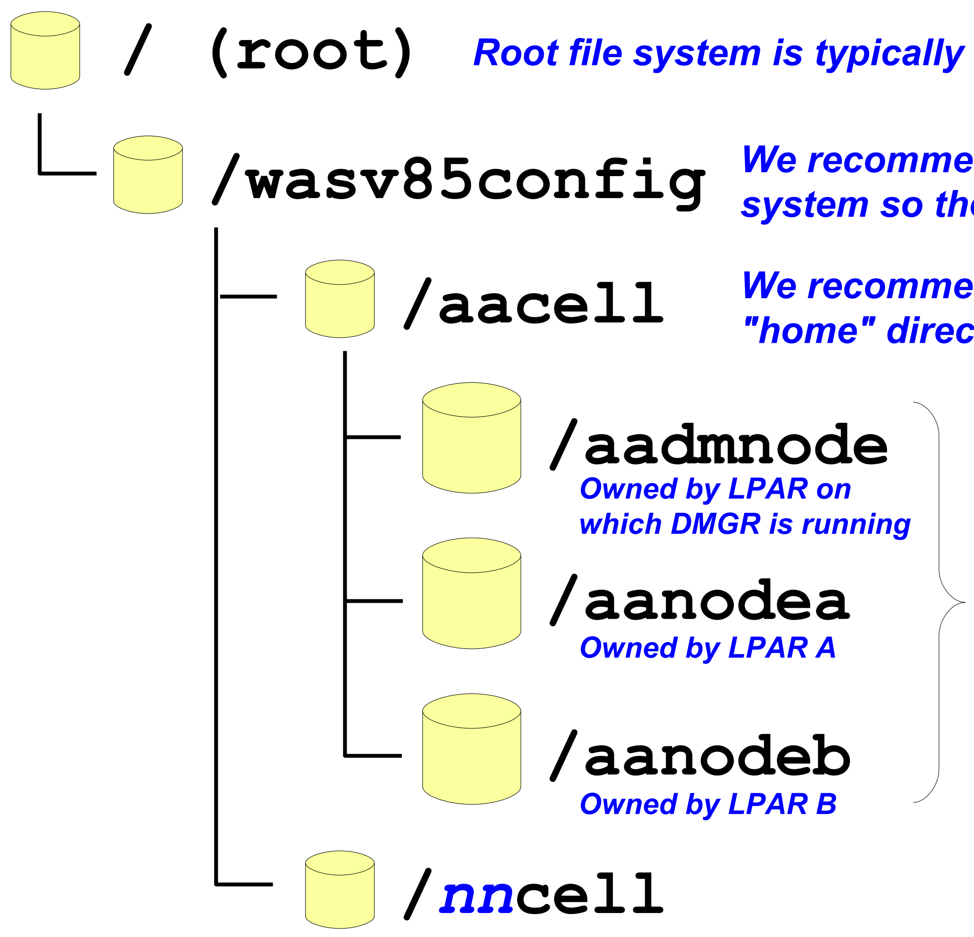
Node Configuration

- Is maintained in a USS file system (HFS or ZFS)
- Can be in same file system as DMGR, but we recommend separate for each node
- Is updated by the Node Agent during synchronization
- Has all the information its node
- Has *some* information about other nodes

Common file system layout …

How We Generally See the File Systems Deployed

Here's a picture that shows how the file system would be created and mounted based on the jobs generated using the PRS4686* planning spreadsheets:



Root file system is typically relatively small

We recommend a version directory with it's own relatively small file system so the root is not exposed to writes and out of space issues

We recommend a cell directory file system for each cell so the ID "home" directories can be built here and house any Java dumps

These are the file systems created by the generated jobs

Config and cell are manually created and mounted by you

They can be quite large depending on the number and size of applications deployed

V8.5 spreadsheets

TechDocs PRS4944

DMGR file system …

The Essential Structure of DMGR Configuration Tree

Here's a snapshot of some of the key elements of the configuration structure:

```
/wasv85config/aacell/aadmnode/DeploymentManager
├─ /bin ◄
├─ /installableApps
├─ /java
├─ /java64
├─ /lib ◄
└─ /profiles
   └─ /default
      ├─ /bin
      ├─ /config
      │  └─ /cells
      │     └─ /<cell_long_name>
      │        │  XML Files
      │        └─ /nodes
      │           └─ /<node_long_name>
      │              │  XML Files
      │              └─ /servers
      │                 └─ /<server_long_name>
      │                       XML Files
      ├─ /logs
      ├─ /properties
      └─ /wstemp
```

Deployment Manager's mount point and root

Java symlinked from here

Shell scripts, JAR files and shared object modules

Other platforms use "profiles" to allow multiple cell configurations in the same install root. But on z/OS we separate configuration from install so one profile is sufficient

This is the heart of it

All the nodes and all the servers and their properties for the whole cell represented here

Identical to distributed in basic structure; different in certain ways we'll show

`logs` -- where things like WSADMIN logging goes
`properties` -- key properties files for functions and activities
`wstemp` -- temporary directory that you can clean periodically

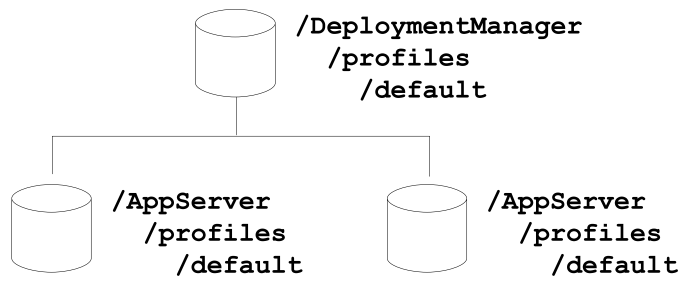
Node file system …

Each Managed Node Has Similar Configuration

The key difference is in that it maintains only partial awareness of *other* nodes but *full awareness of itself:*

```
/wasv85config/aacell/aanodea/AppServer
├─ /bin
├─ /installableApps
├─ /java
├─ /java64
├─ /lib
└─ /profiles
    └─ /default
        ├─ /bin
        ├─ /config
        │   └─ /cells
        │       └─ /<cell_long_name>
        │           │  XML Files
        │           └─ /nodes
        │               └─ /<node_long_name>
        │                   │  XML Files
        │                   └─ /servers
        │                       └─ /<server_long_name>
        │                              XML Files
        ├─ /logs
        ├─ /properties
        └─ /wstemp
```

The mount point and the node root are different because this is a different node from the Deployment Manager node



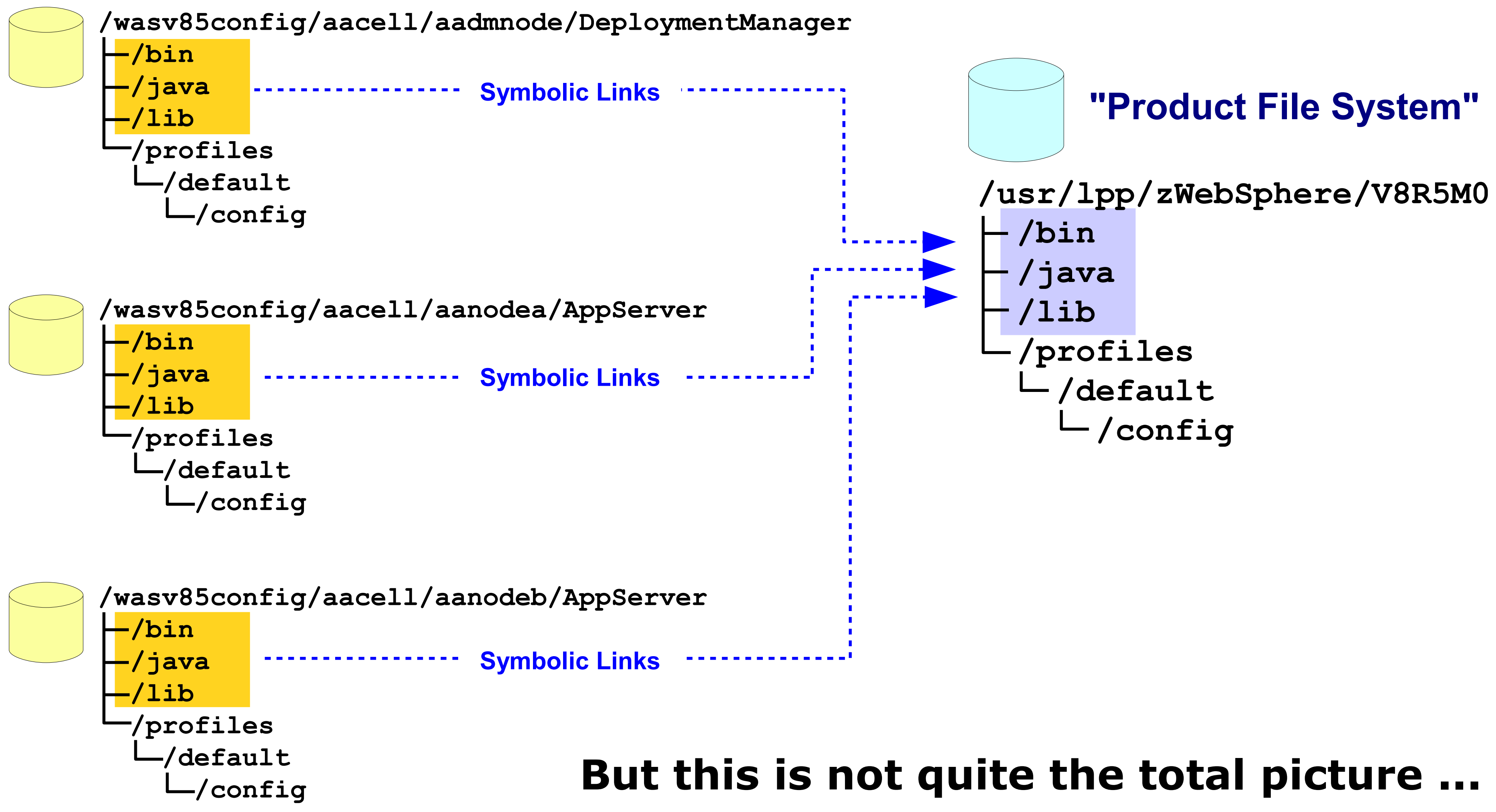
It will have a directory for each node. Other nodes have some meta-data files that provides the partial information.

But the node directory for itself and the servers under it are fully populated with detailed XML

Relationship to install image …

Relationship to "Install Image"

The "install image" is the file system that contains the product binaries. The configuration file systems link to that via symbolic links:

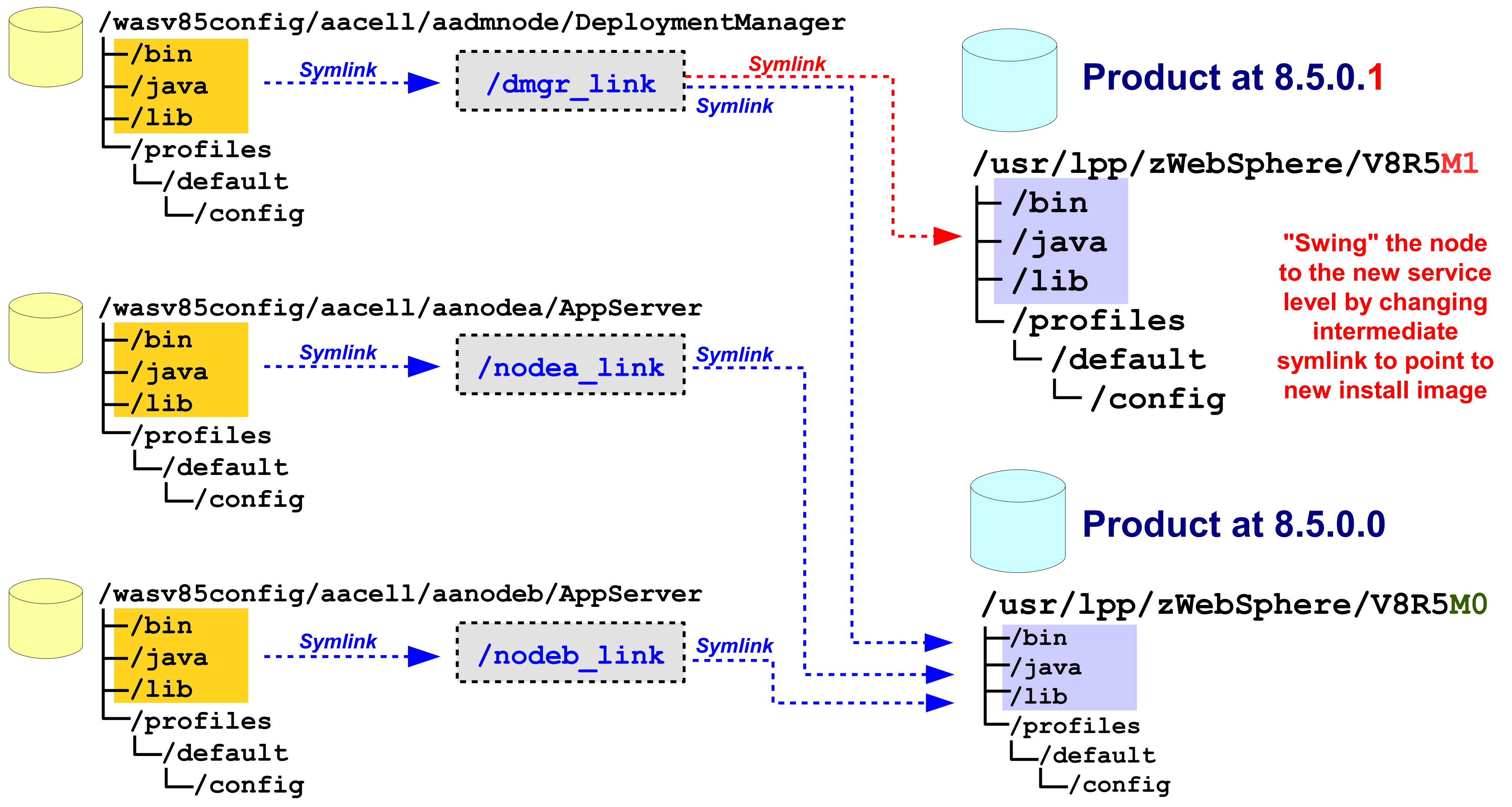


But this is not quite the total picture ...

Intermediate symlinks ...

