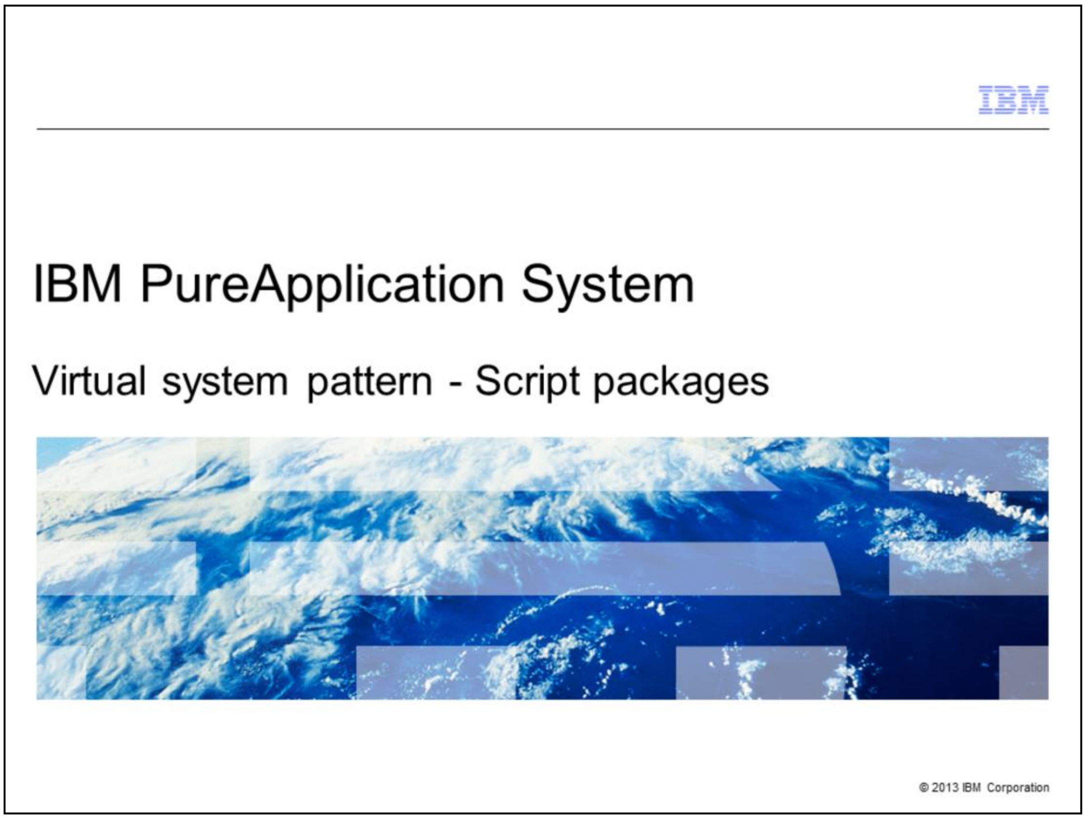
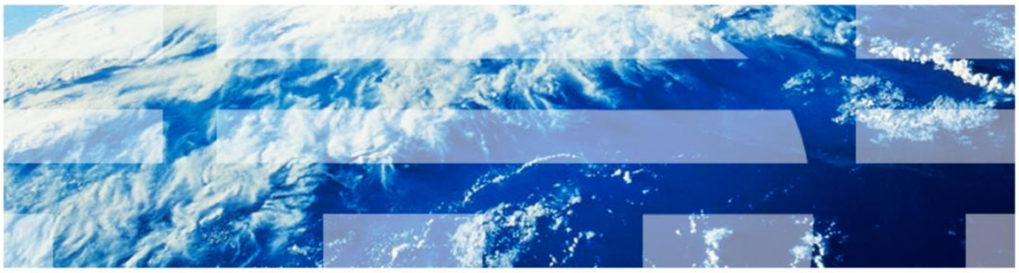


IBM PureApplication System

Virtual system pattern - Script packages



This presentation covers virtual system script packages. Script packages allow you to configure the virtual machine by customizing the operating system or middleware within a deployed virtual machine.

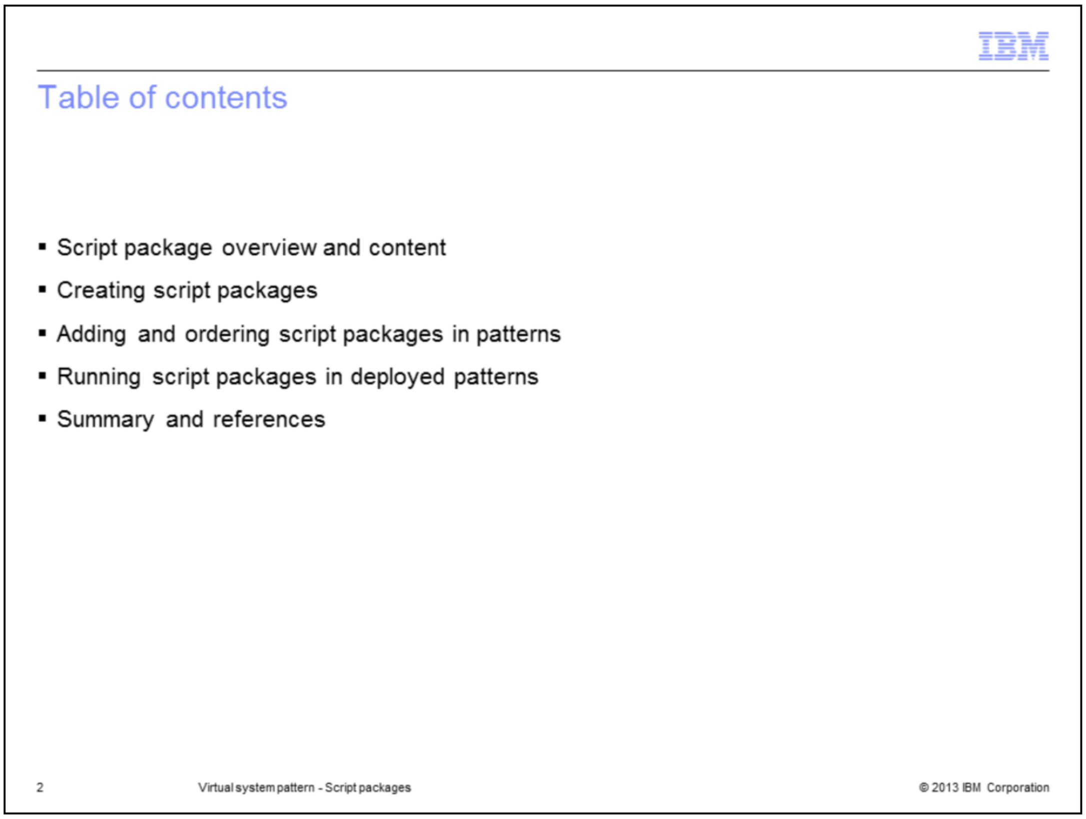
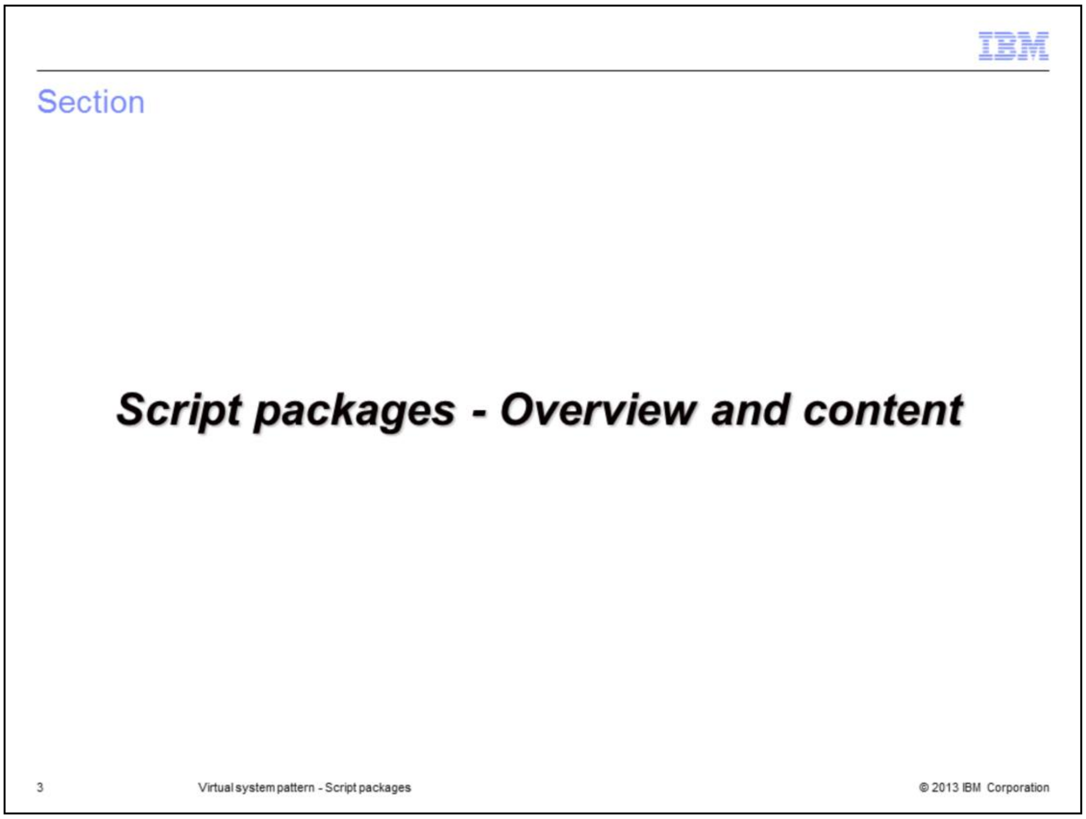


