IBM Software Group

WebSphere Administration

A practical guide to WebSphere scripted administration options

Andrew Simms, Consulting IT Specialist
IBM SWG WebSphere Services
simmsa@uk.ibm.com

e-business on demand software

WebSphere UK User Group - October 2004

WebSphere Administration: What we'll cover

What we're going to do . . .

- Talk through and demonstrate ***configuration*** scripts that:
  - set the initial heap and maximum heap size
  - add or change environment entries
  - set the maximum requests per keep-alive value of a particular transport
  - display the changed values
- Talk through and demonstrate ***application deployment*** scripts.
- From these hope to see the relative merits of each tool

Tool 1: The Admin Console

- You could avoid writing scripts by using the Admin Console to perform configuration tasks and install applications
- If you like typing and don't make mistakes, this is the tool for you
- Going to:
  - select server1
  - navigate to Process Definition + Java Virtual Machine and change heap sizes
  - navigate to Process Definition + Environment Entries and change envvars
  - navigate to Web Container + HTTP Transports, select the SSL-disabled transport, select Custom Properties and add a new property
  - install an application
- Similar navigation on v4
- http://ajsnode1:9090/admin

Admin Console: Comments

- Different look and feel in v4 and v5
- v5.1 & v6 have same look and feel as v5
- Use for ad hoc configuration and operational tasks, e.g.:
  - building a playpen environment
  - examining transactions
  - examining product information
  - controlling trace and examining logs
- Avoid for bulk and repetitive tasks
- Useful early in the test cycle
- Use when some task is just too difficult or expensive to script

So why not just use the Admin Console?

- Consider:
  - you have N_1 domains/cells
  - N_2 test environments
  - N_3 application servers or server groups (clusters) in each domain/cell
  - N_4 enterprise applications
  - N_5 drops received from development for each app
- If $N_i \approx 1$ for each i, then you could use the Admin Console
- When $N_i \gg 1$ then using Admin Console becomes tedious and error prone
- Solution: script domain/cell builds and enterprise application installations
- Up front cost but in the long term you save administrator costs and provide a more reliable service.

Tool 2: wscp

- What is it?
  - v4's configuration and runtime scripting language
- Main features:
  - Extension of Tcl, the Tool Command Language
    - Tcl is portable across many platforms
    - Tcl widely used for scripting tasks, not just WebSphere
    - Simple to learn, powerful to use
  - Interactive or scripted access
  - Typically used for bulk and repeatable tasks, e.g.:
    - Add a set of definitions in a repeatable manner
    - Change the value of a system property on all servers in all domains (but not easily)
  - Can do most things that can be done in the Admin Console

wscp: Style of usage

- Command line /script
- Three modes of operation:
  - interactive
  - single command (-c option)
  - run a script (-f option)
- Access resources using <class> <verb> <object> <options>, e.g.:
  - `J2CResourceAdapter create <name> -attribute {{Name fred} {ArchiveFile freda}}`
- Names look like this:
  - `/Node:ajsnode/Server:server1/`
  - `/JDBCDriver:schumacher/DataSource:button/`
- Lots of curly brackets!!, e.g.:
  - `ServerGroup show <name> -attribute EJBServerAttributes`
  - `ServerGroup modify <name> -attribute {{EJBServerAttributes {WebContainerConfig {SessionManagerConfig {EnableUrlRewriting true}}}}}`

wscp: Example Code

- Code:
  - sample_wscp.tcl (about 90 lines)
  - sample_wscp_install.tcl (about 20 lines)
- Reset before running:
  - sample_swan_reset_wscp.bat
- Run it:
  - sample_wscp.bat
  - sample_wscp_install.bat

Tool 3: XMLConfig

- What is it?
  - v4's XML configuration utility
- Main features:
  - Private to WebSphere, unlike Tcl
  - Exports full or partial repository to an XML file
  - Imports from XML file with arbitrary substitutions
  - Not interactive (but can be called interactively from wscp)
  - Can create, update or delete objects
  - Can start/stop objects
  - Typically used for repeatable tasks, e.g.:
    - Add a set of definitions in a repeatable manner
  - Can be invoked from Admin Console, shell or wscp

XMLConfig: Style of Usage

- Command
- Run `xmlconfig -export` to export the entire domain to an XML file
- Run `xmlconfig -export -partial <file>` to export part of a domain to an XML file, where `<file>` directs what is to be exported
- Edit the generated XML file:
  - remove unwanted bits
  - change values to variable names to allow substitution
- Run `xmlconfig -import` to import from the XML file to a new or the same domain, usually with substitutions
- Can also build the XML file by hand

XMLConfig: Example Code

- Code:
  - sample_xmlconfig_partial_export.xml (7 lines)
  - sample_xmlconfig_partial_export_output.xml (c200 lines, generated)
  - sample_xmlconfig_import.xml (50 lines, edited)
  - sample_xmlconfig_partial_export_install.xml (5 lines)
  - sample_xmlconfig_partial_export_output_install.xml (c70 lines, generated & edited)
- Reset before running:
  - sample_swan_reset_wscp.bat
- Run it:
  - sample_xmlconfig.bat
  - sample_xmlconfig_install.bat

wscp is better than XMLConfig is better than wscp

wscp	XMLConfig
Is a powerful procedural scripting *language* with variables	Is a *utility* that allows simple substitutions and has limited actions
Is based on a Java implementation of a popular scripting language (Tcl)	Its output is XML, but the utility is unique to WebSphere
Represents WebSphere objects as commands (e.g. `ApplicationServer`) with verbs (e.g. "start", "modify")	Places actions in XML attributes (rather like ANT)
Is a natural choice for doing anything in bulk or repetitive	Is a good choice for repetitive tasks where changes are simple substitutions
Is not dependent on having something built manually first	Is dependent on creating something first manually (unless you create the XML by hand)
Can be used for ad hoc purposes interactively (but not always easily)	Export+partial can do some ad hoc tasks

Scripted options in v5 & v5.1 & v6

- Gone!
- Replacement for wscp: wsadmin
  - But provides a completely different set of objects
  - wscp scripts are not migratable to wsadmin
- Replacement for XMLConfig: nothing
  - Repository now a set of XML files
  - Expectation that customers would copy/edit directories/files
  - Proved too hard
  - Some as-is scripts available on WebSphere Developer Domain for doing XMLConfig-like things (v5.1.2)
  - Some more XMLConfig-like things in v6 and beyond