Intermediate Symbolic Links

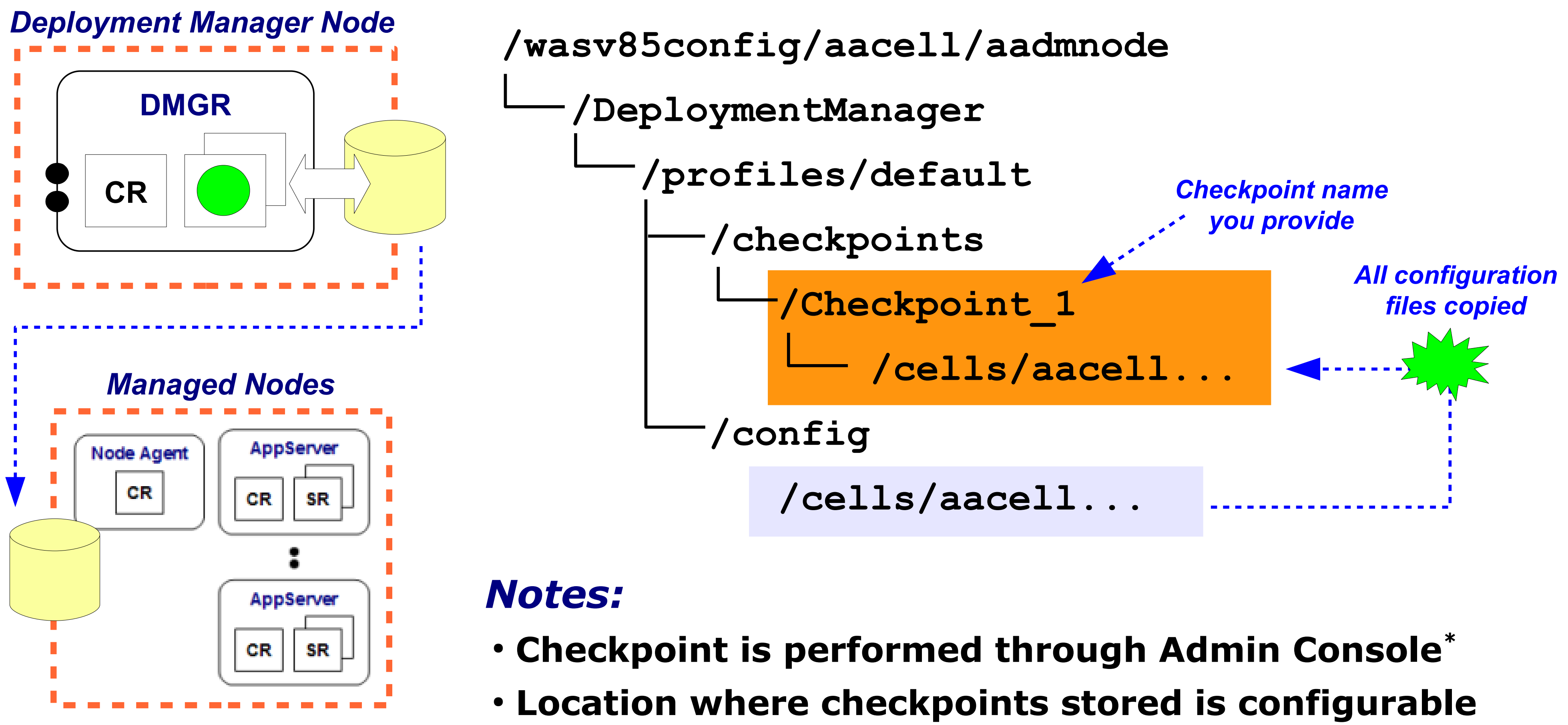
We've taken this symlink structure one step further by introducing an "intermediate symbolic link" for each node between the node and the install image:



Repository Checkpoints …

Repository Checkpoints in Concept

It's fairly simple ... the Admin function now provides a way to "take a snapshot" of the master configuration and restore back to previous snapshots if you wish:



Deployment Manager Node
DMGR
CR
Managed Nodes
Node Agent
CR
AppServer
CR
SR
AppServer
CR
SR
Checkpoint name you provide
All configuration files copied

```
/wasv85config/aacell/aadmnode
└──/DeploymentManager
    └──/profiles/default
        ├──/checkpoints
        │   └─/Checkpoint_1
        │      └── /cells/aacell...
        └──/config
             /cells/aacell...
```

Notes:

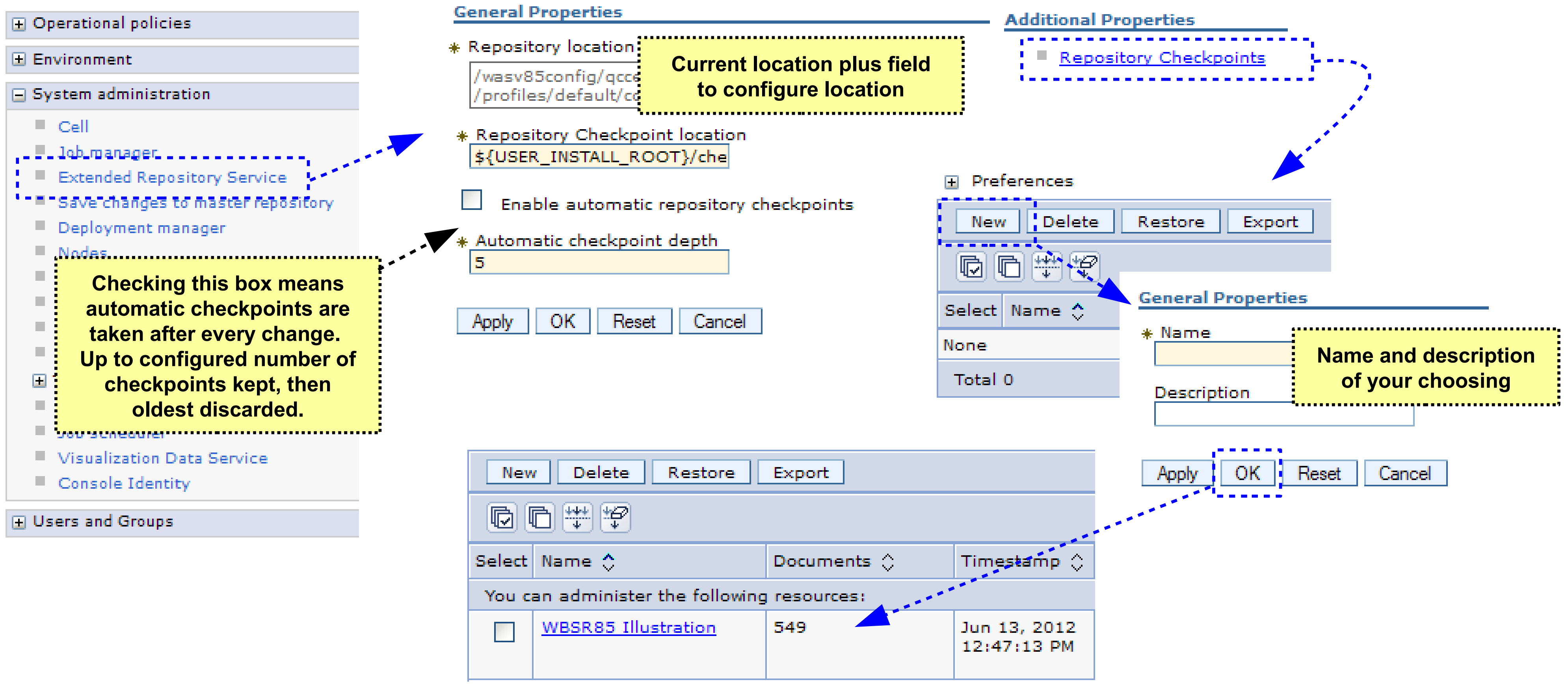
- **Checkpoint is performed through Admin Console***
- **Location where checkpoints stored is configurable**
- **Multiple checkpoints possible**
- **Restore selected checkpoint through Admin Console***

* or WSADMIN

Taking a repository checkpoint …

Admin Console for Repository Checkpoints

Some bitmap captures that illustrate the process of taking a checkpoint backup:



⊞ Operational policies
⊞ Environment
⊟ System administration
- Cell
- Job manager
- Extended Repository Service
- Save changes to master repository
- Deployment manager
- Nodes
- Visualization Data Service
- Console Identity

⊞ Users and Groups

Checking this box means automatic checkpoints are taken after every change. Up to configured number of checkpoints kept, then oldest discarded.

General Properties

* Repository location
/wasv85config/qcc
/profiles/default/c

Current location plus field to configure location

* Repository Checkpoint location
${USER_INSTALL_ROOT}/che

☐ Enable automatic repository checkpoints

* Automatic checkpoint depth
5

Additional Properties
- Repository Checkpoints

⊞ Preferences

Select	Name
None	
Total 0	

General Properties

* Name

Description

Name and description of your choosing

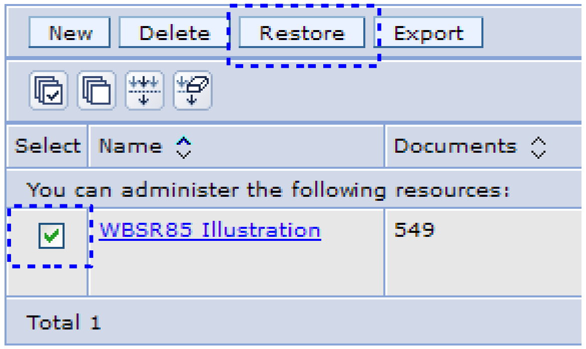
Select	Name	Documents	Timestamp
You can administer the following resources:			
☐	WBSR85 Illustration	549	Jun 13, 2012 12:47:13 PM

Checkpoints may then be "restored" to fall back to a previous configuration checkpoint. Configuration reverts to those settings.

Restoring a saved checkpoint …

Restoring Checkpoints

A couple of notes regarding this ...



Select	Name	Documents
You can administer the following resources:		
✓	WBSR85 Illustration	549
Total 1		

Restoring means the checkpoint configuration files and directories are copied back to the master configuration's `/config/cells` path

Updates to node configuration file systems through normal synchronization process

Process will synchronize with the nodes if auto-synchronize is set for the Node Agents. If not, remember to manually synchronize to the nodes.

You may need to log off the Admin Console and back on to see the restored configuration artifacts in the Admin Console display

Restore puts configuration files back in place, but it does *not* restart servers or applications that were deleted and then restored

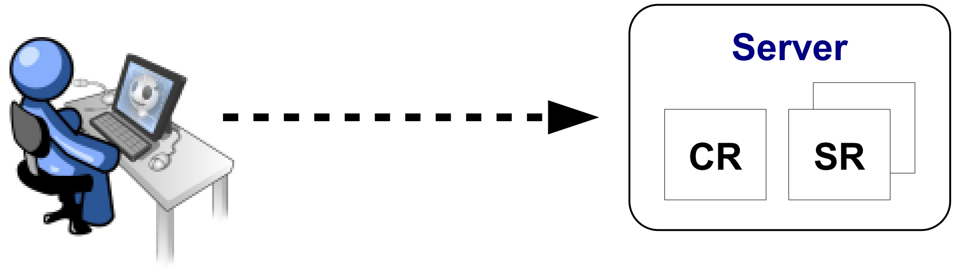
You should *still* have a solid backup/restore process in addition to this checkpoint function

MODIFY command ...

z/OS MODIFY

MODIFY Facility of z/OS Operating System

MODIFY is a means of dynamically displaying information about started task, or dynamically updating the runtime settings for that started task



```
F <jobname>,keyword,keyword...
```

```
F Z9SR01A,HELP
F Z9SR01A,HELP
BBOO0178I THE COMMAND MODIFY MAY BE FOLLOWE
BBOO0179I CANCEL - CANCEL THIS CONTROL REGI
BBOO0179I TRACEALL - SET OVERALL TRACE LEVE
 :
BBOO0179I DISPLAY - DISPLAY STATUS
 :
BBOO0179I WLM_MIN_MAX - RESET WLM MIN/MAX SERVANT SETTINGS
BBOO0179I RECLASSIFY - RE-PROCESS WLM CLASSIFICATION FILE
 :
BBOO0179I FAILOVER - FAILS OVER CONNECTIONS FOR RESOURCE IDENTIFIED BY GIVEN JNDINAME
BBOO0179I FAILBACK - FAILS BACK CONNECTIONS TO RESOURCE IDENTIFIED BY GIVEN JNDINAME
```

Example of output generated by simply specifying `HELP` on the `MODIFY`

35 `MODIFY` commands for WAS z/OS

18 `DISPLAY` options

InfoCenter `rxml_mvsmodify`

First 18 commands …

MODIFY Commands, Part 1

Here's the first 18 of 35 `MODIFY` commands available with WAS z/OS V8:

```
CANCEL - CANCEL THIS CONTROL REGION
TRACEALL - SET OVERALL TRACE LEVEL
TRACEBASIC - SET BASIC TRACE COMPONENTS
TRACEDETAIL - SET DETAILED TRACE COMPONENTS
TRACESPECIFIC - SET SPECIFIC TRACE POINTS
TRACEINIT - RESET TO INITIAL TRACE SETTINGS
TRACENONE - TURN OFF ALL TRACING
TRACETOSYSPRINT - SEND TRACE OUTPUT TO SYSPRINT (YES/NO)
DISPLAY - DISPLAY STATUS
TRACE_EXCLUDE_SPECIFIC - EXCLUDE SPECIFIC TRACE POINTS
JAVACORE - GENERATE JVM CORE DUMP
HEAPDUMP - GENERATE JVM HEAP DUMP
JAVATDUMP - GENERATE JVM TDUMP
TRACEJAVA - SET JAVA TRACE OPTIONS
TRACETOTRCFILE - SEND TRACE OUTPUT TO TRCFILE (YES/NO)
MDBSTATS - MDB DETAILED STATISTICS
PAUSELISTENERS - PAUSE THE COMMUNICATION LISTENERS
RESUMELISTENERS - RESUME THE COMMUNICATION LISTENERS
```

Specifying `,HELP` on many these will then display the parameters acceptable for that particular command

We'll focus on the DISPLAY command in a moment

InfoCenter `rxml_mvsmodify`

Next 17 commands …

MODIFY Commands, Part 2

Here's the final 17 of 35 `MODIFY` commands available with WAS z/OS V8:

```
STACKTRACE - LOG JAVA THREAD STACK TRACEBACKS
TIMEOUTDUMPACTION - SET TIMEOUT DUMP ACTION
TIMEOUTDUMPACTIONSESSION - SET TIMEOUT DUMP ACTION SESSION
TIMEOUT_DELAY - SET TIMEOUT DELAY VALUE
WLM_MIN_MAX - RESET WLM MIN/MAX SERVANT SETTINGS
SMF - SET SMF120 OPTIONS
DPM - DISPATCH PROGRESS MONITOR
RECLASSIFY - RE-PROCESS WLM CLASSIFICATION FILE
TRACERECORD - SET TRACE RECORD WRITE OPTIONS
MSGROUTE - SET ROUTING LOCATION OPTIONS
FORMFEED - ISSUE FORMFEED TO SYSOUT AND SYSPRINT
DISABLEFAILOVER - DISABLES FAILOVER SUPPORT FOR RESOURCE IDENTIFIED BY GIVEN JNDINAME
ENABLEFAILOVER - ENABLES FAILOVER SUPPORT FOR RESOURCE IDENTIFIED BY GIVEN JNDINAME
FAILOVER - FAILS OVER CONNECTIONS FOR RESOURCE IDENTIFIED BY GIVEN JNDINAME
FAILBACK - FAILS BACK CONNECTIONS TO RESOURCE IDENTIFIED BY GIVEN JNDINAME
SETOLATRACE - SET OLA TRACE LEVEL. SETOLATRACE=0..2, RGE | REGNAME | JOBNAME =x...x
SETOLATRACEPROPS - READ OLA TRACE PROPERTIES FILE
```

Specifying `,HELP` on many these will then display the parameters acceptable for that particular command

InfoCenter `rxml_mvsmodify`

DISPLAY ...