Table of contents

This presentation covers the life cycle of a script package, including its creation, adding and defining the order that the script packages will run, and how the scripts are run on the deployed VMs of the patterns. It will also cover the format of a typical script package.

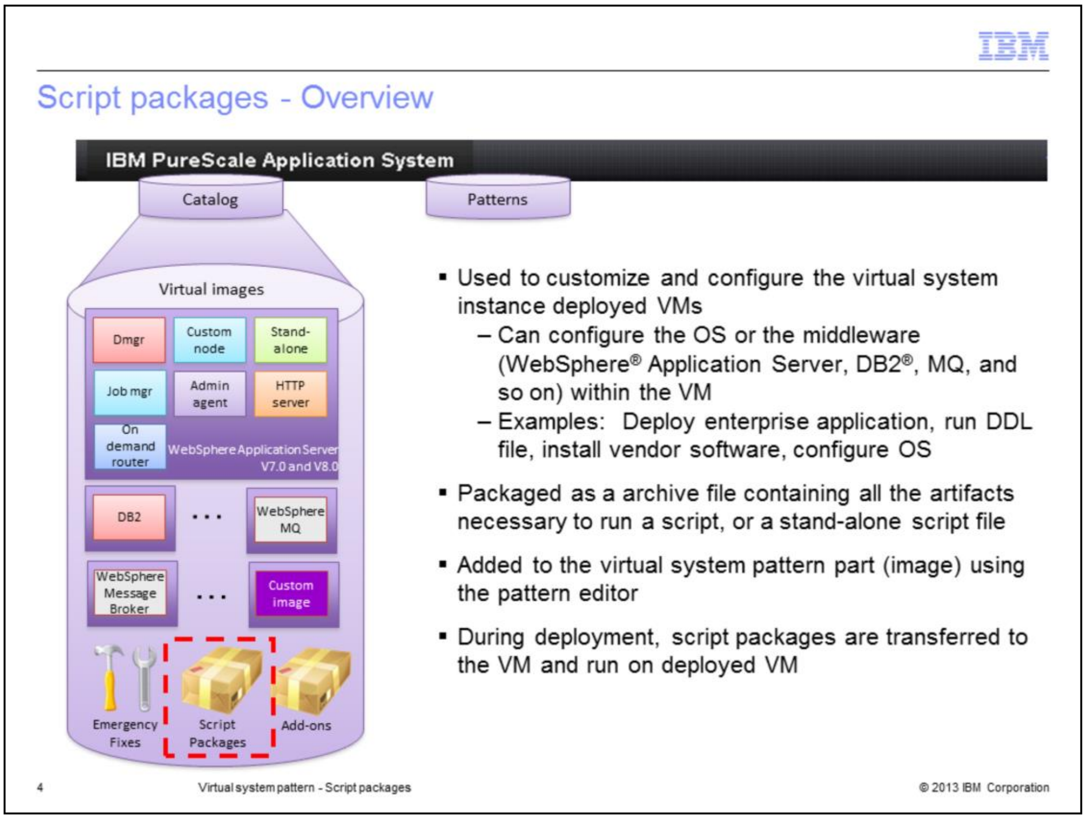


Section

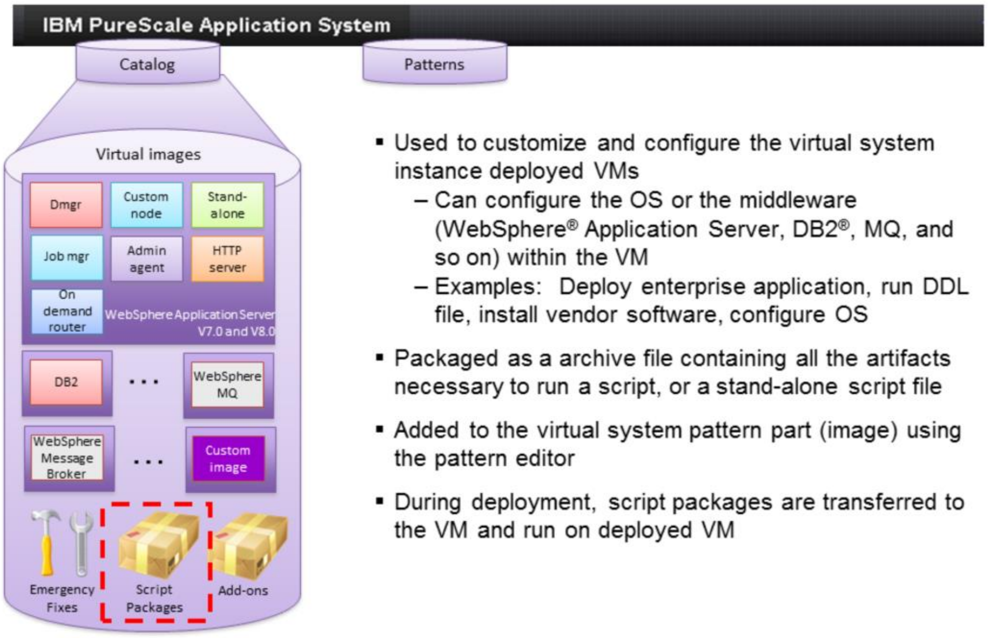
Script packages - Overview and content

3 Virtual system pattern - Script packages © 2013 IBM Corporation

The first section shows you the content of a typical script package.