Tool 4: wsadmin

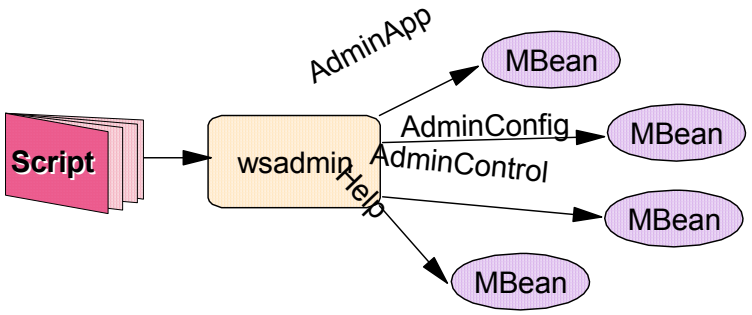
- What is it?
  - v5's scripting language
- Main features:
  - Uses the Bean Scripting Framework (BSF)
    - Provides access to Java objects and methods from supported scripting languages.
    - Architecture for easily incorporating scripting into Java applications and applets
    - Applications independent and not bound to a single scripting language
    - Different language scripts can access Java objects using wsadmin
  - Current supported languages for wsadmin:
    - Jacl - Java Command Language based on Tcl scripting
    - Jython – Java implementation of the OO language Python

Tool 4: wsadmin

- How is it different from wscp?
  - The WebSphere objects are completely different from wscp
  - Tcl is easy to learn, but wsadmin is complex
  - Separates static configuration from dynamic changes
- Typically used for bulk and repeatable tasks, e.g.:
  - Add a set of definitions in a repeatable manner
  - Change the value of an environment variable on all servers in all cells
- Can do everything in the Admin Console, but not in a necessarily intuitive manner

wsadmin: JMX MBeans

- wsadmin acts as an interface to Java objects for access by scripts
  - Objects communicate with MBeans (JMX management objects)
- Objects perform different operations. There are four built-in objects:
  - `$AdminConfig` (Jacl) or `AdminConfig` (Jython)
    - Create or change the WebSphere configuration
  - `$AdminApp` (Jacl) or `AdminApp` (Jython)
    - Install, modify, or administer applications
  - `$AdminControl` (Jacl) or `AdminControl` (Jython)
    - Work with live running objects and perform traces and data type conversion
  - `$Help` (Jacl) or `Help` (Jython)
    - Display general help information and details about which MBeans are running
- Separation between Configuration and Control



wsadmin: Style of usage

- Command line / script
- Three modes of operation:
  - interactive
  - single command (-c option)
  - run a script (-f option)
- Connect over SOAP or RMI
- Pass language identifier to wsadmin command – Jacl assumed
- Access configuration resources using AdminConfig. In Jacl:
  - `set id [$AdminConfig getid <name>]`
  - `$AdminConfig modify $id {{name fred} {desc "fred bloggs"}}`
- And in Jython:
  - `id = AdminConfig.getid(<name>)`
  - `AdminConfig.modify(id, "[[name fred], [desc ""fred bloggs""]]"`
- Configuration names look like this:
  - `/Cell:ajs_cell/Node:ajsnode/Server:server1/`
  - `/JDBCProvider:schumacher/DataSource:button/`

wsadmin: Style of usage (2)

- Access runtime resources using `AdminControl`. In Jacl:
  - `set mbeanId [$AdminConfig getObjectName $id]`
  - `$AdminControl invoke $mbeanId getState`
- And in Jython:
  - `mbeanId = AdminConfig.getObjectName($id)`
  - `AdminControl.invoke(mbeanId, getState)`
- Runtime names look like this:
  - `WebSphere:platform=common,cell=ajs_cell,version=5.0.1,name=NodeAgent,mbeanIdentifier=NodeAgent,type=NodeAgent,node=ajsnode1,process=nodeagent`

wsadmin: Some Operations and Functions

- $AdminConfig
  - attributes
  - convertToCluster
  - create
  - createClusterMember
  - createUsingTemplate
  - defaults
  - getid
  - getObjectName
  - installResourceAdapter
  - list
  - listTemplates
  - modify
  - parents
  - reset
  - save
  - show
  - showall
  - showAttribute
  - types
- $AdminApp
  - deleteUserAndGroupEntries
  - edit
  - editInteractive
  - export
  - exportDDL
  - install
  - installInteractive
  - list
  - listModules
  - options
  - publishWSDL
  - taskInfo
  - uninstall
- $AdminControl
  - completeObjectName
  - getAttribute
  - getAttributes
  - getCell
  - getConfigId
  - getHost
  - getMBeanCount
  - getNode
  - getPort
  - getType
  - invoke
  - makeObjectName
  - queryNames
  - setAttribute
  - setAttributes
  - startServer
  - stopServer
  - testConnection
  - trace
- $Help
  - attributes
  - operations
  - constructors
  - description
  - classname
  - all
  - AdminControl
  - AdminApp
  - AdminConfig
  - wsadmin
  - message

wsadmin: Miscellaneous

- Commands are case-sensitive
- Configuration changes not persisted until "save" call
- If multiple scripts or clients (Administrative Console) are saving configuration changes at the same time, exception thrown on Save call
  - No changes will be persisted
- Upon "save" call, validation procedure verifies updates
  - Create two servers with the same name throws exception
- wsadmin -f "xxxxx" much faster than wsadmin -c "xxxxx"
  - Better to run multiple commands in a file than individual commands
  - Call "save" in file periodically to persist configurations updates
    - Avoids no changes being persisted should an exception occur

wsadmin: Installing an Application

- Use `$AdminApp`
- Can use interactively inside a command
- Can get the configurable options from an EAR file
  - `$AdminApp options myapp.ear`
- Then can get the task information for each option:
  - `$AdminApp taskInfo myapp.ear MapWebModToVH`
- Could then use this info to help build the relevant options for installation:
  - `$AdminApp install myapp.ear {-appname freda -MapWebModToVH {{{Default Application} default_app.war,WEB-INF/web.xml myvhost}} -<lots of other options . . .>}`
- Just as difficult to construct this as in wscp

wsadmin: Example Code

- JACL Code:
  - sample_wsadmin.tcl (about 90 lines)
  - sample_wsadmin_install.tcl (about 25 lines)
- Jython Code:
  - sample_wsadmin.py (about 90 lines)
- Reset before running:
  - sample_swan_reset_wsadmin.bat
- Run JACL:
  - sample_wsadmin_jacl.bat
  - sample_wsadmin_install_jacl.bat
- Reset before running:
  - sample_swan_reset_wsadmin.bat
- Run Jython:
  - sample_wsadmin_py.bat

Jacl is better than Jython is better than Jacl

- Not that different
- If you can program in one you can probably program in the other
- Jacl may suit shell script programmers
- Jython may suit those with OO background
- Both take just as much as code as the other in our example

Tool 5: ws_ant (ANT)

- What is it?
  - ANT – Java-based, platform-neutral, open source build tool
  - ws_ant: Copy of Apache ANT with task extensions to support WebSphere-specific capabilities
- Main Features:
  - Tasks provided include:
    - Install and uninstall applications
    - Start and stop servers
    - Run administration scripts or commands
  - But the admin task is just a call to wsadmin – you still have to write the wsadmin code to do the actual configuration
  - Main advantage is ability to build the EAR, run a configuration script and then install the EAR, all from a single ws_ant build.xml.