The DISPLAY Command

A particularly useful MODIFY command is DISPLAY, which has keywords that allow you to display specific information about the server:

```
F Z9SR01A,DISPLAY,HELP
BBOO0178I THE COMMAND DISPLAY, MAY BE FOLLOWED BY ONE OF THE FOLLOWING KEYWORDS:
BBOO0179I SERVERS - DISPLAY ACTIVE CONTROL PROCESSES
BBOO0179I SERVANTS - DISPLAY SERVANT PROCESSES OWNED BY THIS CONTROL PROCESS
BBOO0179I LISTENERS - DISPLAY LISTENERS
BBOO0179I CONNECTIONS - DISPLAY CONNECTION INFORMATION
BBOO0179I TRACE - DISPLAY INFORMATION ABOUT TRACE SETTINGS
BBOO0179I JVMHEAP - DISPLAY JVM HEAP STATISTICS
BBOO0179I WORK - DISPLAY WORK ELEMENTS
BBOO0179I ERRLOG - DISPLAY THE LAST 10 ENTRIES IN THE ERROR LOG
BBOO0179I MODE - DISPLAY THE EXECUTION BITMODE
BBOO0179I THREADS - DISPLAY THREAD STATUS
BBOO0179I ADAPTER - DISPLAY OLA ADAPTER STATUS
BBOO0179I OLATRACE - DISPLAY ADAPTER TRACE RECORDS. OLATRACE=* or jobname
BBOO0179I WLM - DISPLAY WLM SETTINGS
BBOO0179I SMF - DISPLAY SMF120-9 SETTINGS AND STATUS
BBOO0179I FRCA - DISPLAY FRCA INFORMATION
BBOO0179I DPM - DISPLAY DISPATCH PROGRESS MONITOR SETTINGS
BBOO0179I TRACERECORD - DISPLAY TRACERECORD SETTING
BBOO0179I MSGROUTE - DISPLAY MESSAGE ROUTING SETTINGS
```

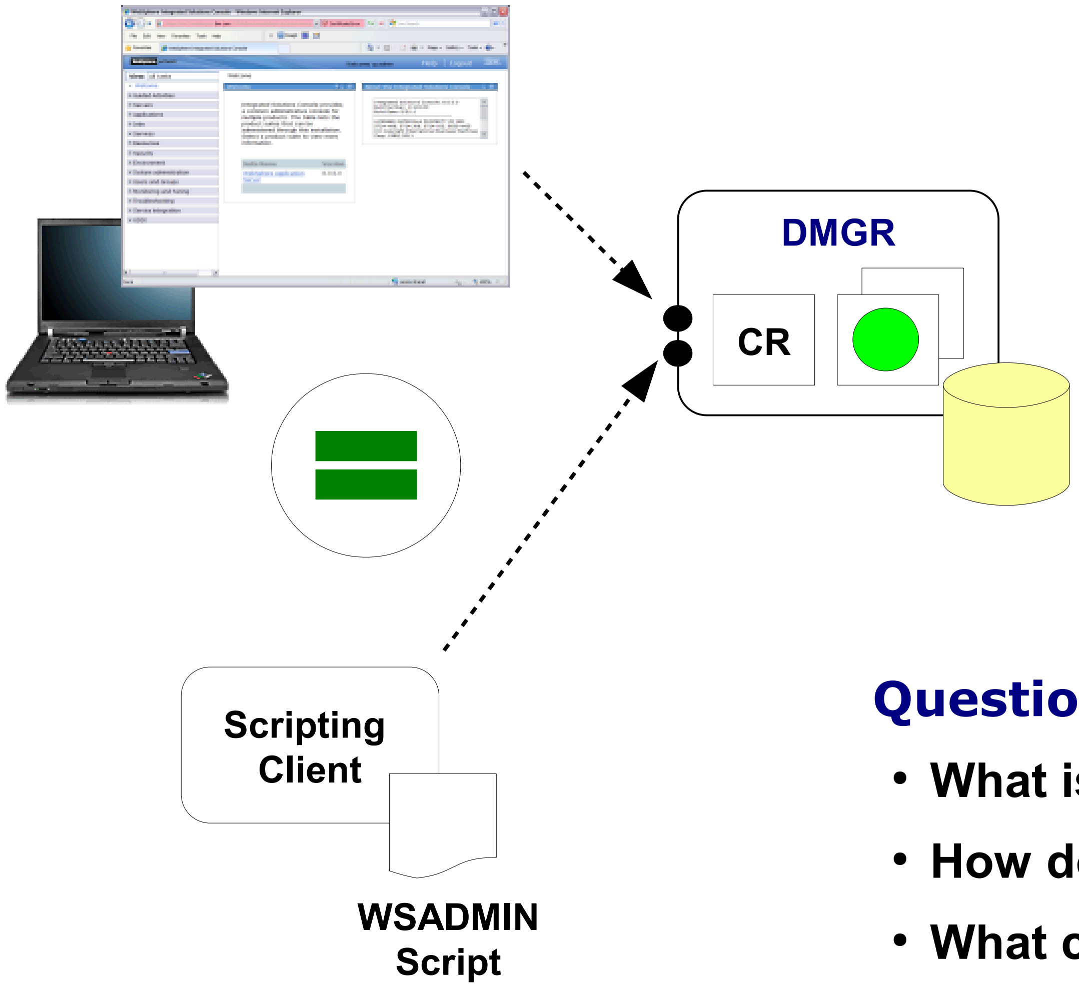
Specifying `,HELP` on many these will then display the parameters acceptable for that particular command

WSADMIN …

WSADMIN

In a Nutshell, WSADMIN is …

… a set of interfaces to the administrative function you may use to automate tasks you might otherwise do with the Admin Console:



Updates to the master configuration, then syncronization out to the nodes

Questions that spill from this:

- What is it?
- How does it work?
- What can I do with it?

Command objects …

The WSADMIN Command Objects

The interface is composed of four main "objects" (commands) that provide the administrative function:

AdminApp
- install
- uninstall
- list
- options
- export
- *… many more*

AdminConfig
- list
- show
- save
- create
- update
- remove
- *… many more*

AdminControl
- startServer
- stopServer
- invoke
- *… many more*

AdminTask
- changeHostName
- modifyServerPort
- *… many more*

Think of AdminTask as commands that contain other more "primitive" WSADMIN commands under the wrapper. It was created as a way to make scripting easier for common tasks ... hence the name "AdminTask"

Key Points:

- WSADMIN is a command interface
- Four major commands, each with many sub-options
- Your script uses these commands to make the changes you wish

A very simple example …

A *Very Simple* Example of Installing an Application

Automating the deployment of applications is a very common use for WSADMIN. Here's an example of a Jython script that installs an application:

Script file

```
server   = 'mysr01a'
node     = 'mynodea'
ear      = '/u/myID/MyIVT.ear'
options = '[-node ' + node + ' -server ' + server + ']'
AdminApp.install(ear,options)
AdminConfig.save()
print AdminApp.list()
```

Set up a few variables to hold the server, node and EAR file location and name

Create a "list variable" (an array) that will hold the server and node values

Invoke the `AdminApp` object using the `install` method and pass in two parameters: the EAR file and the list of options

Invoke the `AdminConfig` object using the `save` method to save changes made to the repository

Use the `AdminApp` object `list` method to validate that the application is indeed installed

Notes:

- **This assumed all deployment defaults taken**
  Deployment options can be set. They become part of "options" ... we just didn't show that to keep this example simple
- **This script did not invoke synchronization**
  Can be done with WSADMIN ... just didn't show to keep simple
- **This script did not start the application**
  Again, can be done with WSADMIN

Slightly more advanced example …

The App Install Script from Upcoming Lab

Uninstalls app if already present, then installs the named application again:

```
import sys
# ---------------------------
#  Uninstall the app if it i
# ---------------------------------
application = ""
applicationlist = AdminApp.list().splitlines()
for application in applicationlist:
    if application == "SuperSnoop":
        print "Found and uninstalling " + application
        AdminApp.uninstall(application)
        AdminConfig.save()
    if application != "SuperSnoop":
        print "Application in list not SuperSnoop. Skipping. Name is: " + application
    continue
# ---------------------------------
# Install SuperSnoop
# ---------------------------------
application = "SuperSnoop"
print "Installing application " + application
earfile = "/wasetc/was8lab/applications/SuperSnoopProj.ear"
appopts = "[-appname "
appopts = appopts + application
appopts = appopts + " -MapModulesToServers [[ SuperSnoopWeb SuperSnoopWeb.war,WEB-INF/web.xml "
appopts = appopts + "WebSphere:cell=z9cell,node=z9nodea,server=z9sr01a ]]]"
# -- debug if needed: comment out if not needed ----------
print "appopts = " + appopts
# -- invoke the AdminApp.install() method ---------
AdminApp.install(earfile, appopts)
AdminConfig.save()
print "Application                                      started"
```

For Loop

Brings in a set of support functions useful to Jython

Retrieves all installed applications in the cell. `splitlines()` splits into a list that can be iterated through

Loops through list. If it finds SuperSnoop it uninstalls it, otherwise it notes application skipped over

Pointer to location of EAR

Building the application install options list array

Debug print ... just to see what the option list passed to `AdminApp.install()` actually looks like.

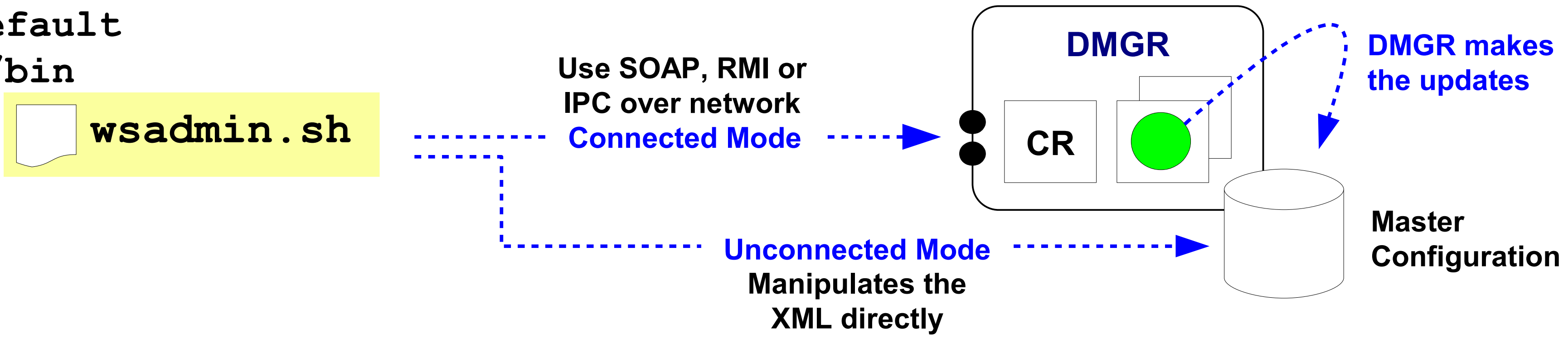
Invoke `AdminApp.install()` and then save the changes to the Master. This does *not* synchronize.

WSADMIN client …

The WSADMIN Client Shell Script and Invocation

To use WSADMIN you must invoke the `wsadmin.sh` client. You pass in the script file you have written. It then works against the interface to do the work ...

```
/wasv85config/aacell/aadmnode/DeploymentManager
└/profiles
  └/default
    └/bin
```



Connected Mode (Recommended whenever DMGR is available)

```
./wsadmin.sh -lang jython -conntype SOAP
    -host www.myhost.com -port 10002
    -user myadmin -password myadmin -f /u/myhome/myscript.jy args
```

Or RMI or IPC with corresponding port

Unconnected Mode

```
./wsadmin.sh -lang jython -conntype NONE -f /u/myhome/myscript.jy args
```

Passing in arguments …

Passing in Arguments to a Script

Scripts may be made even more flexible by passing in arguments on the invocation command and using those passed-in arguments within the script

```
./wsadmin.sh -lang jython -conntype SOAP
   -host www.myhost.com -port 10002 -user myadmin -password myadmin
   -f /u/myhome/myscript.jy z9sr01a z9nodea /u/myID/MyIVT.ear
```

server *node* *EAR location*

```
import sys
server = ""
node   = ""
ear    = ""
if(len(sys.argv) > 2):
    server = sys.argv[0]
    node   = sys.argv[1]
    ear    = sys.argv[2]
else:
    print "Not enough arguments ... exiting"
    sys.exit()
  :
  :
```

Initialize variables

Check to see if there's enough arguments to satisfy the script's requirements. This script expects to see 3 variables. Check for greater than 2.

Parse the arguments and assign each to the respective Jython variable that will be used elsewhere in the script.

Note that sequence of arguments not enforced by Jython. You either assume it's correct or build in more error checking.