Script packages - Overview

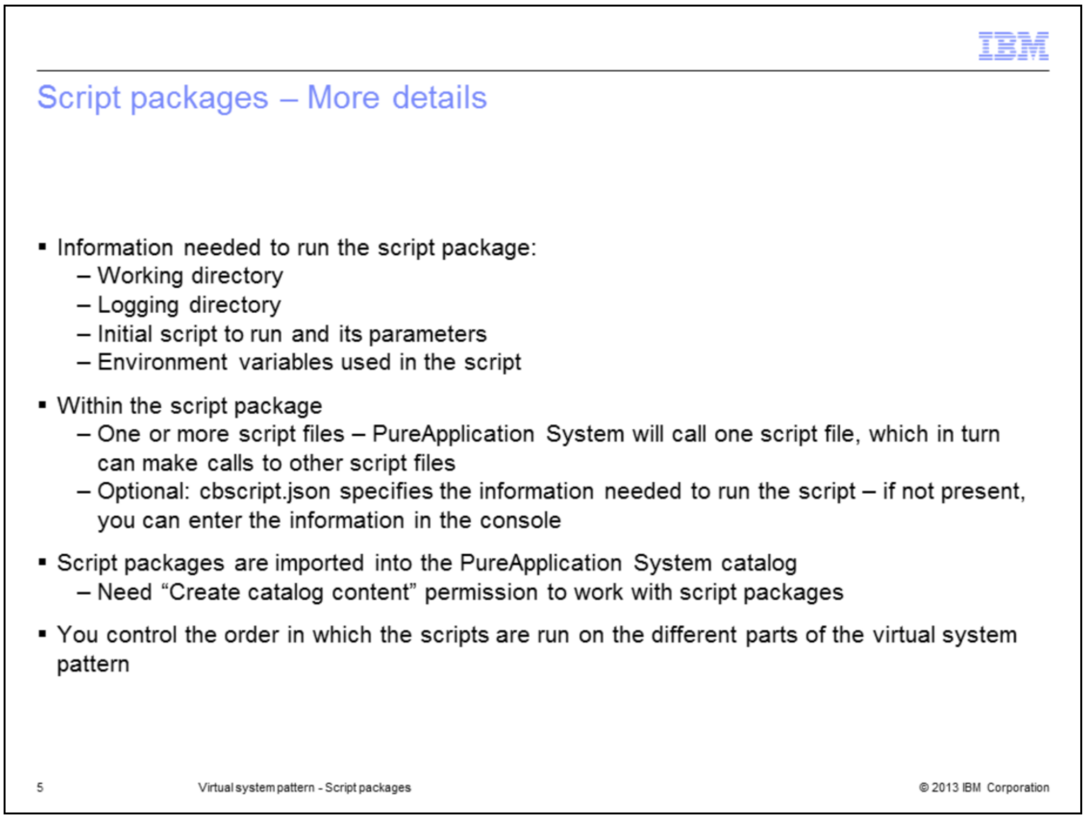


- Used to customize and configure the virtual system instance deployed VMs
  - Can configure the OS or the middleware (WebSphere® Application Server, DB2®, MQ, and so on) within the VM
  - Examples: Deploy enterprise application, run DDL file, install vendor software, configure OS
- Packaged as a archive file containing all the artifacts necessary to run a script, or a stand-alone script file
- Added to the virtual system pattern part (image) using the pattern editor
- During deployment, script packages are transferred to the VM and run on deployed VM

4 Virtual system pattern - Script packages © 2013 IBM Corporation

Script packages are listed in the PureApplication™ System catalog, along with the virtual image parts that they customize. Script packages are very flexible and can do just about anything you want, including running operating system commands or middleware commands like wsadmin for WebSphere Application Server. There are some examples that come pre-loaded in the PureApplication System catalog. However, you will want to write scripts to do customization that is specific to your installation. A script package is an archive file, but can also be a stand-alone script file. The next slide describes the content of a script package archive file.

Script packages are added to the virtual system pattern part, in other words virtual image, within a virtual system pattern, using the virtual system pattern editor. They are run on the deployed virtual machines.



Script packages – More details

- Information needed to run the script package:
  - Working directory
  - Logging directory
  - Initial script to run and its parameters
  - Environment variables used in the script
- Within the script package
  - One or more script files – PureApplication System will call one script file, which in turn can make calls to other script files
  - Optional: cbscript.json specifies the information needed to run the script – if not present, you can enter the information in the console
- Script packages are imported into the PureApplication System catalog
  - Need “Create catalog content” permission to work with script packages
- You control the order in which the scripts are run on the different parts of the virtual system pattern

5 Virtual system pattern - Script packages © 2013 IBM Corporation

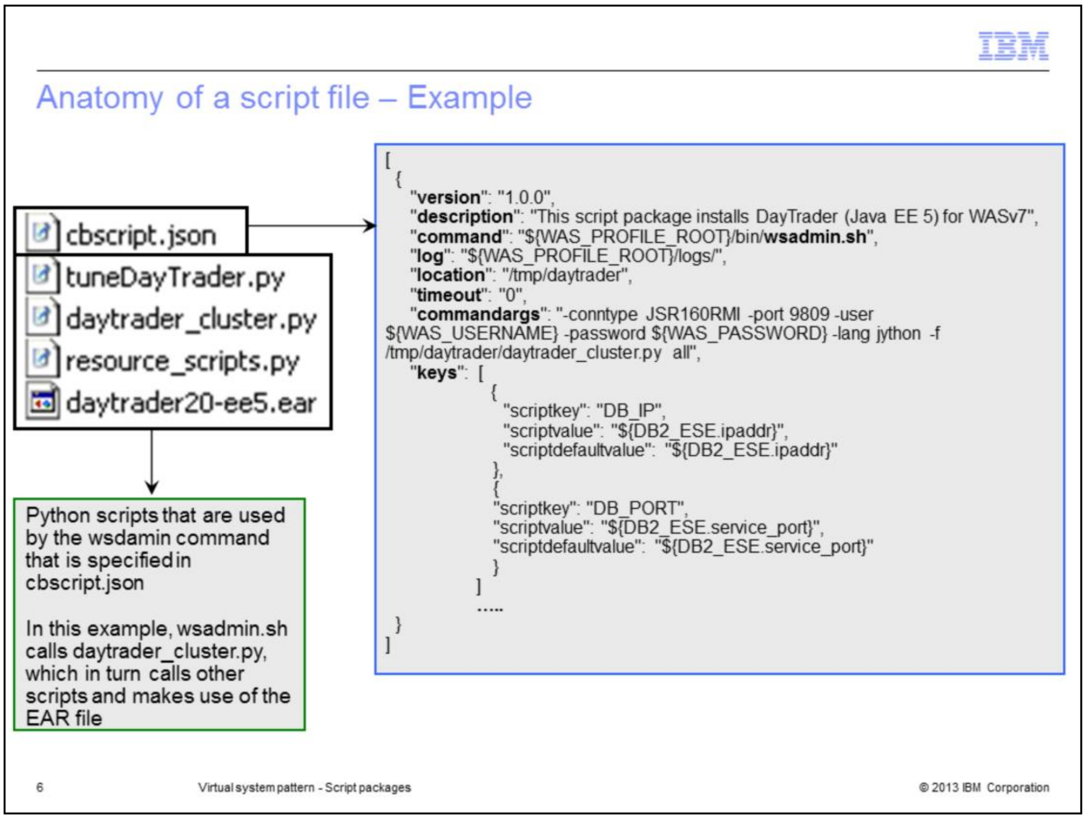
As part of the script package, you need to specify the working directory where the script is run and the logging directory where the logs are created. Also, you need to specify the initial script to be run, its parameters and any environment variables used by the script.