ws_ant: Style of Usage

- Command-oriented
- Specify tasks, targets and dependencies between targets
- Connect over SOAP or RMI
- Pass language identifier to wsadmin command – Jacl assumed

ws_ant: Example Code

- Code:
  - build.xml (45 lines)
  - sample.props (20 lines)
- Reset:
  - sample_swan_reset_wsadmin.bat
- Run it:
  - sample_wsant_wsadmin.bat
  - sample_wsant_install.bat

Tool 6: WsAdmin Automation Platform (WAP)

- What is it?
  - Set of utility functions sitting over wsadmin
  - Eases a lot of wsadmin hassle
- Main features:
  - Available from IBM Techdocs, the Technical Sales Library
  - Not supported by IBM: provided "as is"
  - Greatly simplifies many wsadmin scripting tasks
  - Provides a library of Tcl procedures for creating, modifying and deleting WebSphere objects (e.g. application servers, data sources)
  - Runs on 5.0 and 5.1
  - Syntax (on Windows):
    - `set WAP_SOURCE=/installed/wap`
    - `set PATH=%PATH%;%WAP_SOURCE%`
    - `jwap.bat`

WAP: Style of usage

- Command line / script
- Runs over wsadmin, so:
  - Three modes of operation:
    - interactive
    - single command (-c option)
    - run a script (-f option)
  - Connect over SOAP or RMI
- Jacl only (not Jython)
- Set subsequent commands' scope:
  - `setNode ajsnode1`
  - `setServer server1`
- Provides a number of Jacl procedures named:
  - `createXXX, removeXXX, modifyXXX`
  - e.g. `createCustomService, modifyJDBCProvider`
- Pass parameters as dash-options, plus "you-name-it" options

WAP: Style of usage (2)

- Names become simpler (no need for `/Cell:xxx/Node:xxx/Server:xxx/`)
- Not all documented:
  - `getIDByListing` & `modifyConfigObject` are not documented but very useful
- Mainly aimed at create/modify/delete
  - little support for display & list
  - "list" does not return as Tcl lists so you can't do "foreach" style processing

WAP: Example Code

- Code:
  - sample_wap.tcl (c90 lines)
- Reset:
  - sample_swan_reset_wsadmin.bat
- Run it:
  - sample_wap.bat

WAP: Installing an application

- Use the `installApplication` command
- Pass customisations in same way as wsadmin's `$AdminApp` install:

```
installApplication -appname fred -earfile
   /installed/websphere/appserver/base5.1/installableApps/ivtApp.ear \
   {-MapModulesToServers {{{IVT Application} ivt_app.war,WEB-
   INF/web.xml WebSphere:cell=ajs_cell,node=ajsnode1,server=server1}}}
```

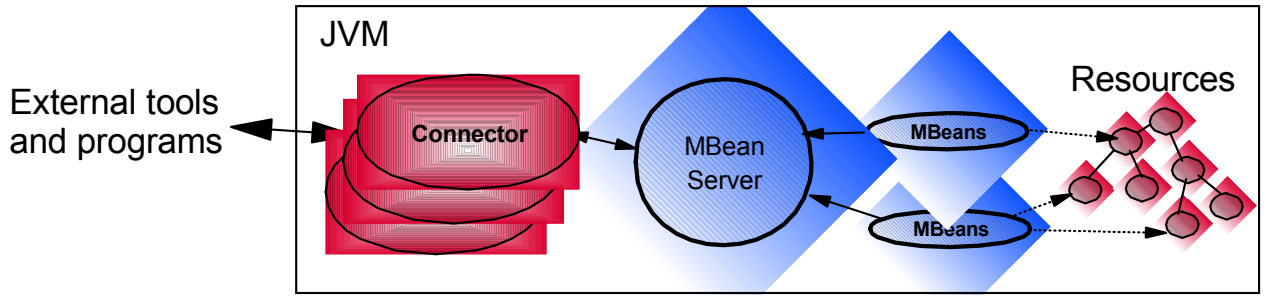
- `-interact` option to customise by answering questions (couldn't get this to work)
- Can install then customise afterwards using the `editApplication` command
- Not really any different from `$AdminApp`
- About 15 long statements in our example

Tool 7: JMX

- What is it?
  - Java Management Extensions (JMX) is a framework oriented at managing applications and resources (like application servers)
    - Allows for exposing your application to remote or local management tools
    - JMX is a specification oriented to the application and network management
    - Part of J2SE
- Main Features:
  - Some of the JMX functionality:
    - Allows you to query the configuration settings and change them at runtime
    - Provides services such as monitoring, event notification, timers, ...
    - Allows you to load, initialize, change, and monitor application components and resources
  - Structured in three layers:
    - Instrumentation layer - MBeans
    - Agent layer - MBeanServer and agents
    - Distributed Services (this part is still out of the scope of the current level of the specs)
      JSR-077

About JMX: How does JMX work?