With arguments now held in Jython variables you may move on to the rest of your script processing

Script file character encoding …

Character Encoding of the Script file on z/OS

May be either ASCII or EBCDIC. WSADMIN by default expects ASCII. If you want to use EBCDIC you have to tell WSADMIN:

File stored in z/OS USS as ASCII

By default WSADMIN expects script file to have ASCII character encoding so the -javaoption is not needed if ASCII

File stored in z/OS USS as EBCDIC

```
./wsadmin.sh -lang jython -javaoption -Dscript.encoding=Cp1047
   -conntype SOAP -host www.myhost.com -port 10002
   -user myadmin -password myadmin -f /u/myhome/myscript.jy
```

WSADMIN and JCL batch …

WSADMIN and Batch

JCL invoking BPXBATCH works quite well ...

```
//WSADMIN JOB (ACCTNO,ROOM),REGION=0M,USER=MYADMIN,PASSWORD=MYADMIN
//STEP1    EXEC PGM=IKJEFT01
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 BPXBATCH SH +
  /wasv85config/z9cell/z9dmnode/DeploymentManager/profiles/default+
  /bin/wsadmin.sh +
  -lang jython +
  -javaoption -Dscript.encoding=Cp1047 +
  -conntype SOAP +
  -host www.myhost.com +
  -port 10002 +
  -user MYADMIN +
  -password MYADMIN +
  -f /u/myID/myscript.jy args +
 1> /tmp/myID.out +
 2> /tmp/myID.err
/*
```

Complete pointer to the wsadmin.sh client

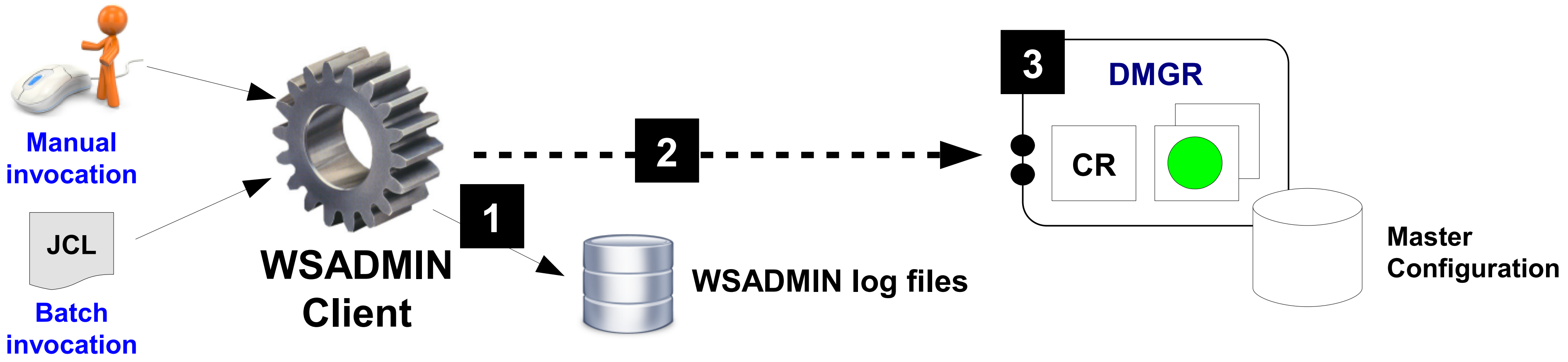
The invocation command is no different than before

This does bring up a few security issues we need to discuss ...

Security ...

WSADMIN and Security

Within what we've discussed so far are three key security considerations that must be taken into account for WSADMIN to work:



1. **File permission access to WSADMIN log and trace files**
   Need write access, which requies at least GROUP access.
   This is the ID used to log into Telnet or USS, or the ID on batch JOB (or effective ID). **WAS Admin ID**

2. **Ability to establish SSL to DMGR when security enabled**
   Implies access to the CA certificate used to sign the DMGR's server certificate.
   This is the ID used to log into Telnet or USS, or the ID on batch JOB (or effective ID). **WAS Admin ID**

3. **Authentication and authorization to in the DMGR to perform the tasks**
   Valid RACF ID and proper access to EJBROLEs.
   This is the userid/password coded on the wsadmin.sh parameters. **Again, WAS Admin ID.**

Other IDs *can* be made to have these properties … WAS Admin ID has it by default

Resources …

Resources for Learning and Reference

The following resources are available to gaining more experience with WSADMIN:

IBM Techdocs -- `ibm.com/support/techdocs`

Techdocs Library > White papers > **WebSphere z/OS V6.1 - WSADMIN Primer (with Jython)**	`WP101014`
Techdocs Library > White papers > **Using Jython Scripting Language With WSADMIN**	`WP100963`
Techdocs Library > Hints, tips & Technotes > **Creating a New Server in WebSphere V7 for z/OS**	`TD105447`
Techdocs Library > White papers > **Staged Application Deployment in WebSphere on z/OS V7**	`WP101641`

IBM InfoCenter -- `publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp`

Network Deployment (z/OS), Version 8.0 > Scripting the application serving environment (wsadmin)

Getting started with wsadmin scripting `txml_script`

Very good reference source for searches on specific WSADMIN commands or methods

WSADMIN client "Help" object and "help" methods

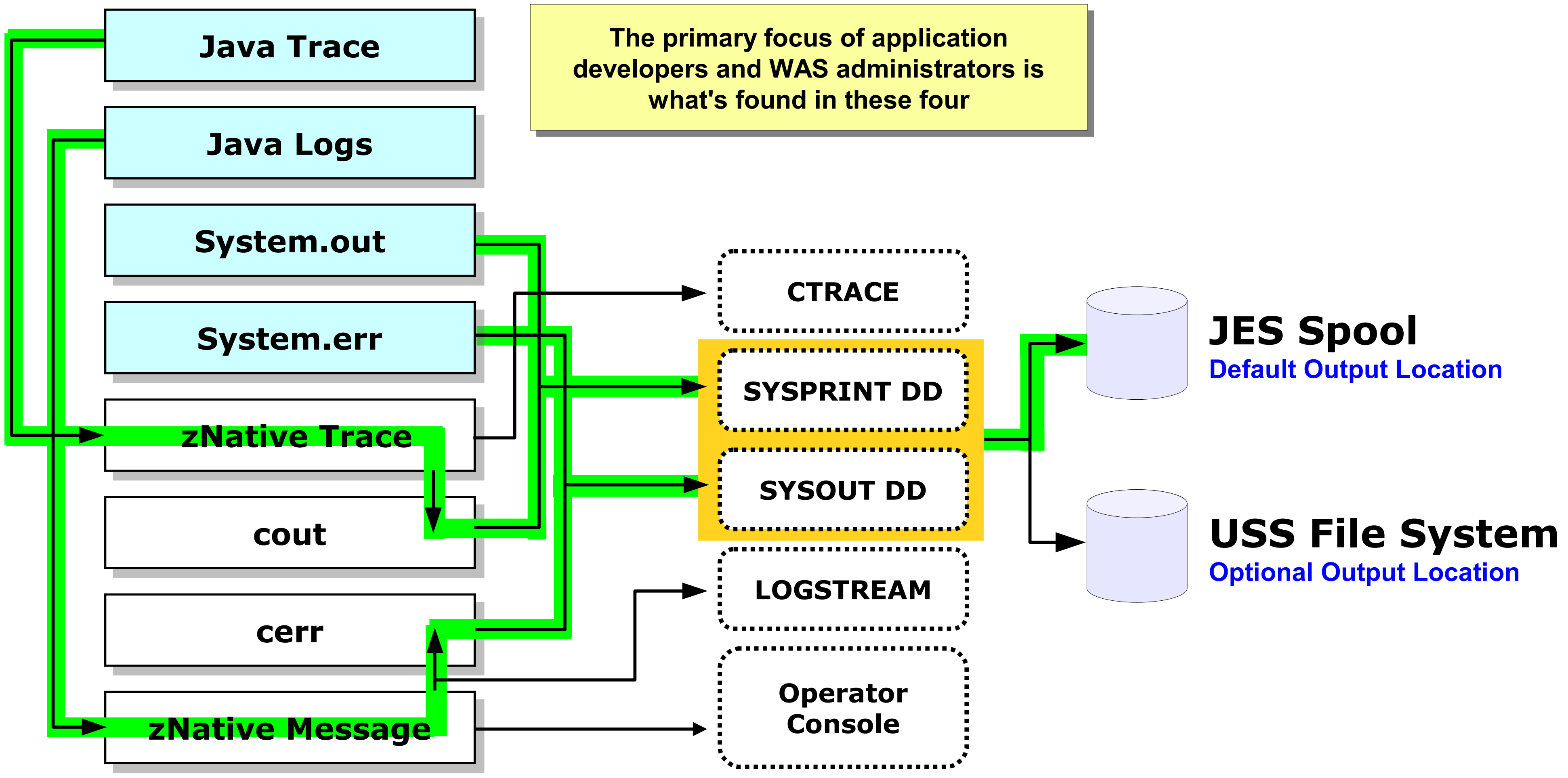
The WSADMIN client has extensive online help in its command syntax. It provides a way to drill down on syntax and usage for specific objects, method and attributes

Logging …

Logging

The Default WAS z/OS Log and Trace Model

There are many sources of logging and tracing in WAS and WAS z/OS. This picture shows where output goes by default in WAS z/OS V8:

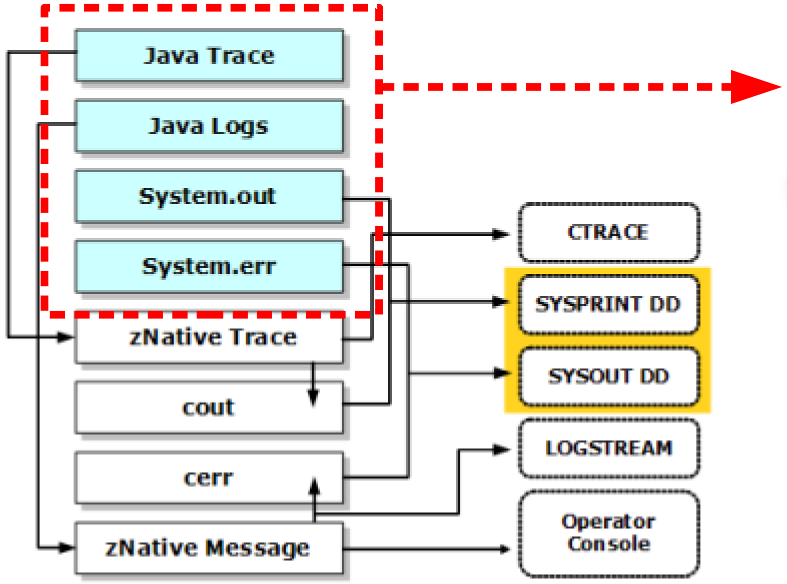


Key Point: **Multiple source of logging and tracing. On z/OS default routes most of it to SYSPRINT and SYSOUT, which then goes to JES spool.**

HPEL …

Introducing High Performance Extensible Logging

Or "HPEL" for short ... it is a new binary logging mechanism in V8 for all platforms. It provides a more efficient logging mechanism than clear text logging



WAS-specific binary format log file

Write to memory buffer, then file

Controls to dictate size limits, what to do when limit reached, how to trim files, start new files, etc.

Viewing the Log:

Admin Console View Facility

Usable tool to view binary file contents. Has ability to filter on criteria to limit what is seen

Log Viewer Shell Script Utility

File is logViewer.sh and it has parameters to limit what is seen in the produced text-readable output file

Download to PC

HPEL file is converted to readable text and downloaded as ZIP to your PC where standard text editors may be employed to view

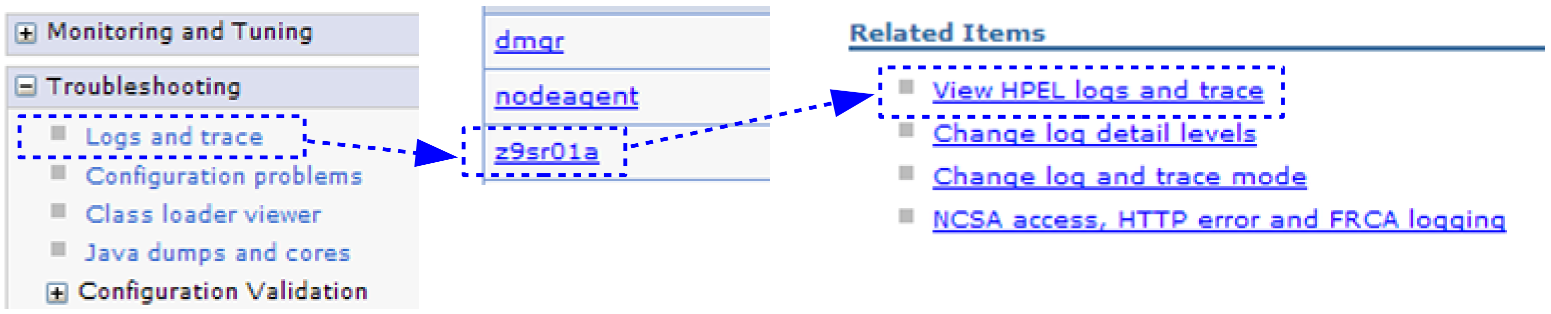
Optional ... Traditional Mode still available and is default

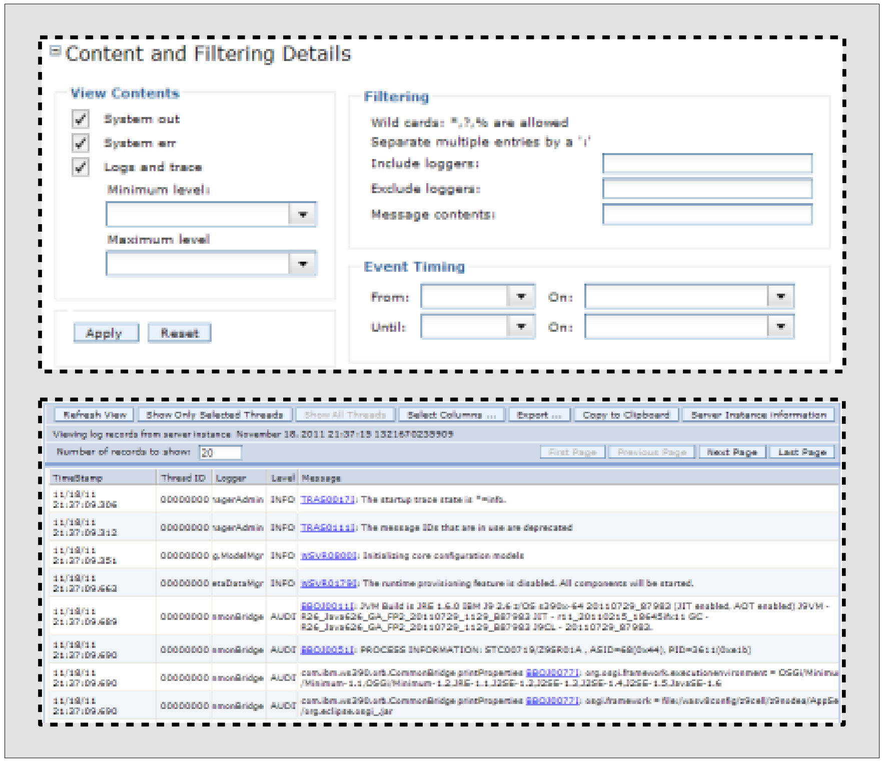
Configurable on a server by server basis

High level of log viewer in Admin Console ...