These values can be provided during the import or they can be provided by the script package owner in a JSON file, named “cbscript.json”. Providing the values in the JSON file avoids the possibility of errors being introduced when they are provided in the user interface.

The package contains one or more script files. The script that is specified as the initial script is run first. It can call other scripts. Scripts can use the pre-defined environment variables that are available for each part, such as the IP address and host name, or your script can use environment variables whose values are provided at deployment time.

Only users who have permission to “create catalog content” can import the script packages.

On a part, such as Deployment Manager, you can add multiple script packages and specify the order in which they run.



Anatomy of a script file – Example

cbscript.json
tuneDayTrader.py
daytrader_cluster.py
resource_scripts.py
daytrader20-ee5.ear

```
[
  {
    "version": "1.0.0",
    "description": "This script package installs DayTrader (Java EE 5) for WASv7",
    "command": "${WAS_PROFILE_ROOT}/bin/wsadmin.sh",
    "log": "${WAS_PROFILE_ROOT}/logs/",
    "location": "/tmp/daytrader",
    "timeout": "0",
    "commandargs": "-conntype JSR160RMI -port 9809 -user
${WAS_USERNAME} -password ${WAS_PASSWORD} -lang jython -f
/tmp/daytrader/daytrader_cluster.py  all",
    "keys": [
              {
                "scriptkey": "DB_IP",
                "scriptvalue": "${DB2_ESE.ipaddr}",
                "scriptdefaultvalue": "${DB2_ESE.ipaddr}"
              },
              {
              "scriptkey": "DB_PORT",
              "scriptvalue": "${DB2_ESE.service_port}",
              "scriptdefaultvalue": "${DB2_ESE.service_port}"
              }
            ]
            .....
  }
]
```

Python scripts that are used by the wsadmin command that is specified in cbscript.json

In this example, wsadmin.sh calls daytrader_cluster.py, which in turn calls other scripts and makes use of the EAR file

6 Virtual system pattern - Script packages © 2013 IBM Corporation

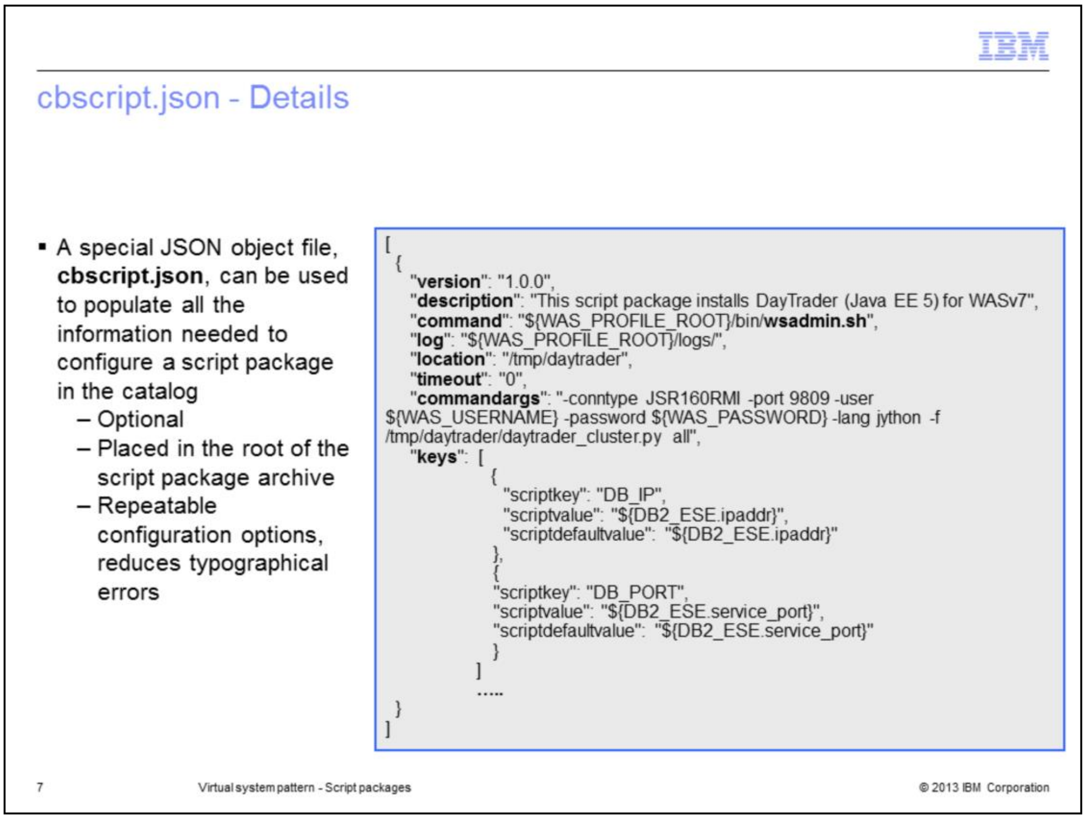
An example of a script package is shown here. This script package archive file installs a Java EE EAR file to a WebSphere Application Server cluster.

The file named cbscript.json supplies the parameters that define how the script is run at deployment time. It contains all the parameters that need to be supplied when you load a script package into the PureApplication System catalog. When the parameter values are provided in the cbscript.json file, they are automatically loaded into the user interface, which avoids data-entry errors. It also makes it easier to move script files from one PureApplication System to another. Therefore, it is a good practice to include the cbscript.json file in the package. This file must be in the root of the script package archive.

In this example, the command to be run is wsadmin.sh. This is a command-line configuration script that is shipped with WebSphere Application Server; therefore the wsadmin script file is not included in the package. But, you can include your own script file in the package and specify it as the command to run.

Continuing with the example shown on the slide, the wsadmin script uses the daytrader_cluster python script file, as specified in the line that begins with “commandargs”. The EAR file and the other python scripts that are included in the package are used by the daytrader_cluster python script.

As you can see from the format, the script package is very flexible and allows you to run any OS or middleware commands.