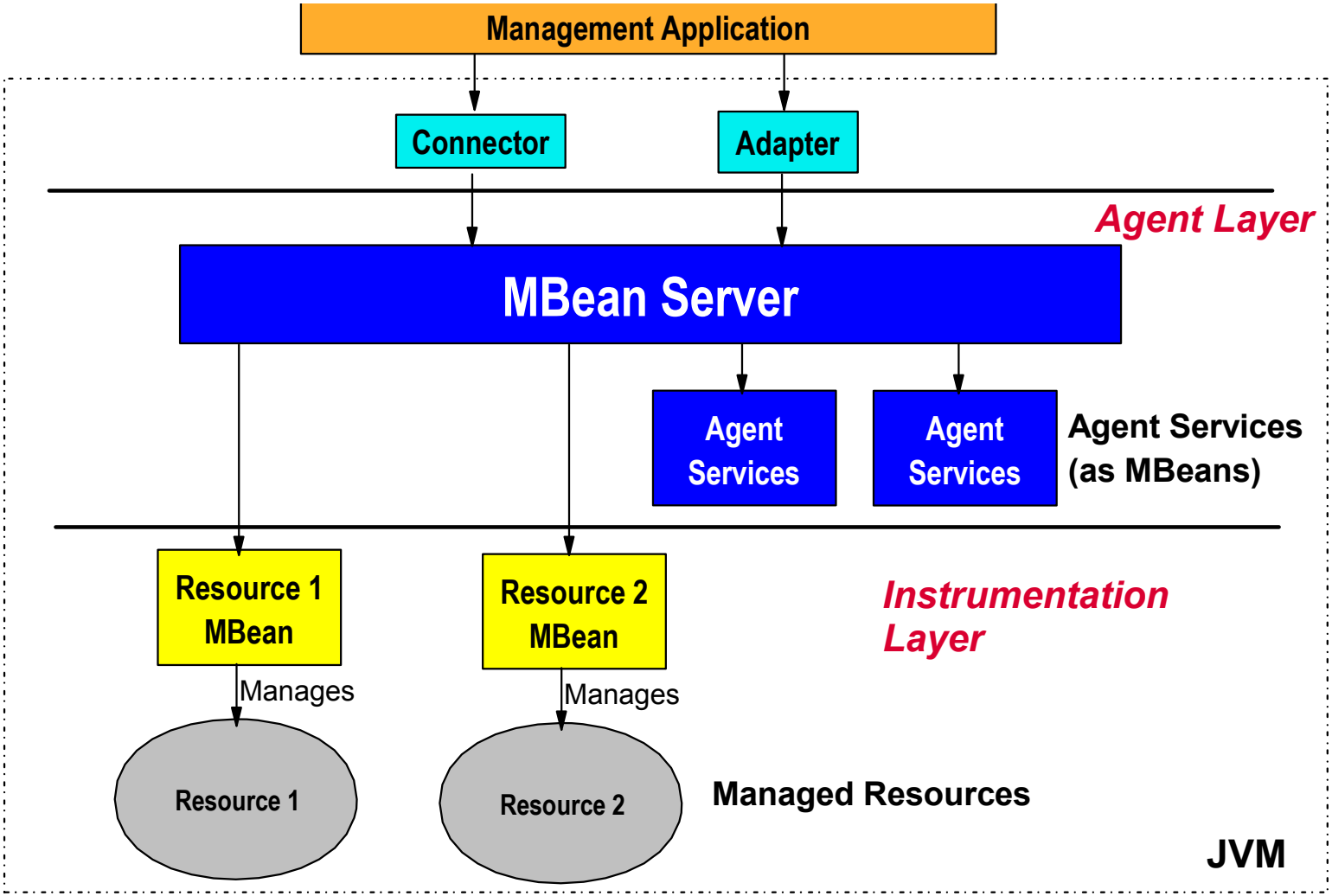
- Resources are managed by Managed Beans (MBeans)
  - These are simple JavaBeans that perform operational or configuration changes on resources
- MBeans provide APIs to the external world to manage its resources
  - Each JMX-enabled JVM contains a MBean Server (Managed Bean Server) that registers all the MBeans in the system
    - MBean Server provides access to all of its registered MBeans
  - External programs access MBeans registered on the MBean Server via Connectors/Adapters



JMX Benefits

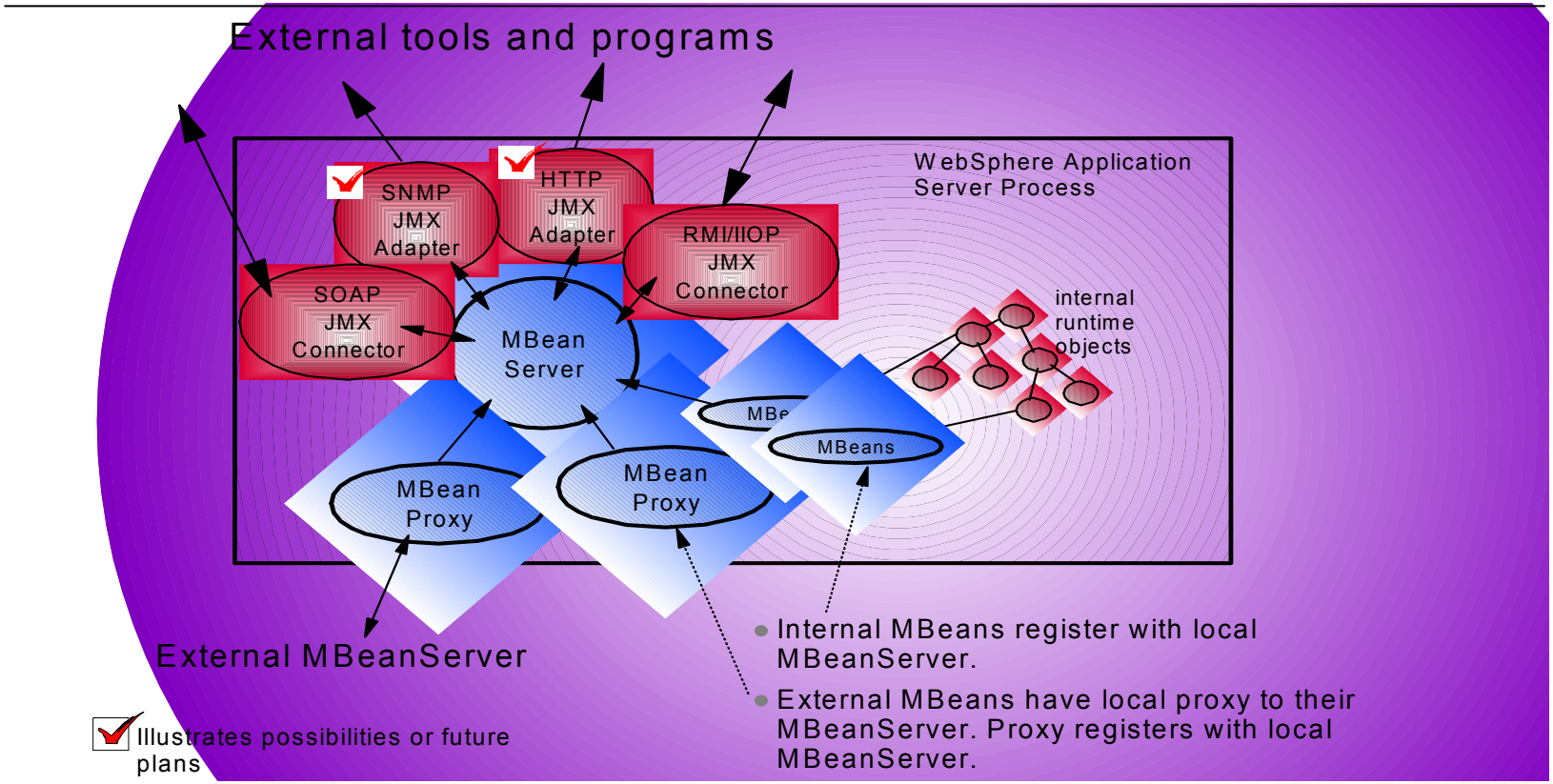
- Enables Java applications to be managed without heavy investment
  - Relies on a core managed object server that act as a 'management agent'
  - Java application simply needs to embed a managed object server and make some of its functionality available as one or several Manageable Beans registered in the object server
- Provides a scalable management architecture
  - Every JMX agent service is an independent module that can be plugged into the management agent
- Integrates existing management solutions
  - JMX smart agents are capable of being managed through HTML browsers or by various management protocols such as Web Services, JMS, SNMP, etc.
- Defines only the interfaces necessary for management