High-Level of the Admin Console Log Viewer

This is a graphical log viewer supplied as part of the Admin Console:





Viewing log records from server instance November 18, 2011 21:37:15 1321670235909

Number of records to show: 20

TimeStamp	Thread ID	Logger	Level	Message
11/18/11 21:27:09.306	00000000	nagerAdmin	INFO	TRAS0017I: The startup trace state is *=info.
11/18/11 21:27:09.312	00000000	nagerAdmin	INFO	TRAS0111I: The message IDs that are in use are deprecated
11/18/11 21:27:09.351	00000000	g.ModelMgr	INFO	WSVR0800I: Initializing core configuration models
11/18/11 21:27:09.662	00000000	etaDataMgr	INFO	WSVR0179I: The runtime provisioning feature is disabled. All components will be started.
11/18/11 21:27:09.689	00000000	imonBridge	AUDI	BBOJ0011I: JVM Build is JRE 1.6.0 IBM J9 2.6 z/OS s390x-64 20110729_87982 (JIT enabled, AOT enabled) J9VM - R26_Java626_GA_FP2_20110729_1129_B87982 JIT - r11_20110215_18645ifx11 GC - R26_Java626_GA_FP2_20110729_1129_B87982 J9CL - 20110729_87982.
11/18/11 21:27:09.690	00000000	imonBridge	AUDI	BBOJ0051I: PROCESS INFORMATION: STC00719/Z9SR01A , ASID=68(0x44), PID=3611(0xe1b)
11/18/11 21:27:09.690	00000000	imonBridge	AUDI	com.ibm.ws390.orb.CommonBridge printProperties BBOJ0077I: org.osgi.framework.executionenvironment = OSGi/Minimu /Minimum-1.1,OSGi/Minimum-1.2,JRE-1.1,J2SE-1.2,J2SE-1.3,J2SE-1.4,J2SE-1.5,JavaSE-1.6
11/18/11 21:27:09.690	00000000	imonBridge	AUDI	com.ibm.ws390.orb.CommonBridge printProperties BBOJ0077I: osgi.framework = file:/wasv8config/z9cell/z9nodea/AppSe /org.eclipse.osgi_.jar

Log selection and content filtering section. This influences what appears below …

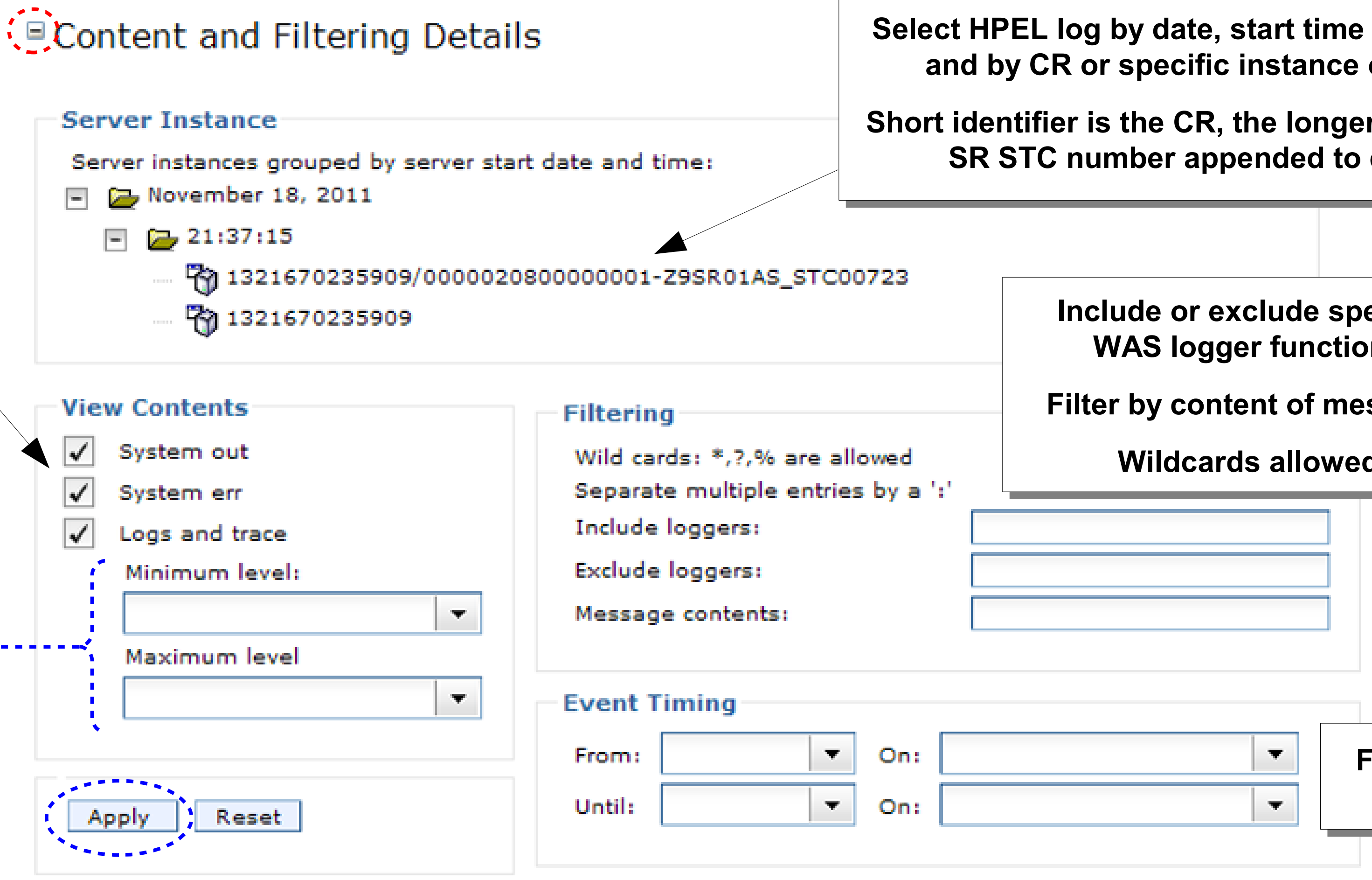
… record by record display of HPEL content based on filtering down above.

Content and filtering details …

Content and Filtering Details

This allows you to determine, with a fair degree of granularity, what HPEL records will be displayed in the output result set:

Expand the section with this twisty



Content and Filtering Details

Server Instance

Server instances grouped by server start date and time:

- November 18, 2011
  - 21:37:15
    - 1321670235909/0000020800000001-Z9SR01AS_STC00723
    - 1321670235909

View Contents

- [✓] System out
- [✓] System err
- [✓] Logs and trace

Minimum level:

Maximum level

Filtering

Wild cards: *,?,% are allowed

Separate multiple entries by a ':'

Include loggers:

Exclude loggers:

Message contents:

Event Timing

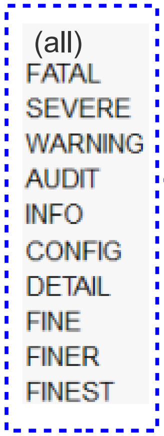
From: On:

Until: On:

Select HPEL log by date, start time of server, and by CR or specific instance of SR.

Short identifier is the CR, the longer the SRs. SR STC number appended to end.

Filter by type of output



Include or exclude specific WAS logger functions

Filter by content of message

Wildcards allowed

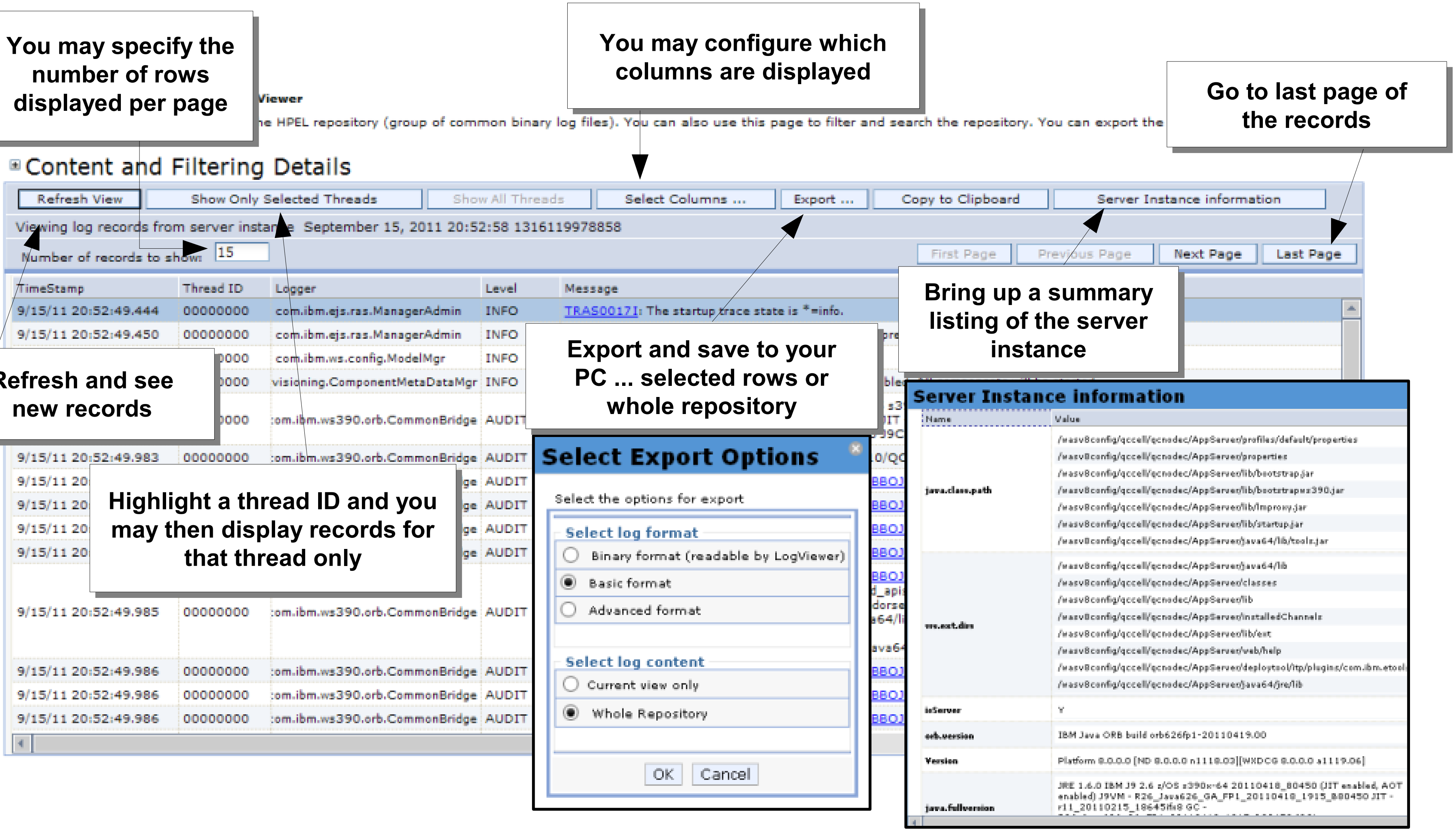
Filter by range of time and date

"Apply" will put into effect your filtering selection

Record display …

Record Display of Content Based on Filtering

This displays in your Admin Console for the selected server's HPEL log:



You may specify the number of rows displayed per page

You may configure which columns are displayed

Go to last page of the records

Refresh and see new records

Bring up a summary listing of the server instance

Export and save to your PC ... selected rows or whole repository

Highlight a thread ID and you may then display records for that thread only

Viewer

he HPEL repository (group of common binary log files). You can also use this page to filter and search the repository. You can export the

Content and Filtering Details

Viewing log records from server instance September 15, 2011 20:52:58 1316119978858

Number of records to show: 15

TimeStamp	Thread ID	Logger	Level	Message
9/15/11 20:52:49.444	00000000	com.ibm.ejs.ras.ManagerAdmin	INFO	TRAS0017I: The startup trace state is *=info.
9/15/11 20:52:49.450	00000000	com.ibm.ejs.ras.ManagerAdmin	INFO	
	0000	com.ibm.ws.config.ModelMgr	INFO	
	0000	visioning.ComponentMetaDataMgr	INFO	
	0000	:om.ibm.ws390.orb.CommonBridge	AUDIT	
9/15/11 20:52:49.983	00000000	:om.ibm.ws390.orb.CommonBridge	AUDIT	
9/15/11 20		ge	AUDIT	
9/15/11 20		ge	AUDIT	
9/15/11 20		ge	AUDIT	
9/15/11 20		ge	AUDIT	
9/15/11 20:52:49.985	00000000	:om.ibm.ws390.orb.CommonBridge	AUDIT	
9/15/11 20:52:49.986	00000000	:om.ibm.ws390.orb.CommonBridge	AUDIT	
9/15/11 20:52:49.986	00000000	:om.ibm.ws390.orb.CommonBridge	AUDIT	
9/15/11 20:52:49.986	00000000	:om.ibm.ws390.orb.CommonBridge	AUDIT	

Select Export Options

Select the options for export

Select log format
- ( ) Binary format (readable by LogViewer)
- [✓ Basic format]
- ( ) Advanced format

Select log content
- ( ) Current view only
- [✓ Whole Repository]