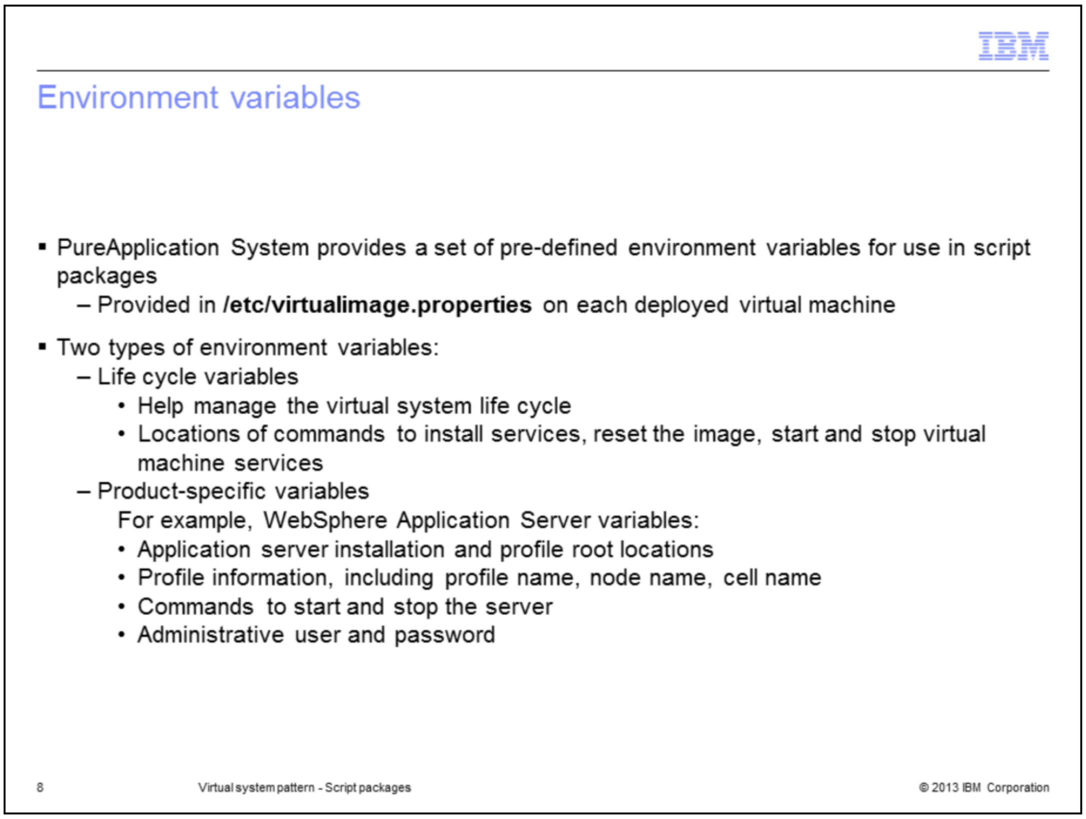
cbscript.json - Details

- A special JSON object file, **cbscript.json**, can be used to populate all the information needed to configure a script package in the catalog
  - Optional
  - Placed in the root of the script package archive
  - Repeatable configuration options, reduces typographical errors

```
[
  {
    "version": "1.0.0",
    "description": "This script package installs DayTrader (Java EE 5) for WASv7",
    "command": "${WAS_PROFILE_ROOT}/bin/wsadmin.sh",
    "log": "${WAS_PROFILE_ROOT}/logs/",
    "location": "/tmp/daytrader",
    "timeout": "0",
    "commandargs": "-conntype JSR160RMI -port 9809 -user
${WAS_USERNAME} -password ${WAS_PASSWORD} -lang jython -f
/tmp/daytrader/daytrader_cluster.py  all",
    "keys": [
              {
                "scriptkey": "DB_IP",
                "scriptvalue": "${DB2_ESE.ipaddr}",
                "scriptdefaultvalue": "${DB2_ESE.ipaddr}"
              },
              {
              "scriptkey": "DB_PORT",
              "scriptvalue": "${DB2_ESE.service_port}",
              "scriptdefaultvalue": "${DB2_ESE.service_port}"
              }
            ]
            .....
  }
]
```

7 Virtual system pattern - Script packages © 2013 IBM Corporation

In the cbscript.json file, the parameters, which are highlighted on the slide in bold, correspond to the fields you will see in the user interface for script package definitions. The "**description**" field is optional, but recommended, to describe the purpose of the script package. The "**command**" field defines the command to be run when the script package is run. This example script package installs an application into the WebSphere Application Server instance, by using the wsadmin script. You can also specify the "**log**" location for log output. The "**location**" field tells PureApplication System where to expand the archive file on the deployed virtual machine. This is important if you need to reference any of the files included in the archive file, for example in the "**commandargs**" field. The "**commandargs**" field is where you specify the parameters for the command. In this example, the wsadmin script requires the location of the jython file that it needs to run in order to perform the application installation. The "**timeout**" parameter specifies the maximum amount of time that PureApplication System should wait for this script package to finish running on the virtual machine. A value of zero means PureApplication System should wait indefinitely. The last field, "**keys**", is where you define additional properties that are needed by this script. In this example. one of the properties that is needed at deployment time is the application location. PureApplication System prompts for these property values during the pattern deployment.



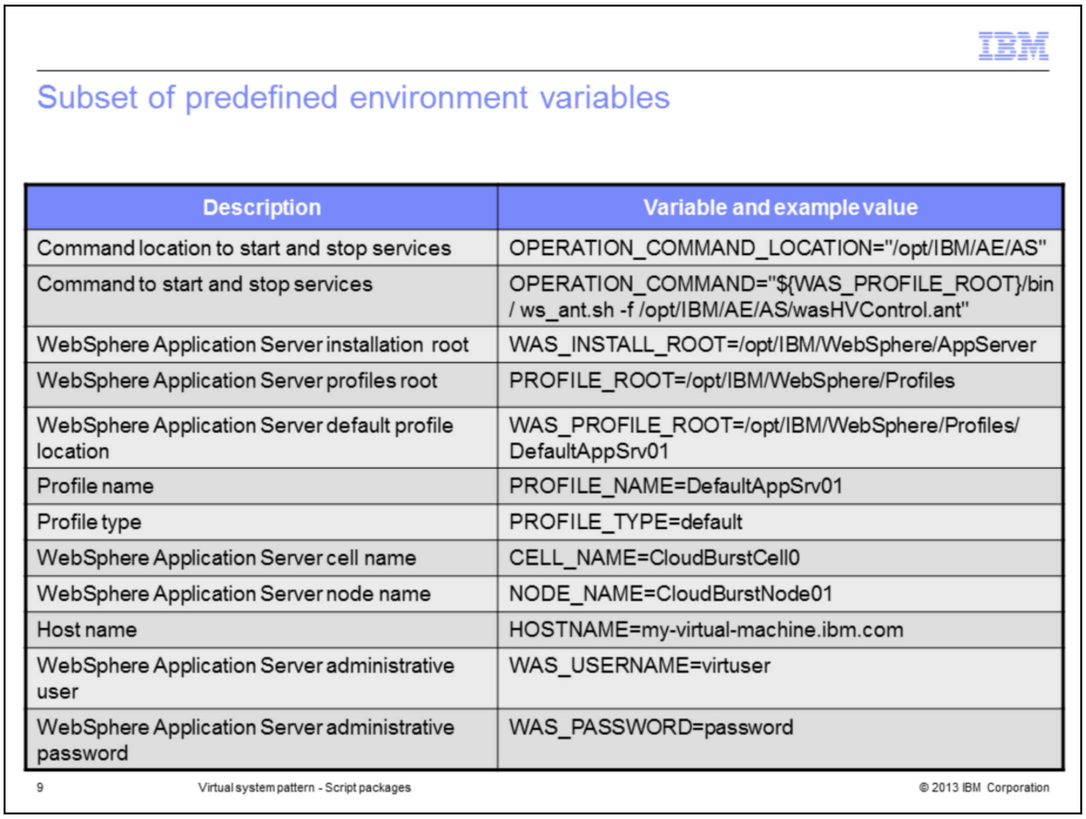
Environment variables

- PureApplication System provides a set of pre-defined environment variables for use in script packages
  - Provided in **/etc/virtualimage.properties** on each deployed virtual machine
- Two types of environment variables:
  - Life cycle variables
    - Help manage the virtual system life cycle
    - Locations of commands to install services, reset the image, start and stop virtual machine services
  - Product-specific variables