JMX: Architectural Principles

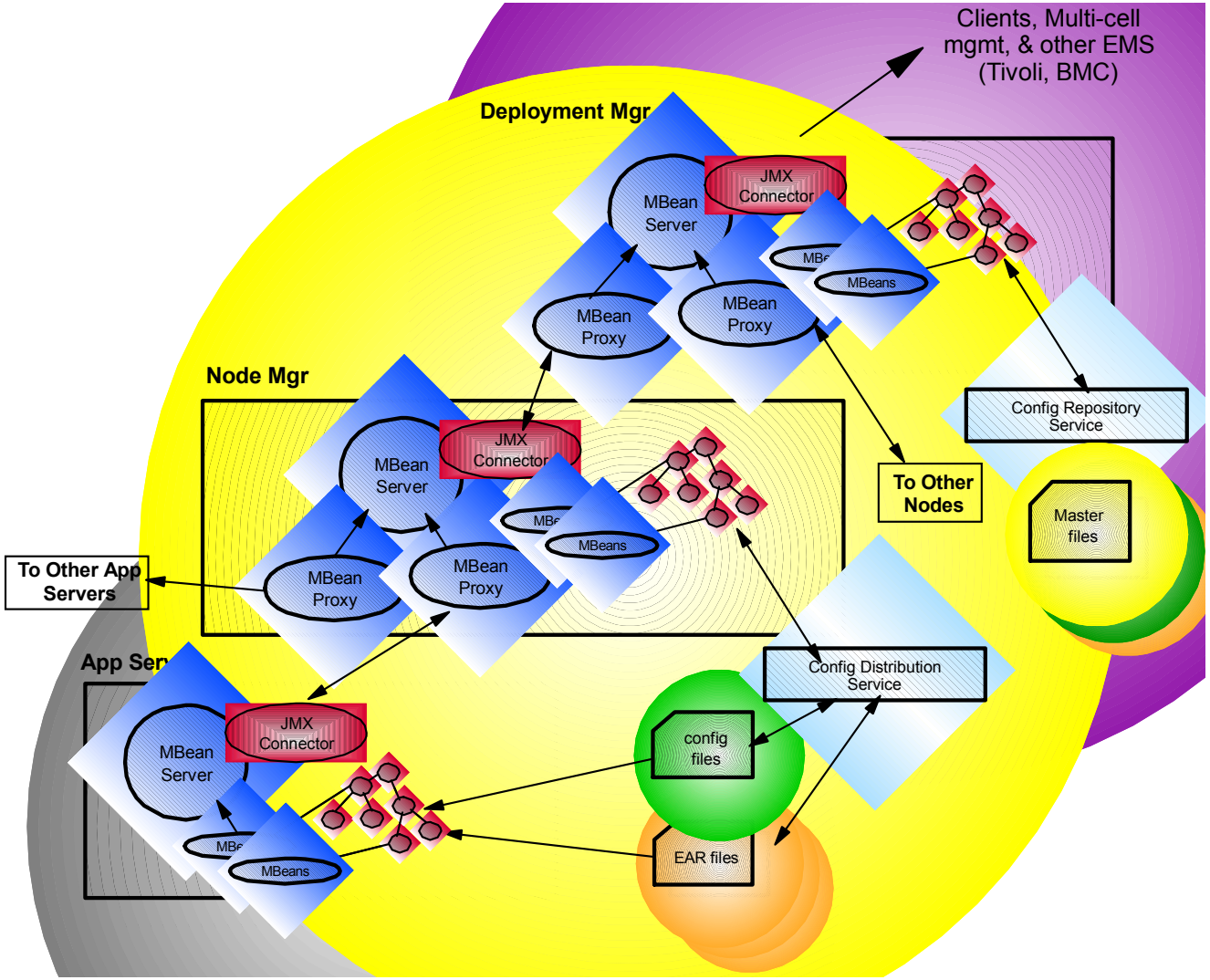


WebSphere JMX Architecture

- All WebSphere 5.0 processes run the JMX agent
- All WebSphere 5.0 runtime administration is done through JMX operations



WebSphere Distributed Administration



JMX: Some MBeans

- Cell
- Node
- ManagedObject
- ApplicationServer
- ServerCluster
- JMSServer
- Application, EJBModuleConfig, WebModuleConfig
- EJBContainer, WebContainer
- ListenerPort
- JMSProvider
- JMSConnectionFactory
- JMSDestination
- J2CResourceAdapter
- J2CConnectionFactory
- JavaMailProvider
- MailSession
- URLProvider
- URL
- JDBCProvider
- DataSource, WAS40DataSource
- ConnectionPool
- ORBPlugin
- JavaVirtualMachine
- SecurityPermission
- NameServer
- LocalOSUserRegistry

These Mbeans are built into WebSphere – 224 of them in v5.1

How Can Customers Use JMX?

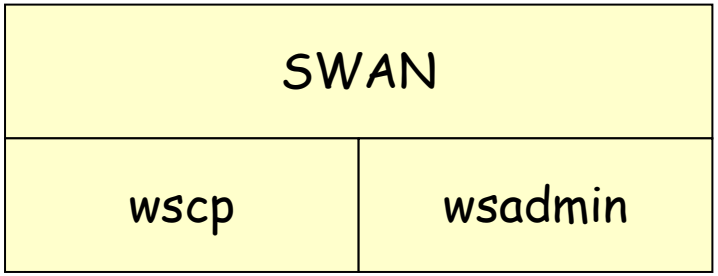
- All WAS 5.0 Admin client programs use JMX
  - Web Admin console
  - wsadmin scripting
  - Admin Client Java API
  - Included in both Base App Server package and Advanced Deployment
- Use the APIs to control WebSphere runtime
- Extend the set of managed objects with custom JMX MBeans
- Attach Admin clients to Cell Manager in order to access entire Admin domain
- Node Agents start & monitor individual App Servers

JMX: Example Code

- Code:
  - sample_jmx.java (c160 lines)
- Reset:
  - sample_swan_reset_wsadmin.bat
- Run it:
  - sample_jmx.bat

Tool 8: SWAN

- What is it?
  - Simplified WebSphere AdministratioN
  - Greatly eases wscp and wsadmin hassle
- Main features:
  - Runs in wscp and wsadmin
  - Obtainable from the author
  - Runs on WebSphere v4, v5, v5.1
  - Runs on Windows, AIX (probably all UNIXes) and z/OS
  - Simplifies writing wscp and wsadmin scripts
  - Simplifies interactive access
  - Provides a high degree of portability between WebSphere versions



SWAN Syntax

- <class> <verb> [<object>] [<options>] [<attributes>]
- Examples:
  - appserv list
  - dsource list –format -short
  - jcafact delete jack –ra jill
  - resadapt create ada –rarfile /usr/product/somefile.rar
  - cluster show clouseau +maxheap +minheap
  - appserv show jeeves +webhttpbacklog –constraint {Host * Port 9080}
  - appserv modify jeeves +webhttpbacklog 53 –constraint {Host * Port 9080}
  - appserv modify –constraint {Host * Port 9080} +webhttpbacklog 57 jeeves
  - jmsdest exists jack –jmsprov jill