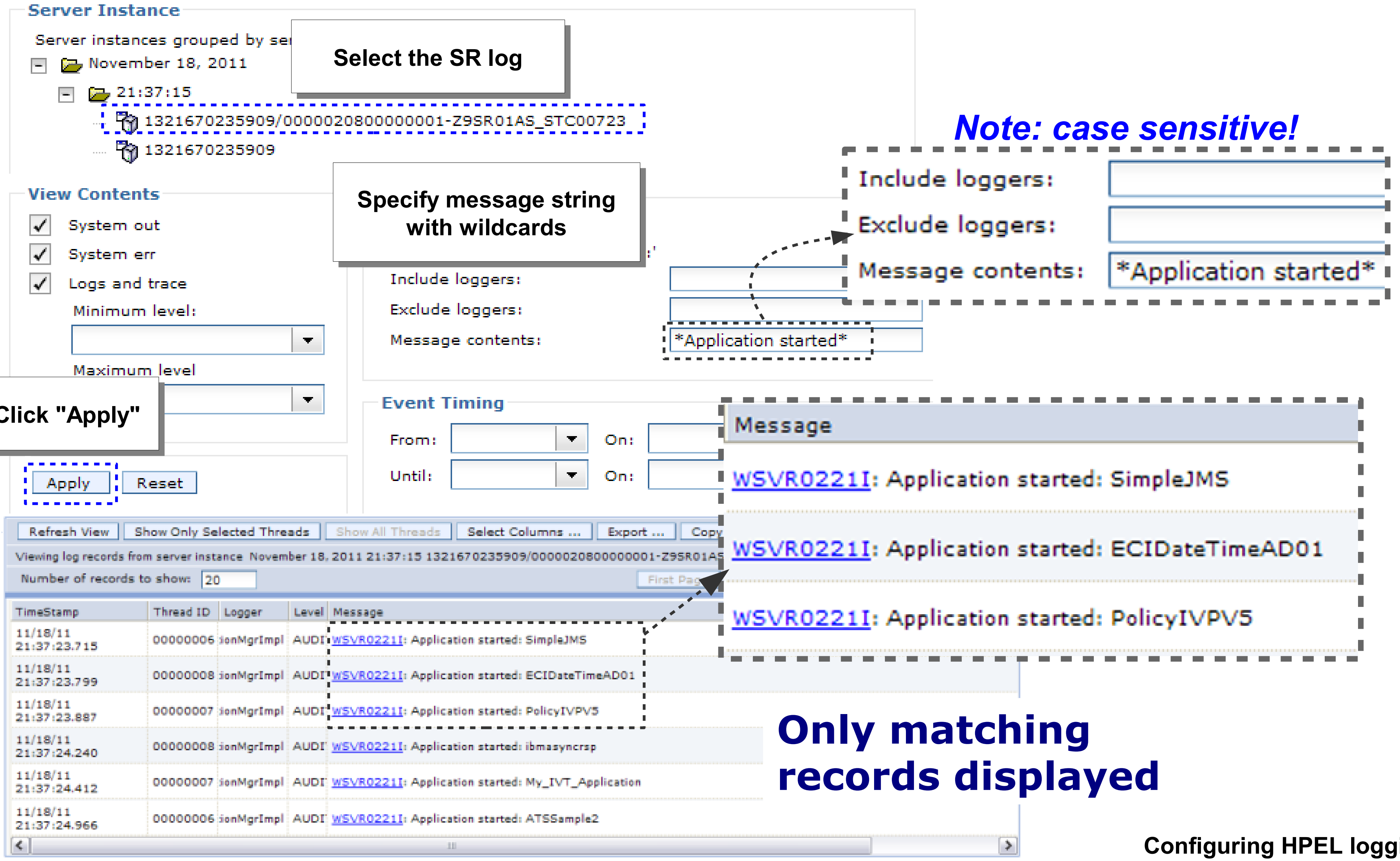
Server Instance information

Name	Value
java.class.path	/wasv8config/qccell/qcnodec/AppServer/profiles/default/properties /wasv8config/qccell/qcnodec/AppServer/properties /wasv8config/qccell/qcnodec/AppServer/lib/bootstrap.jar /wasv8config/qccell/qcnodec/AppServer/lib/bootstrapws390.jar /wasv8config/qccell/qcnodec/AppServer/lib/lmproxy.jar /wasv8config/qccell/qcnodec/AppServer/lib/startup.jar /wasv8config/qccell/qcnodec/AppServer/java64/lib/tools.jar
ws.ext.dirs	/wasv8config/qccell/qcnodec/AppServer/java64/lib /wasv8config/qccell/qcnodec/AppServer/classes /wasv8config/qccell/qcnodec/AppServer/lib /wasv8config/qccell/qcnodec/AppServer/installedChannels /wasv8config/qccell/qcnodec/AppServer/lib/ext /wasv8config/qccell/qcnodec/AppServer/web/help /wasv8config/qccell/qcnodec/AppServer/deploytool/itp/plugins/com.ibm.etools /wasv8config/qccell/qcnodec/AppServer/java64/jre/lib
isServer	Y
orb.version	IBM Java ORB build orb626fp1-20110419.00
Version	Platform 8.0.0.0 [ND 8.0.0.0 n1118.03][WXDCG 8.0.0.0 a1119.06]
java.fullversion	JRE 1.6.0 IBM J9 2.6 z/OS s390x-64 20110418_80450 (JIT enabled, AOT enabled) J9VM - R26_Java626_GA_FP1_20110418_1915_B80450 JIT - r11_20110215_18645ifx8 GC -

Example ...

Simple Example of Filtering

Suppose you wish to see all the "Application started" messages:



Select the SR log

Specify message string with wildcards

Click "Apply"

Note: case sensitive!

Only matching records displayed

Server Instance

Server instances grouped by se

November 18, 2011

21:37:15

1321670235909/0000020800000001-Z9SR01AS_STC00723

1321670235909

View Contents

- ✓ System out
- ✓ System err
- ✓ Logs and trace

Minimum level:

Maximum level

Include loggers:

Exclude loggers:

Message contents: *Application started*

Event Timing

From: On:

Until: On:

Viewing log records from server instance November 18, 2011 21:37:15 1321670235909/0000020800000001-Z9SR01AS

Number of records to show: 20

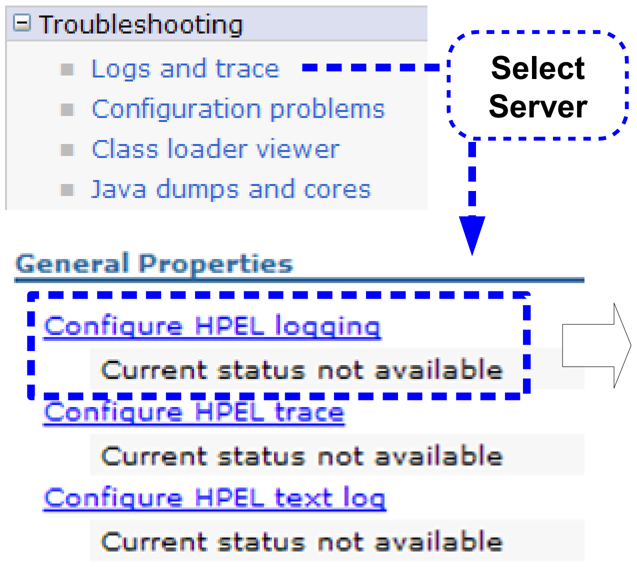
TimeStamp	Thread ID	Logger	Level	Message
11/18/11 21:37:23.715	00000006	ionMgrImpl	AUDI	WSVR0221I: Application started: SimpleJMS
11/18/11 21:37:23.799	00000008	ionMgrImpl	AUDI	WSVR0221I: Application started: ECIDateTimeAD01
11/18/11 21:37:23.887	00000007	ionMgrImpl	AUDI	WSVR0221I: Application started: PolicyIVPV5
11/18/11 21:37:24.240	00000008	ionMgrImpl	AUDI	WSVR0221I: Application started: ibmasyncrsp
11/18/11 21:37:24.412	00000007	ionMgrImpl	AUDI	WSVR0221I: Application started: My_IVT_Application
11/18/11 21:37:24.966	00000006	ionMgrImpl	AUDI	WSVR0221I: Application started: ATSSample2

Message
WSVR0221I: Application started: SimpleJMS
WSVR0221I: Application started: ECIDateTimeAD01
WSVR0221I: Application started: PolicyIVPV5

Configuring HPEL logging …

Configuring HPEL Logging for a Server

Process is relatively easy with a great deal of configurable options ...

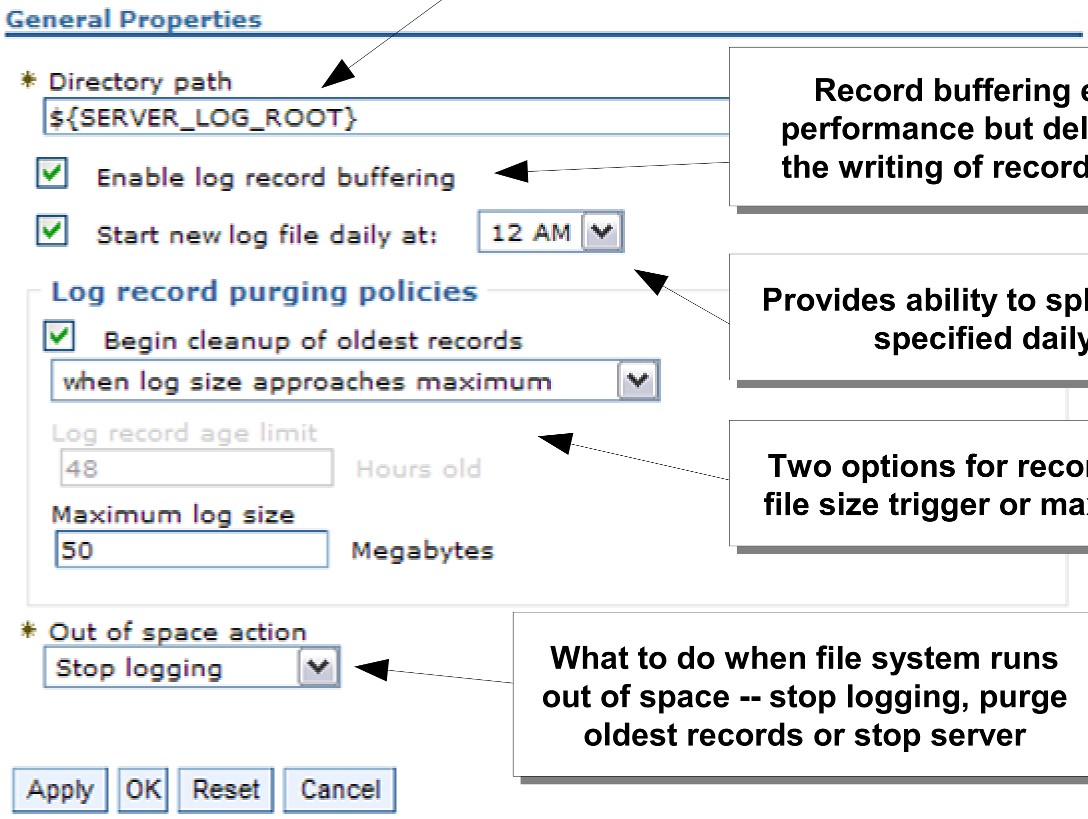


⊟ Troubleshooting
- Logs and trace
- Configuration problems
- Class loader viewer
- Java dumps and cores

Select Server

General Properties

Configure HPEL logging
Current status not available
Configure HPEL trace
Current status not available
Configure HPEL text log
Current status not available



General Properties

* Directory path
${SERVER_LOG_ROOT}

✓ Enable log record buffering

✓ Start new log file daily at: 12 AM

Log record purging policies

✓ Begin cleanup of oldest records
when log size approaches maximum

Log record age limit
48 Hours old

Maximum log size
50 Megabytes

* Out of space action
Stop logging

Specifies location where HPEL log directories and files will reside

Record buffering enhances performance but delays slightly the writing of records to the file

Provides ability to split the logs at specified daily time

Two options for record purging -- file size trigger or max age trigger

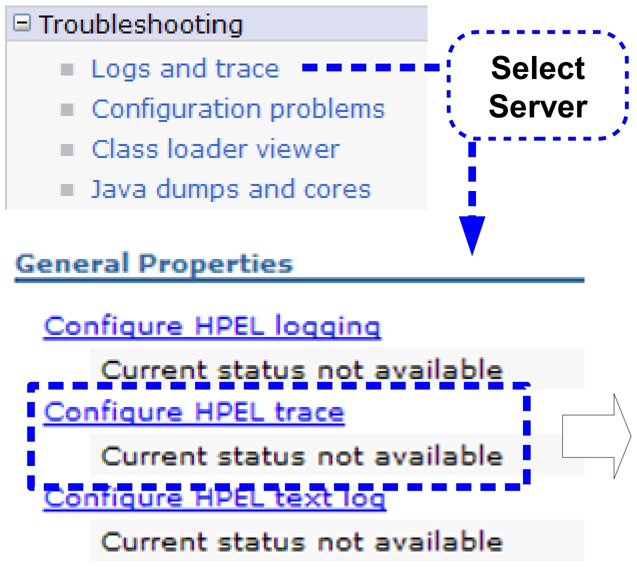
What to do when file system runs out of space -- stop logging, purge oldest records or stop server

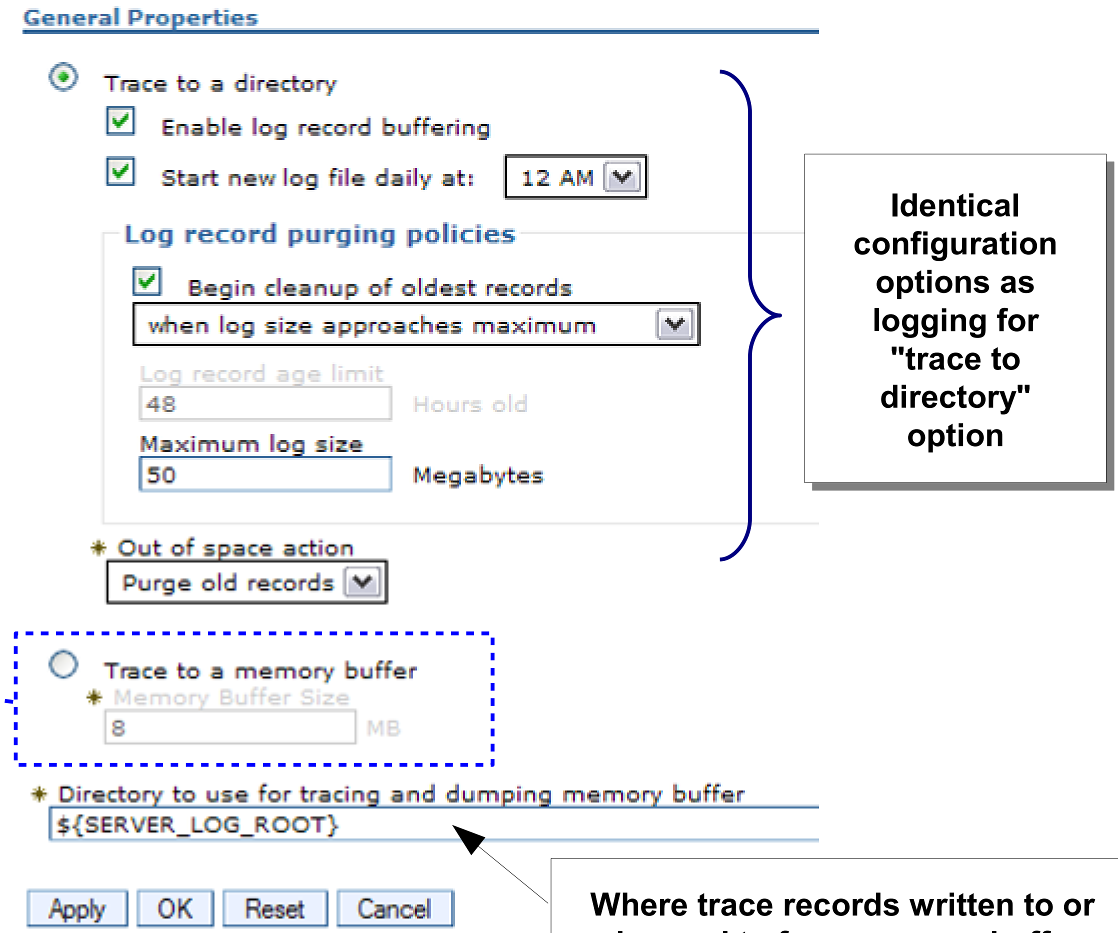
Server Restart Needed

Configuring HPEL tracing …

Configuring HPEL Tracing for a Server

Process is relatively easy with a great deal of configurable options ...