    For example, WebSphere Application Server variables:
    - Application server installation and profile root locations
    - Profile information, including profile name, node name, cell name
    - Commands to start and stop the server
    - Administrative user and password

8 Virtual system pattern - Script packages © 2013 IBM Corporation

PureApplication System provides some environment variables that can be used in script packages. These variables are defined in the /etc/virtualimage.properties file on the deployed virtual machines. There are two types of variables. The first type is life cycle variables. These are related to the management of the virtual system. For example, there might be variables for the locations of the commands for installing services or for starting and stopping the virtual machine services. The second type of environment variables are related to a product that is installed on the virtual machine. For example, for WebSphere Application Server there are variables for the application server installation location, the profile root location, and the profile, node and cell names.

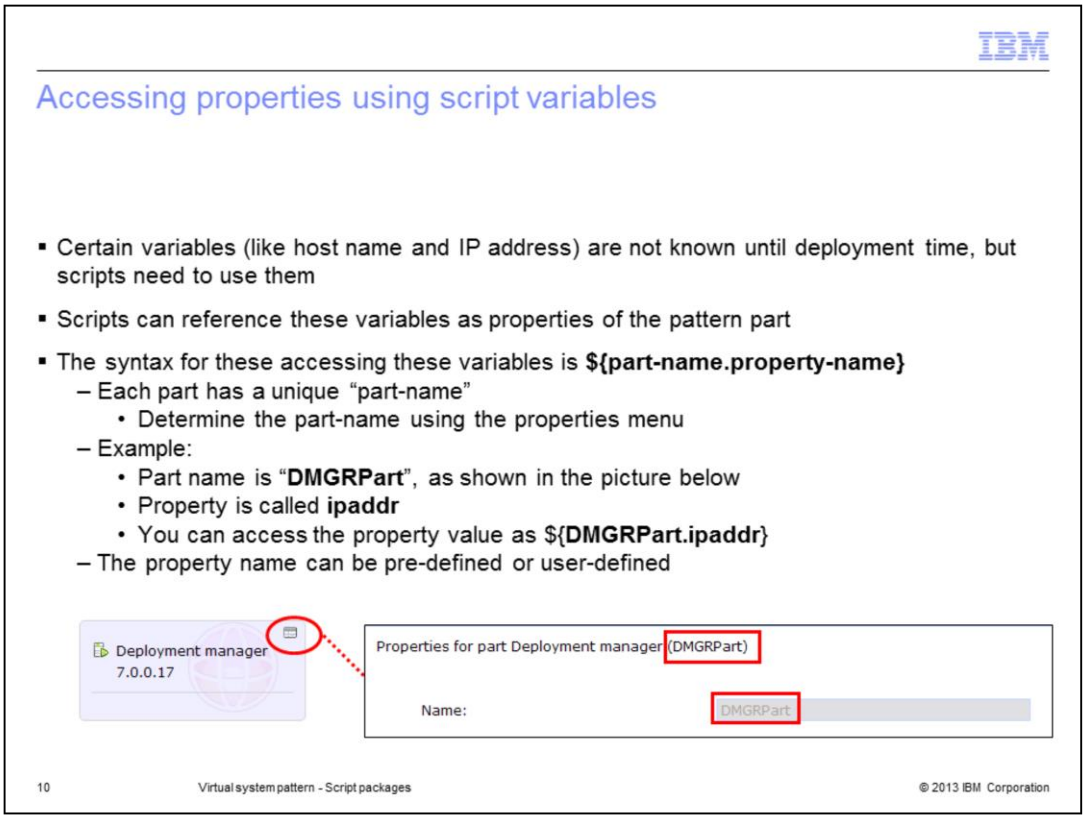


Subset of predefined environment variables

Description	Variable and example value
Command location to start and stop services	OPERATION_COMMAND_LOCATION="/opt/IBM/AE/AS"
Command to start and stop services	OPERATION_COMMAND="${WAS_PROFILE_ROOT}/bin / ws_ant.sh -f /opt/IBM/AE/AS/wasHVControl.ant"
WebSphere Application Server installation root	WAS_INSTALL_ROOT=/opt/IBM/WebSphere/AppServer
WebSphere Application Server profiles root	PROFILE_ROOT=/opt/IBM/WebSphere/Profiles
WebSphere Application Server default profile location	WAS_PROFILE_ROOT=/opt/IBM/WebSphere/Profiles/ DefaultAppSrv01
Profile name	PROFILE_NAME=DefaultAppSrv01
Profile type	PROFILE_TYPE=default
WebSphere Application Server cell name	CELL_NAME=CloudBurstCell0
WebSphere Application Server node name	NODE_NAME=CloudBurstNode01
Host name	HOSTNAME=my-virtual-machine.ibm.com
WebSphere Application Server administrative user	WAS_USERNAME=virtuser
WebSphere Application Server administrative password	WAS_PASSWORD=password

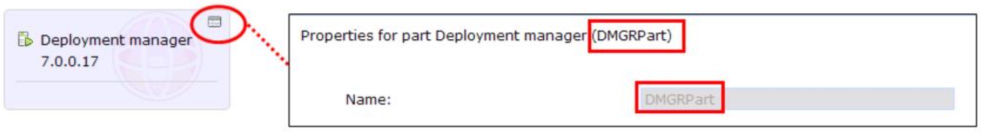
9 Virtual system pattern - Script packages © 2013 IBM Corporation

This slide shows a small subset of the environment variables that are available for use by your scripts. The first two are examples of the life cycle variables. The others are some of the more common ones for WebSphere Application Server deployments. The values are specific to each deployment. Some values do not change, however, such as the installation root.



Accessing properties using script variables

- Certain variables (like host name and IP address) are not known until deployment time, but scripts need to use them
- Scripts can reference these variables as properties of the pattern part
- The syntax for these accessing these variables is **${part-name.property-name}**
  - Each part has a unique "part-name"
    - Determine the part-name using the properties menu
  - Example:
    - Part name is "**DMGRPart**", as shown in the picture below
    - Property is called **ipaddr**
    - You can access the property value as ${**DMGRPart.ipaddr**}
  - The property name can be pre-defined or user-defined