SWAN is better than wscp/wsadmin is better than SWAN

- In wscp/wsadmin you have to know how to reach a particular attribute
  - Not a problem for many cases
    - e.g. JDBCProvider has no attribute hierarchy
  - Big problem for server groups and application servers
    - have to understand the hierarchy to navigate through it
  - SWAN solves this by hiding the hierarchy
- In wscp/wsadmin you have long names:
  - Long names for configuration and runtime objects
  - Much of which is probably constant in your script (e.g. cell name & node name)
  - SWAN solves this by setting a scope and providing the –short option
- In wscp/wsadmin you need long Tcl lists to set attributes:
  - SWAN solves this because it has flattened the attribute hierarchy
- wscp scripts cannot be migrated to wsadmin without complete rewriting:
  - But wscp scripts written to use the SWAN syntax will be migratable
  - Some changes will be needed though because of differences in v5

SWAN is better than wscp/wsadmin is better than SWAN

- SWAN does not implement all of the object types:
  - There are 224 in wsadmin in v5.1
  - SWAN implements the most commonly used object types
  - SWAN can be extended to implement the others – easily if they follow the patterns of the current SWAN types
  - You can always use the wsadmin types directly in your scripts
- SWAN is not supported by IBM:
  - SWAN is provided as-is
  - Whereas wscp and wsadmin are supported

SWAN: Further Examples

- Getting help:
  - help
  - dsource help
  - jdbcdrv help create
  - jcafact attributes
  - jcafact attributes -details
  - cluster create clouseau -member sellers –vertical 3
  - cluster showmembers clouseau
  - appserv makecluster jeeves –cluster wooster
  - cluster start wooster
  - cluster stop wooster -force

SWAN: Coverage

appserv	Application servers
cell	Domain or cell
cluster	Clusters (server groups)
dsource	Data sources
entapp	Enterprise applications
jcafact	J2C connection factories
jdbcdrv	JDBC drivers (providers)
jmsdest	JMS destinations
jmsfact	JMS connection factories
jmsprov	Generic JMS providers
mail	Java mail
nagent	Node agents
resadapt	Resource adapters
vhost	Virtual hosts

Installing enterprise applications using SWAN

- Packager or Deployer runs "entapp produceaid":
  - Creates an XML file called an ***E***nterprise ***Ar***chive ***A***pplication ***I***nstallation ***D***escriptor (EAR AID)
  - EAR AID houses all of the deployable options from the EAR file's deployment descriptors
- Intention is that the Deployer then edits this file using an XML or text editor to set appropriate values (as would be done through the console)
- Deployer then runs "entapp create":
  - takes the EAR file and uses the AID file to install the enterprise app
  - same procedure whether v4 or v5, but different EAR AID file
- Non-SWAN software available to produce an Excel spreadsheet containing the EAR AID file, and to transform it back
  - makes it easier to edit

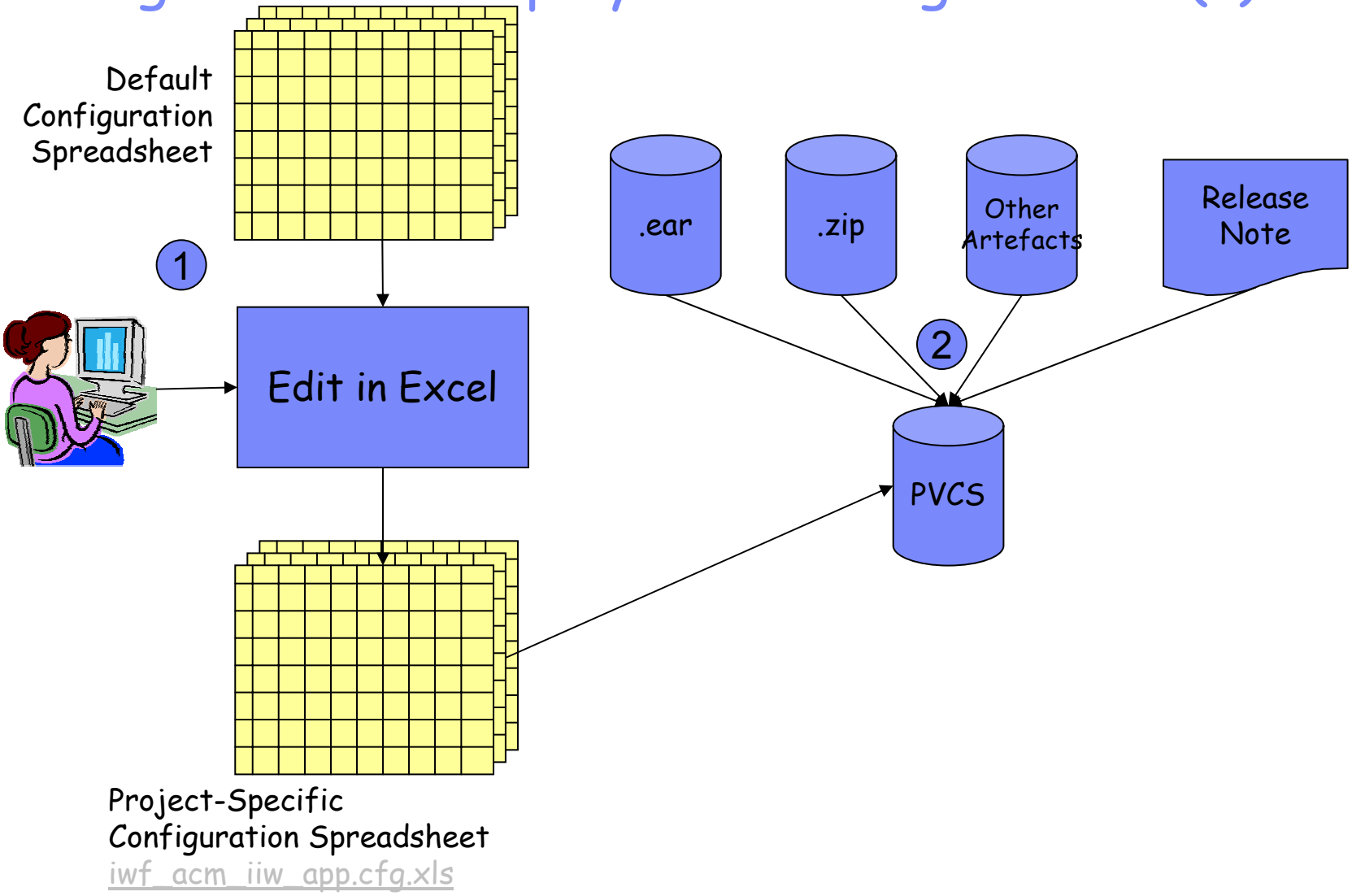
SWAN: Example Code

- Code:
  - sample_swan.tcl (25 lines, does both v4 and v5)
- Reset:
  - sample_swan_reset_wscp.bat
  - sample_swan_reset_wsadmin.bat
- Run it:
  - sample_swan_wscp.bat
  - sample_swan_wsadmin.bat