Identical configuration options as logging for "trace to directory" option
Where trace records written to or dumped to from memory buffer
Apply
OK
Reset
Cancel

General Properties

● Trace to a directory

✓ Enable log record buffering

✓ Start new log file daily at: 12 AM

Log record purging policies

✓ Begin cleanup of oldest records: when log size approaches maximum

Log record age limit: 48 Hours old

Maximum log size: 50 Megabytes

* Out of space action: Purge old records

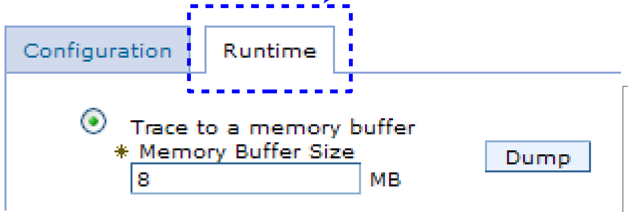
○ Trace to a memory buffer

* Memory Buffer Size: 8 MB

* Directory to use for tracing and dumping memory buffer: ${SERVER_LOG_ROOT}

Server Restart Needed

Alternatively, trace to a memory buffer



● Trace to a memory buffer

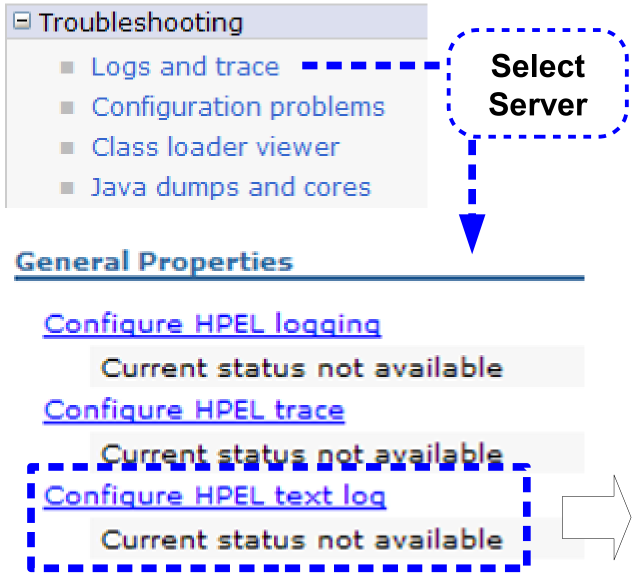
* Memory Buffer Size: 8 MB

Dump button writes memory buffer to file trace log where log viewer may display it

Text logger …

Configuring Optional HPEL Text Logging for a Server

Process is relatively easy with a great deal of configurable options ...



⊟ Troubleshooting

- Logs and trace
- Configuration problems
- Class loader viewer
- Java dumps and cores

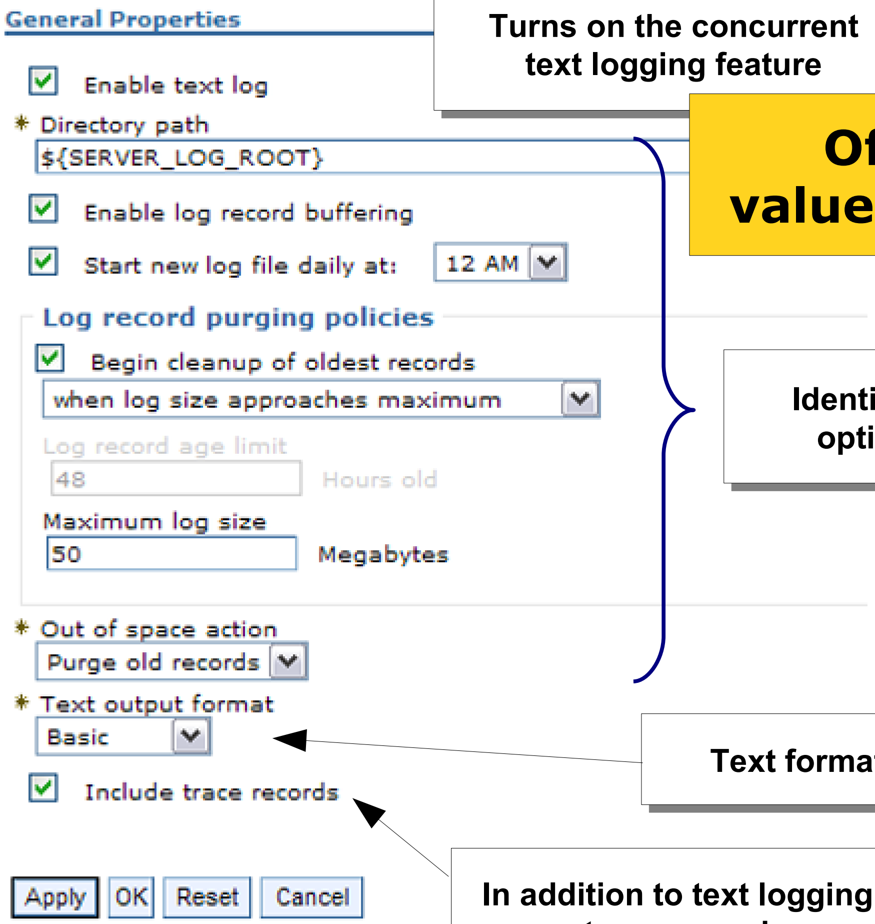
Select Server

General Properties

Configure HPEL logging
Current status not available

Configure HPEL trace
Current status not available

Configure HPEL text log
Current status not available



General Properties

- [✓] Enable text log
- * Directory path: ${SERVER_LOG_ROOT}
- [✓] Enable log record buffering
- [✓] Start new log file daily at: 12 AM

Log record purging policies

- [✓] Begin cleanup of oldest records: when log size approaches maximum
- Log record age limit: 48 Hours old
- Maximum log size: 50 Megabytes

- * Out of space action: Purge old records
- * Text output format: Basic
- [✓] Include trace records

Apply | OK | Reset | Cancel

Turns on the concurrent text logging feature

Of limited value … CR only!

Identical configuration options as logging

Text format

In addition to text logging, include trace records as well

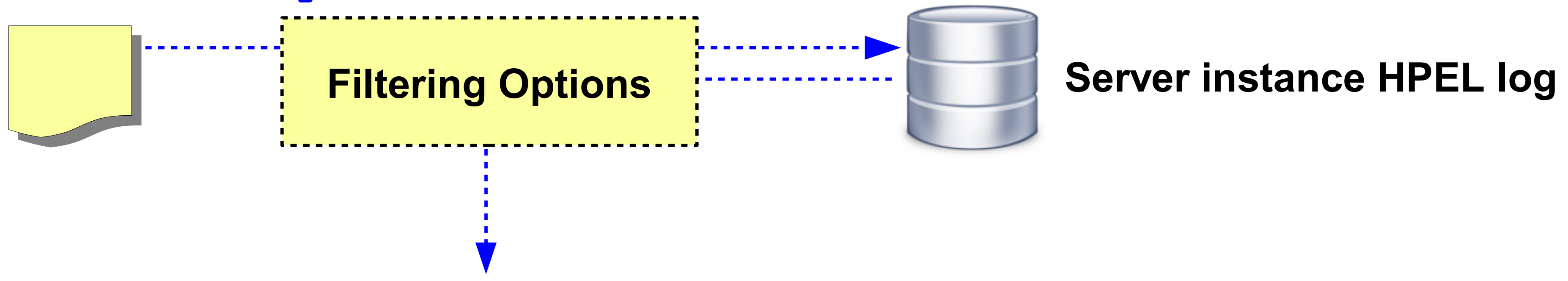
Server Restart Needed

Command line logViewer.sh …

The `logViewer.sh` Utility

A command-line utility to extract and view information from the binary HPEL logs. It has the same capabilities as the Admin Console log viewer:

`logViewer.sh` ***`-parameters`***



- Extract to a text-readable file
- Extract to a separate HPEL log
- Continuously monitor new output

Let's take a tour of the parameters and options of this shell script utility. In lab you'll get a chance to use it.

Simple example …

The `logViewer.sh` Utility - Simple Starting Example

Here's a starting example of using `logViewer.sh` ... first list the server instances, then use the server instance ID to extract a human-readable output file:

Reminder of default location of HPEL logs *for a given server:*

```
/wasv85config/z9cell/z9nodea/AppServer/profiles/default/logs/z9sr01a
```

... and the location of the `logViewer.sh` shell script:

```
/wasv85config/z9cell/z9nodea/AppServer/profiles/default/bin/logViewer.sh
```

Determines the HPEL server instance logs that are present *(all on one line)*

```
./logViewer.sh
  -repositoryDir /wasv85config/z9cell/z9nodea/AppServer/profiles/default/logs/z9sr01a
  -listInstances
```

```
Instance ID                                              Start Date
1321728440474                                            11/19/11 13:47:20.474 EST
1321728440474/0000012000000040-Z9SR01AS_STC00730         11/19/11 13:47:27.081 EST
1321728440474/0000010800000046-Z9SR01AS_STC00731         11/19/11 13:47:40.617 EST
```

Extract the HPEL log to a text-readable file ... view or download *(all on one line)*

```
./logViewer.sh
  -repositoryDir /wasv85config/z9cell/z9nodea/AppServer/profiles/default/logs/z9sr01a
  -instance 1321728440474/0000012000000040-Z9SR01AS_STC00730
  -outLog /tmp/hpelout.txt
```

This file has the same format on z/OS as on other platforms. In EBCDIC, so download using "ascii"

Syntax ...

Details of `logViewer.sh` Parameters

Here is the complete list of options and a brief description of each. Notice how the parameters mirror the Admin Console log viewer options:

`logViewer.sh`

`-repositoryDir <directory name>`	*Location of HPEL repository to read from*
`-outLog <file_name>`	*Path and file name of the output file*
`-startDate <date_time>`	*Extract only records after this date and time (help provides syntax options)*
`-stopDate <date_time>`	*Extract only records before this date and time*
`-level FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`	*Level to extract*
`-minLevel FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`	*Start range to extract*
`-maxLevel FINEST | FINER | FINE | DETAIL | CONFIG | INFO | AUDIT | WARNING | SEVERE | FATAL`	*End range to extract*
`-format <basic | advanced | CBE-1.0.1>`	*Format of output file*
`-monitor [interval]`	*Continuously monitor and update output file*
`-includeLoggers <logger_names>`	*Include loggers by logger class name*
`-excludeLoggers <logger_names>`	*Exclude loggers by logger class name*
`-thread <thread_id>`	*Extract only for specified thread*
`-extractToNewRepository <directory_name>`	*Option to create a sub-repository based on extract rules*
`-listInstances`	*List the server process instances found in the repository*
`-instance <instanceid>`	*Extract only named process instance*
`-latestInstance`	*Extract only the most recent server process instance*
`-message <message>`	*Extract records that match message mask … asterisk and question mark wildcards allowed*

APAR PM74923 …

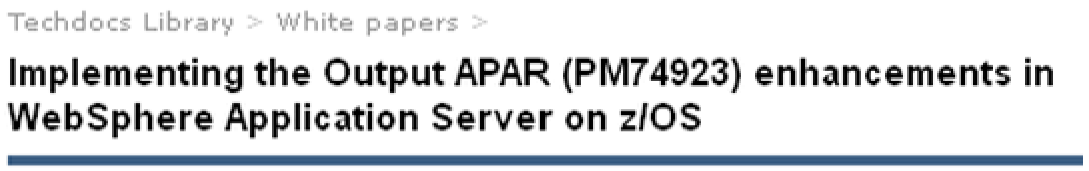
Newest Method of Managing Output

New function introduced via an APAR to V7, V8 or V8.5:

- **Introduced by APAR PM74923**

Version	Maintenance
V7	7.0.0.29
V8	8.0.0.6
V8.5	8.5.0.2

- **Fully describe in techdoc:**



Techdocs Library > White papers >

Implementing the Output APAR (PM74923) enhancements in WebSphere Application Server on z/OS

`http://www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102267`

Overview …

The Method Introduced By the APAR

Produces log output to a file system location you specify:

- **File based.**
- **WebSphere ensures that there will be no naming conflicts**
- **Works for all components (Daemon, Dmgr, Nodeagents, Servers (both controllers, adjuncts, and servants)**
- **File switching is simple using a z/OS MODIFY command**
- **No need for users to access the filesystem**
- **Access to the output can be uncontrolled or controlled...**
  - **At the cell level**
  - **At the node**
  - **Server by server**
  - **Via normal security on the filesystem ... or ...**
  - **Via any security system that the HTTP server supports (SAF, LDAP, etc.)**

Variables ...

How does it work? What do I do to set it up?

It's really fairly simple ... a few WebSphere variables:

In WebSphere:

Add variables at appropriate scopes.

- **They will be inherited by lower levels...**

  Simplest setup is to just add them at the cell level and let all components write to the same path

  The same variable at a lower level will take precedence

- **Variable names:**

  `DAEMON_redirect_server_output_dir` **(for the Daemon)**

  `redirect_server_output_dir` **(for everything else)**

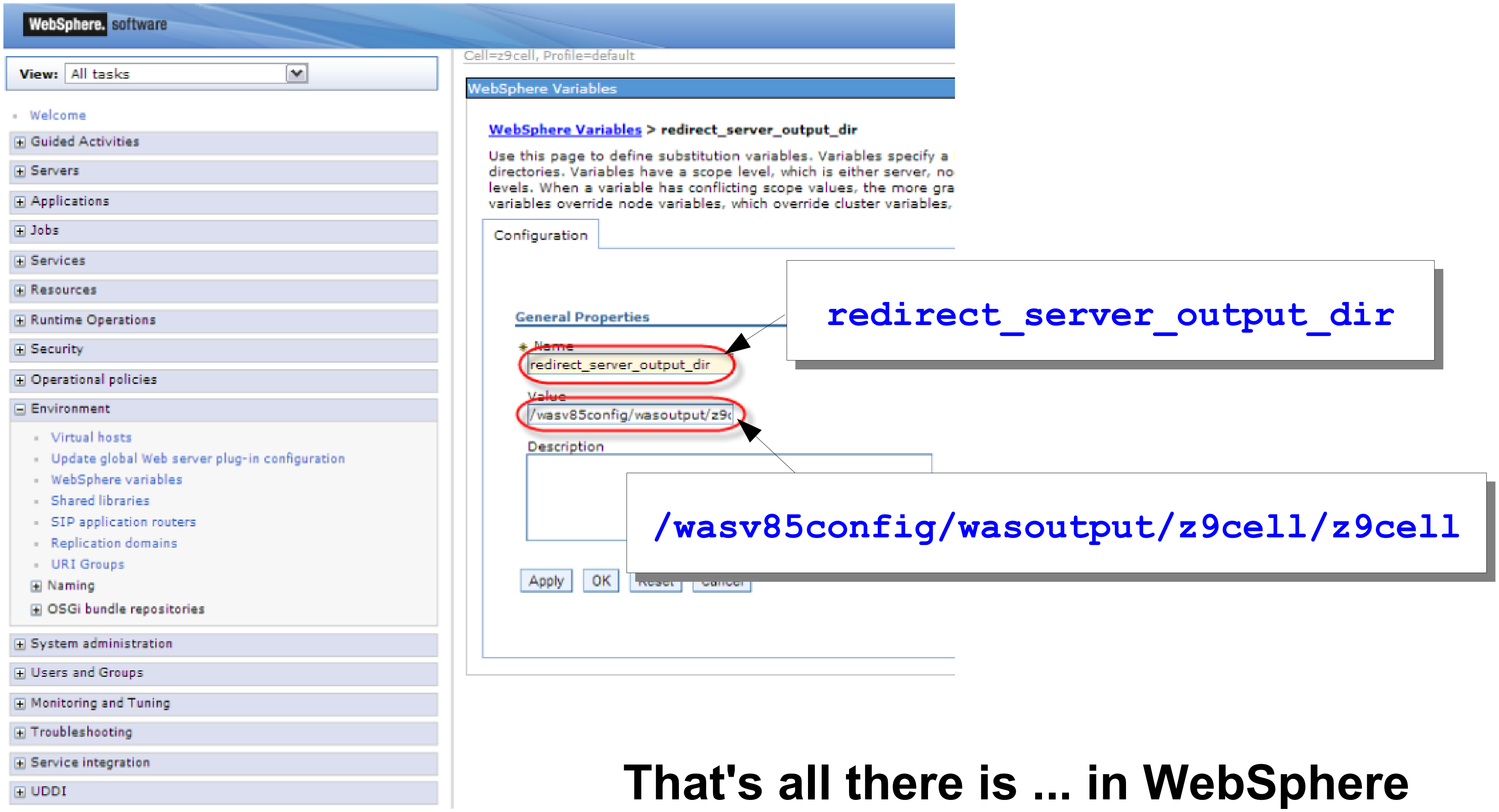
  Value is simply the path name where you wish the output to be written

  Example: `/wasv85config/wasoutput/z9cell/z9cell`

Admin Console ...