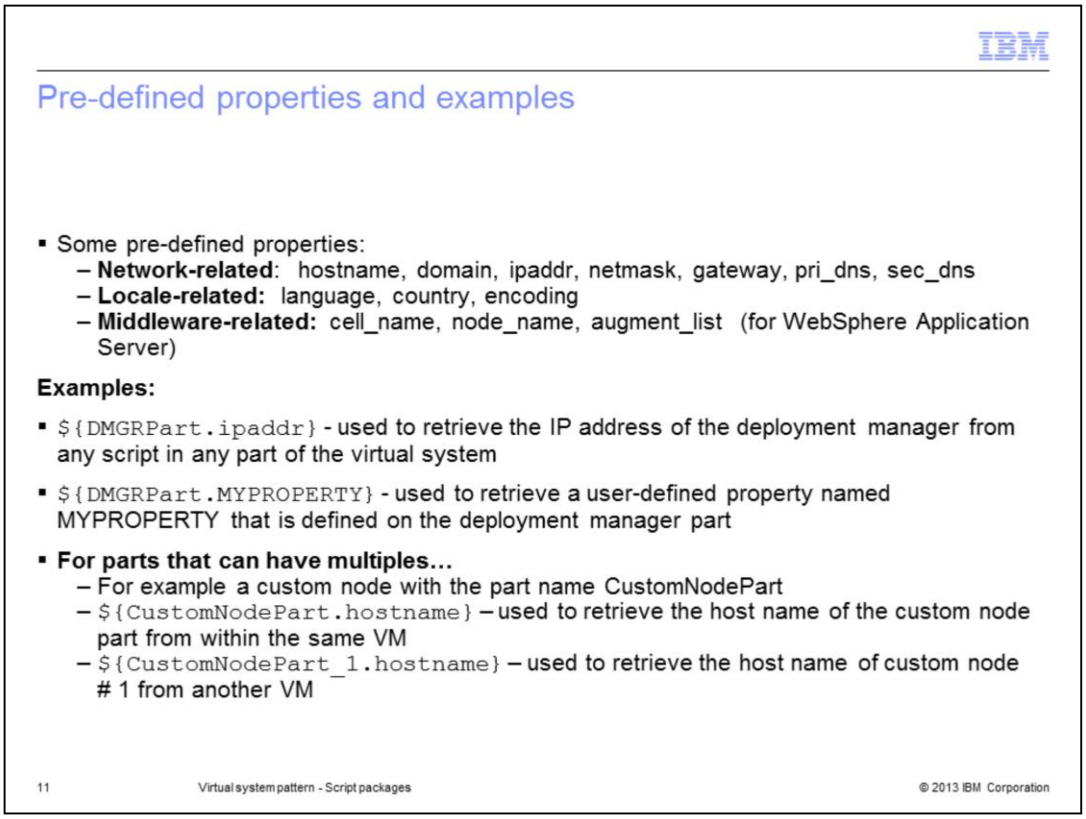


10 Virtual system pattern - Script packages © 2013 IBM Corporation

The values for some properties, such as host name and IP address, are not known before deployment. A script running on one virtual machine might need the value for one of these properties for another virtual machine. For example, the script that configures the JDBC resource for a database needs to know the host name or IP address of the database virtual machine. PureApplication System allows the script to reference the variables for other virtual images in the pattern by using the part name.

Each part has a unique name that can be found by clicking the properties icon of the part, as shown on the slide. The value in the Name field is the part name. For example, the part name for the deployment manager part is DMGRPart. Referencing a property associated with that part can be done using the syntax that is shown on the slide. Note that this syntax does not support the use of the space character in the part name, such as Core OS.

PureApplication System will substitute the property values before running the script. This is very useful in a multi-part pattern where many of the properties like host name and IP address are not known until deployment time.



Pre-defined properties and examples

- Some pre-defined properties:
  - **Network-related**: hostname, domain, ipaddr, netmask, gateway, pri_dns, sec_dns
  - **Locale-related:** language, country, encoding
  - **Middleware-related:** cell_name, node_name, augment_list (for WebSphere Application Server)

Examples:

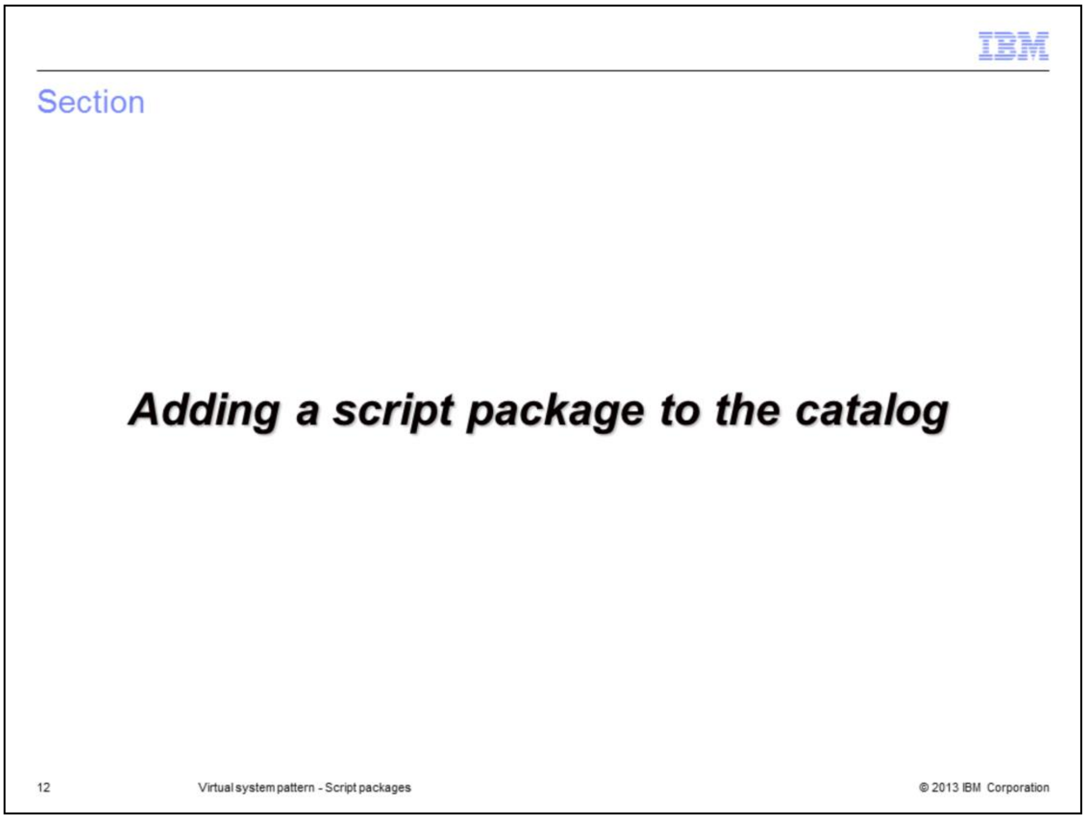
- `${DMGRPart.ipaddr}` - used to retrieve the IP address of the deployment manager from any script in any part of the virtual system
- `${DMGRPart.MYPROPERTY}` - used to retrieve a user-defined property named MYPROPERTY that is defined on the deployment manager part
- **For parts that can have multiples…**
  - For example a custom node with the part name CustomNodePart
  - `${CustomNodePart.hostname}` – used to retrieve the host name of the custom node part from within the same VM
  - `${CustomNodePart_1.hostname}` – used to retrieve the host name of custom node # 1 from another VM

11 Virtual system pattern - Script packages © 2013 IBM Corporation

Listed here are some pre-defined properties. The network-related properties are host name, domain, IP address, net mask, and primary and secondary DNS of the deployed virtual machine. The locale-related properties are the language, country and encoding that are used in the VM. The middleware-related properties are specific to the middleware that is installed on the virtual machine. For WebSphere Application Server, some of the properties are cell name, node name and augment list.

These pre-defined properties allow any script in the pattern get the network, locale and middleware properties of any deployed virtual machine for a part in a pattern.