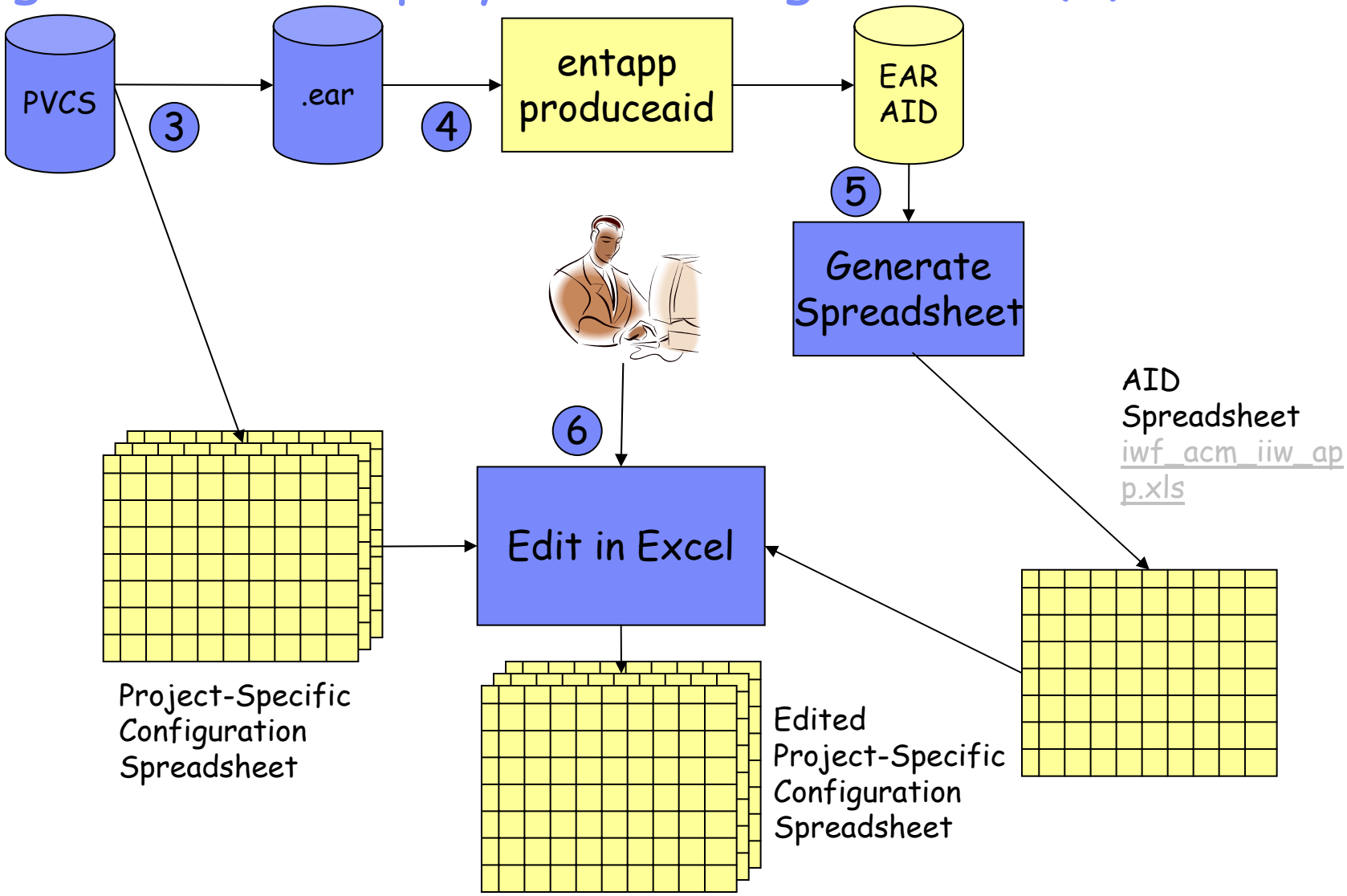
SWAN Internals: Code Size

File Type	# lines non-commentary code
Configuration files	800
v4 code	3700
v5 code	3600
Common v4 code	1400
Common v5 code	900
Other common code	1500
Total non-commentary code	**11,400**
Plus comment lines	4,300
Includes debug/trace code	1000

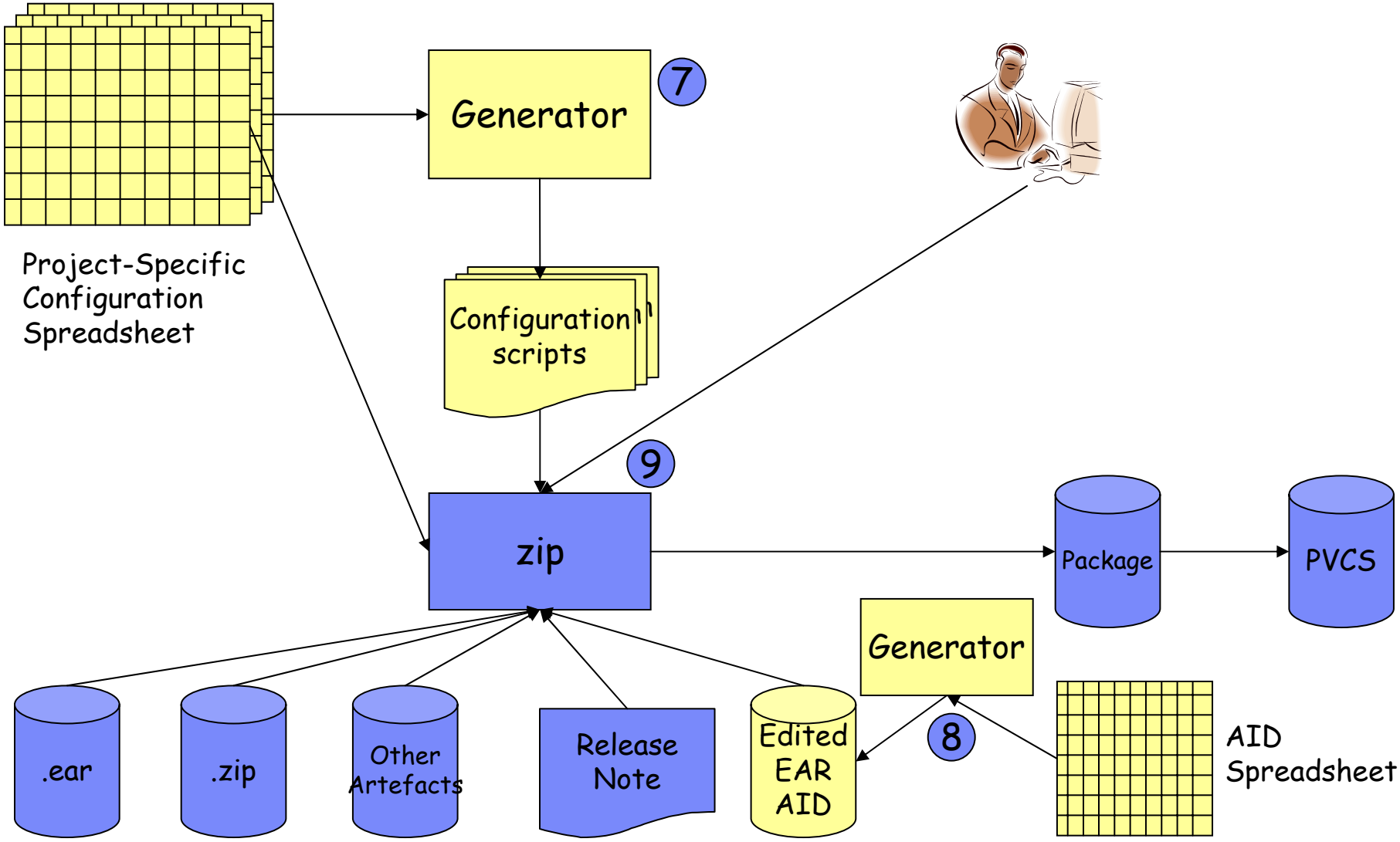
Configuration & Deployment Using SWAN (1)