Setup in WebSphere

This is standard WebSphere environment variable setup:



Cell=z9cell, Profile=default

WebSphere Variables

WebSphere Variables > redirect_server_output_dir

Use this page to define substitution variables. Variables specify a directories. Variables have a scope level, which is either server, no levels. When a variable has conflicting scope values, the more gra variables override node variables, which override cluster variables.

Name: redirect_server_output_dir

Value: /wasv85config/wasoutput/z9c

Description

redirect_server_output_dir

/wasv85config/wasoutput/z9cell/z9cell

That's all there is ... in WebSphere

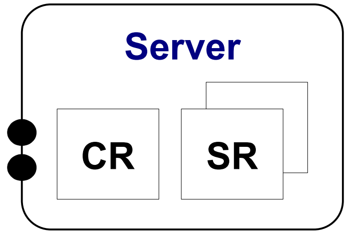
Server restart …

Server Restart and the Result ...

In WebSphere all that is left is to restart the components you believe that you have modified, and the output will be redirected to the path you've specified.

`DAEMON_redirect_server_output_dir` (for the Daemon)

`redirect_server_output_dir` (for everything else)



```
:/shared/wasoutput/z9cell/z9cell
-> ls
Z9CELL.Z9DMNODE.WG31.Z9DEMN.STC12090.DAEMON.130701.174549.SYSOUT.txt
Z9CELL.Z9DMNODE.WG31.Z9DEMN.STC12090.DAEMON.130701.174549.SYSPRINT.txt
Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGR.STC12089.CTL.130701.134548.SYSOUT.txt
Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGR.STC12089.CTL.130701.134548.SYSPRINT.txt
Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGRS.STC12091.SR.130701.134606.SYSOUT.txt
Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGRS.STC12091.SR.130701.134606.SYSPRINT.txt
Z9CELL.Z9NODEA.Z9AGNTA.Z9AGNTA.STC12092.CTL.130701.134710.SYSOUT.txt
Z9CELL.Z9NODEA.Z9AGNTA.Z9AGNTA.STC12092.CTL.130701.134710.SYSPRINT.txt
```

Output for the Daemon

Output for other servers in the cell

Rolling log files …

Rolling Log Files

If you wish to switch to a new file, from any z/OS Console:

Server JOBNAME

Modify Action

```
F Z9DEMN,ROLL_LOGS

BBOO0211I MODIFY COMMAND ROLL_LOGS COMPLETED SUCCESSFULLY
```

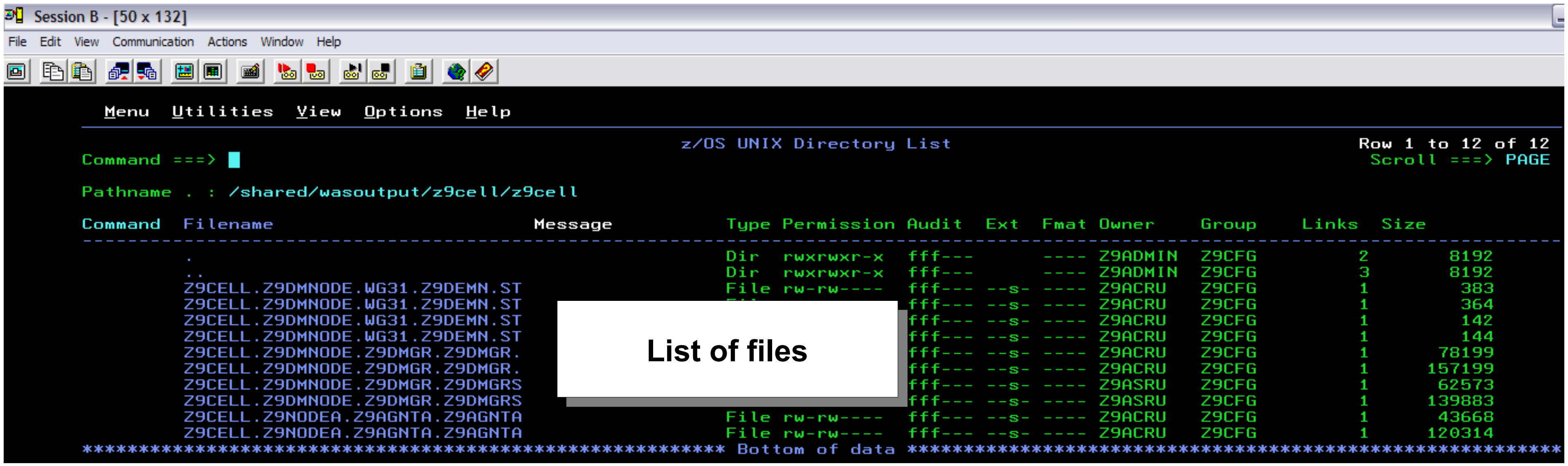
```
:/shared/wasoutput/z9cell/z9cell
-> ls
Z9CELL.Z9DMNODE.WG31.Z9DEMN.STC12090.DAEMON.130701.174549.SYSOUT.txt
Z9CELL.Z9DMNODE.WG31.Z9DEMN.STC12090.DAEMON.130701.174549.SYSPRINT.txt
Z9CELL.Z9DMNODE.WG31.Z9DEMN.STC12090.DAEMON.130701.174907.SYSOUT.txt
Z9CELL.Z9DMNODE.WG31.Z9DEMN.STC12090.DAEMON.130701.174907.SYSPRINT.txt
Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGR.STC12089.CTL.130701.134548.SYSOUT.txt
Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGR.STC12089.CTL.130701.134548.SYSPRINT.txt
Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGRS.STC12091.SR.130701.134606.SYSOUT.txt
Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGRS.STC12091.SR.130701.134606.SYSPRINT.txt
Z9CELL.Z9NODEA.Z9AGNTA.Z9AGNTA.STC12092.CTL.130701.134710.SYSOUT.txt
Z9CELL.Z9NODEA.Z9AGNTA.Z9AGNTA.STC12092.CTL.130701.134710.SYSPRINT.txt
```

At this point, if all you want is ISPF browse or telnet access, you are done...

Result …

The Result

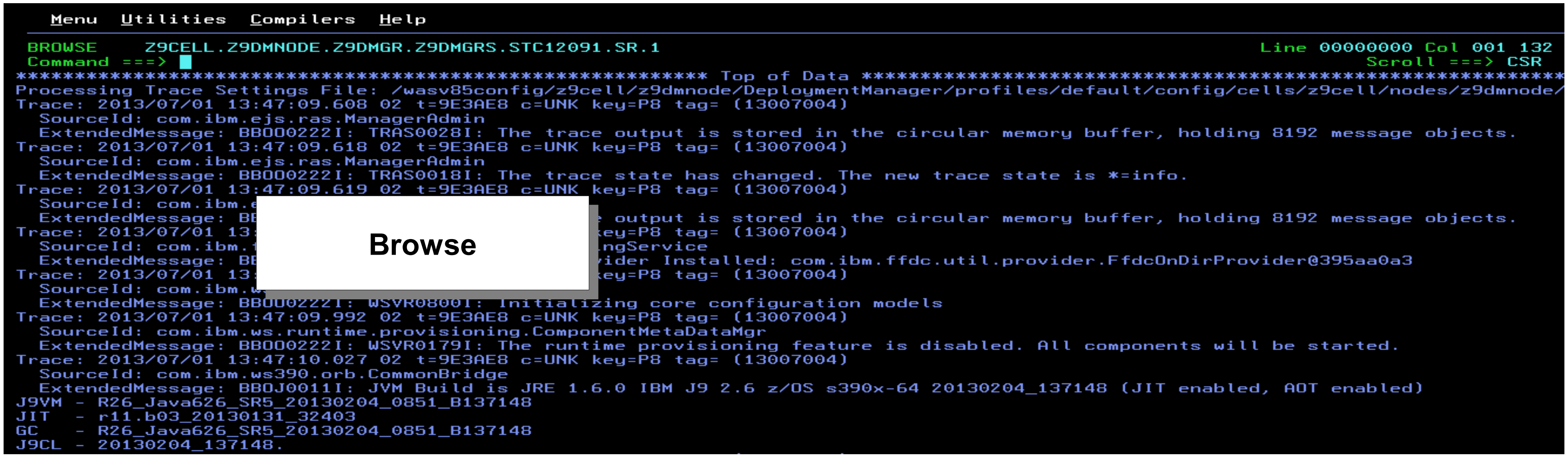
Then browse any file name you choose:



```
   Menu  Utilities  View  Options  Help
 ------------------------------------------------------------------------------------------------------------------------------------
                                     z/OS UNIX Directory List                                              Row 1 to 12 of 12
 Command ===>                                                                                               Scroll ===> PAGE

 Pathname . : /shared/wasoutput/z9cell/z9cell

 Command  Filename                     Message           Type Permission Audit  Ext  Fmat Owner    Group    Links  Size
 ------------------------------------------------------------------------------------------------------------------------------------
          .                                              Dir  rwxrwxr-x  fff---      ---- Z9ADMIN  Z9CFG       2       8192
          ..                                             Dir  rwxrwxr-x  fff---      ---- Z9ADMIN  Z9CFG       3       8192
          Z9CELL.Z9DMNODE.WG31.Z9DEMN.ST                 File rw-rw----  fff--- --s- ---- Z9ACRU   Z9CFG       1        383
          Z9CELL.Z9DMNODE.WG31.Z9DEMN.ST                             fff--- --s- ---- Z9ACRU   Z9CFG       1        364
          Z9CELL.Z9DMNODE.WG31.Z9DEMN.ST                             fff--- --s- ---- Z9ACRU   Z9CFG       1        142
          Z9CELL.Z9DMNODE.WG31.Z9DEMN.ST                             fff--- --s- ---- Z9ACRU   Z9CFG       1        144
          Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGR.                             fff--- --s- ---- Z9ACRU   Z9CFG       1      78199
          Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGR.                             fff--- --s- ---- Z9ACRU   Z9CFG       1     157199
          Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGRS                             fff--- --s- ---- Z9ASRU   Z9CFG       1      62573
          Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGRS                             fff--- --s- ---- Z9ASRU   Z9CFG       1     139883
          Z9CELL.Z9NODEA.Z9AGNTA.Z9AGNTA                 File rw-rw----  fff--- --s- ---- Z9ACRU   Z9CFG       1      43668
          Z9CELL.Z9NODEA.Z9AGNTA.Z9AGNTA                 File rw-rw----  fff--- --s- ---- Z9ACRU   Z9CFG       1     120314
 ******************************************************** Bottom of data ***********************************************************
```



```
    Menu  Utilities  Compilers  Help
 ------------------------------------------------------------------------------------------------------------------------------------
 BROWSE    Z9CELL.Z9DMNODE.Z9DMGR.Z9DMGRS.STC12091.SR.1                                              Line 00000000 Col 001 132
 Command ===>                                                                                                  Scroll ===> CSR
******************************************************** Top of Data ***************************************************************
Processing Trace Settings File: /wasv85config/z9cell/z9dmnode/DeploymentManager/profiles/default/config/cells/z9cell/nodes/z9dmnode/
Trace: 2013/07/01 13:47:09.608 02 t=9E3AE8 c=UNK key=P8 tag= (13007004)
  SourceId: com.ibm.ejs.ras.ManagerAdmin
  ExtendedMessage: BBOO0222I: TRAS0028I: The trace output is stored in the circular memory buffer, holding 8192 message objects.
Trace: 2013/07/01 13:47:09.618 02 t=9E3AE8 c=UNK key=P8 tag= (13007004)
  SourceId: com.ibm.ejs.ras.ManagerAdmin
  ExtendedMessage: BBOO0222I: TRAS0018I: The trace state has changed. The new trace state is *=info.
Trace: 2013/07/01 13:47:09.619 02 t=9E3AE8 c=UNK key=P8 tag= (13007004)
  SourceId: com.ibm.e
  ExtendedMessage: BB                          e output is stored in the circular memory buffer, holding 8192 message objects.
Trace: 2013/07/01 13:                          key=P8 tag= (13007004)
  SourceId: com.ibm.f                          ngService
  ExtendedMessage: BB                          vider Installed: com.ibm.ffdc.util.provider.FfdcOnDirProvider@395aa0a3
Trace: 2013/07/01 13                           key=P8 tag= (13007004)
  SourceId: com.ibm.w
  ExtendedMessage: BBOO0222I: WSVR0800I: Initializing core configuration models
Trace: 2013/07/01 13:47:09.992 02 t=9E3AE8 c=UNK key=P8 tag= (13007004)
  SourceId: com.ibm.ws.runtime.provisioning.ComponentMetaDataMgr
  ExtendedMessage: BBOO0222I: WSVR0179I: The runtime provisioning feature is disabled. All components will be started.
Trace: 2013/07/01 13:47:10.027 02 t=9E3AE8 c=UNK key=P8 tag= (13007004)
  SourceId: com.ibm.ws390.orb.CommonBridge
  ExtendedMessage: BBOJ0011I: JVM Build is JRE 1.6.0 IBM J9 2.6 z/OS s390x-64 20130204_137148 (JIT enabled, AOT enabled)
J9VM - R26_Java626_SR5_20130204_0851_B137148
JIT  - r11.b03_20130131_32403
GC   - R26_Java626_SR5_20130204_0851_B137148
J9CL - 20130204_137148.
```