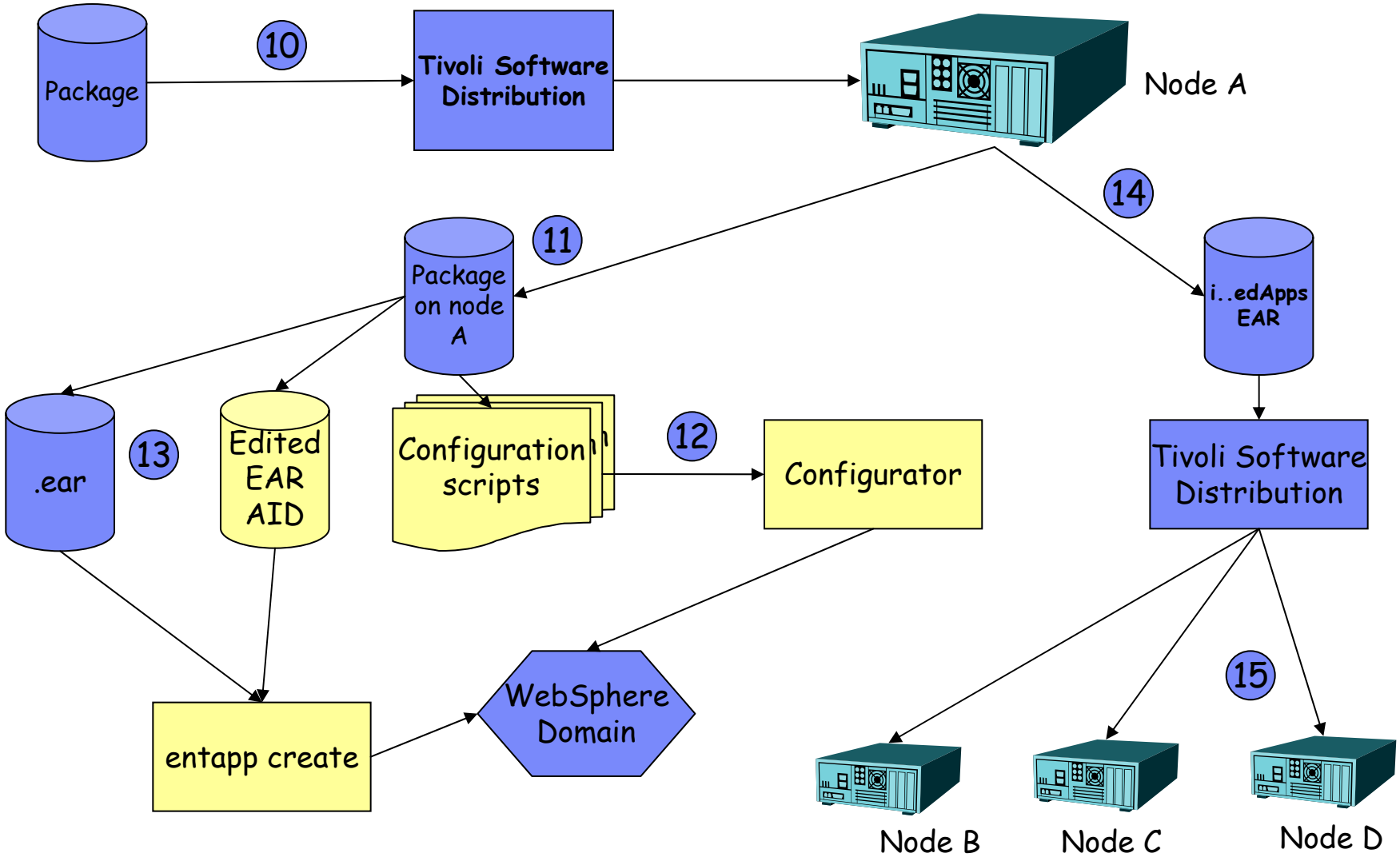
Configuration & Deployment Using SWAN (2)



Configuration & Deployment Using SWAN (3)



Configuration & Deployment Using SWAN (4)



Tool 9: J2EE 1.4 Management and Deployment

- What is it?
  - J2EE 1.4 APIs for product-independent configuration & deployment
- Main features:
  - No standard up to J2EE 1.3
  - J2EE 1.4 (WebSphere v6) introduces:
    - J2EE Management Specification (JSR-77)
      - ability to configure and control J2EE objects in a Java-application-server-independent manner
    - J2EE Deployment Specification (JSR-88)
      - ability to install enterprise applications in a Java-application-server-independent manner
    - wsadmin still works
    - write programs to these specs to lessen vendor lock-in
    - many configuration/deployment possibilities:
      - Admin Console
      - wsadmin scripts in Jacl and Jython
      - Programs written to the JMX API
      - Programs written to JSR-77 & JSR-88

Comparison: Lines of code

	wscp	XmlConfig	wsadmin Jacl	wsadmin Jython	ws_ant	JMX	SWAN
Heap sizes	2	60	4	6	45 (plus wsadmin)	9	2
Environment	16		24	19		35	6
Transport	43		23	?		40	3
Display	30	8 (but displays everything)	30	?		50	6
Installation	20	75	25	?		?	4

Summary

- References
  - "IBM WebSphere System Administration" by Leigh Williamson et al (ISBN: 0-13-144604-5)
  - "IBM WebSphere Deployment and Advanced Configuration" by Barcia/Hines/Alcott/Botzum (ISBN: 0-13-146862-6)
  - WAP:
    - http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS981
  - SWAN:
    - simmsa@uk.ibm.com