Some examples of how to access these properties are shown here.

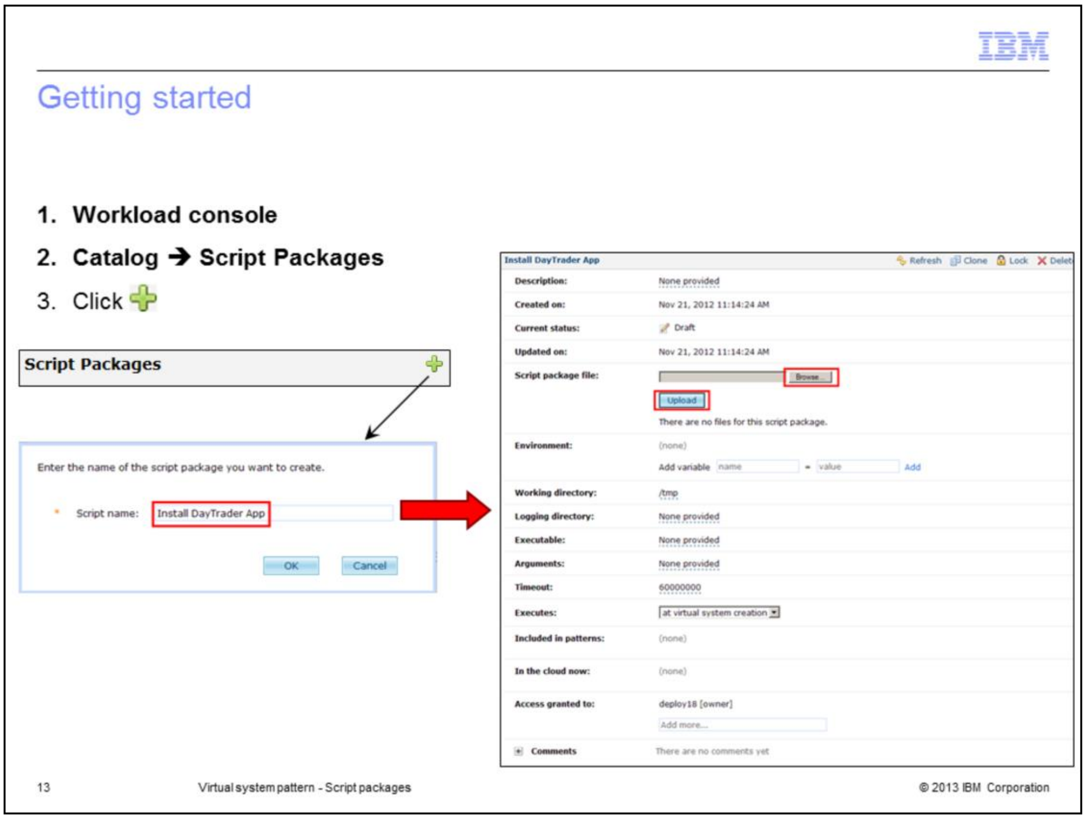


Section

Adding a script package to the catalog

12 Virtual system pattern - Script packages © 2013 IBM Corporation

Now that you have seen what goes into a script package archive, this next section will cover adding a script package to the catalog.



Getting started

1. **Workload console**
2. **Catalog ➔ Script Packages**
3. Click ✚

Enter the name of the script package you want to create.

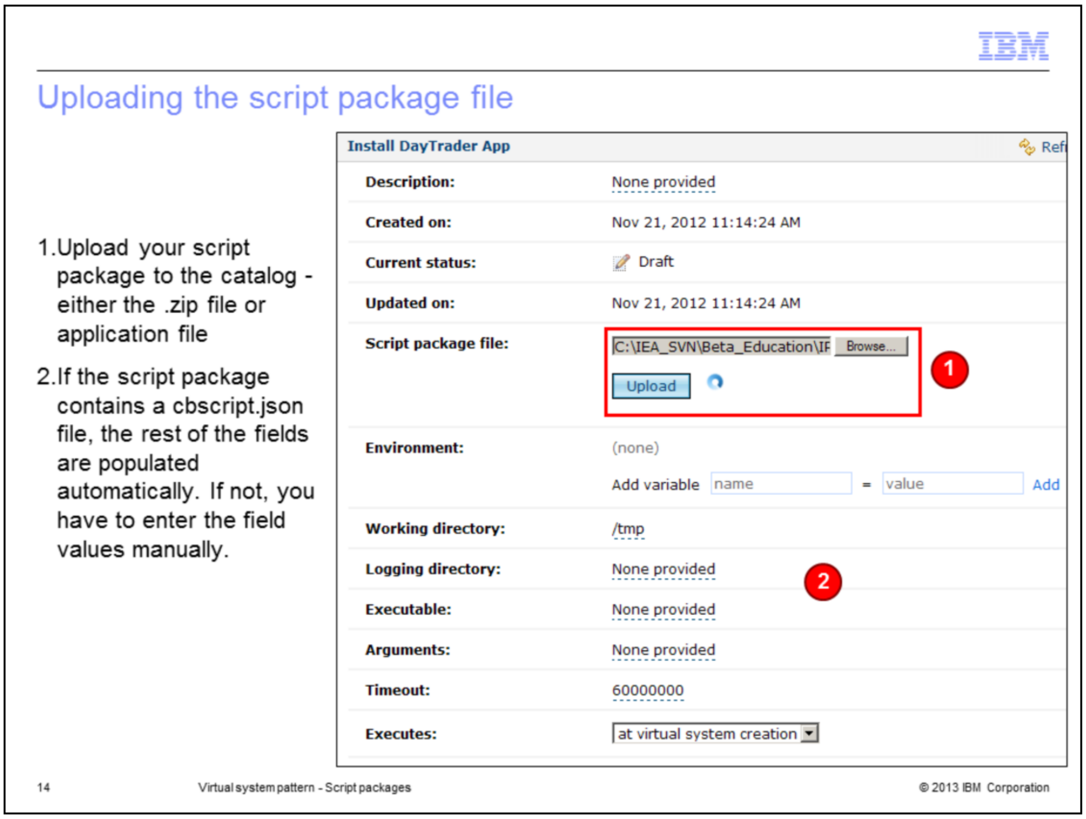
* Script name: Install DayTrader App

Install DayTrader App

Description:	None provided
Created on:	Nov 21, 2012 11:14:24 AM
Current status:	Draft
Updated on:	Nov 21, 2012 11:14:24 AM
Script package file:	Browse... Upload There are no files for this script package.
Environment:	(none) Add variable name = value Add
Working directory:	/tmp
Logging directory:	None provided
Executable:	None provided
Arguments:	None provided
Timeout:	60000000
Executes:	at virtual system creation
Included in patterns:	(none)
In the cloud now:	(none)
Access granted to:	deploy18 [owner] Add more...
Comments	There are no comments yet

13 Virtual system pattern - Script packages © 2013 IBM Corporation

To add a script package to the PureApplication System catalog, in the Workload console, go to Catalog > Script Packages and click the green plus sign to add a new script package. You are prompted for the script name. Then the next screen is where you will provide all the details about the script package. The first thing you should do is upload the script package using the Browse and Upload buttons.



Uploading the script package file

1. Upload your script package to the catalog - either the .zip file or application file
2. If the script package contains a cbscript.json file, the rest of the fields are populated automatically. If not, you have to enter the field values manually.

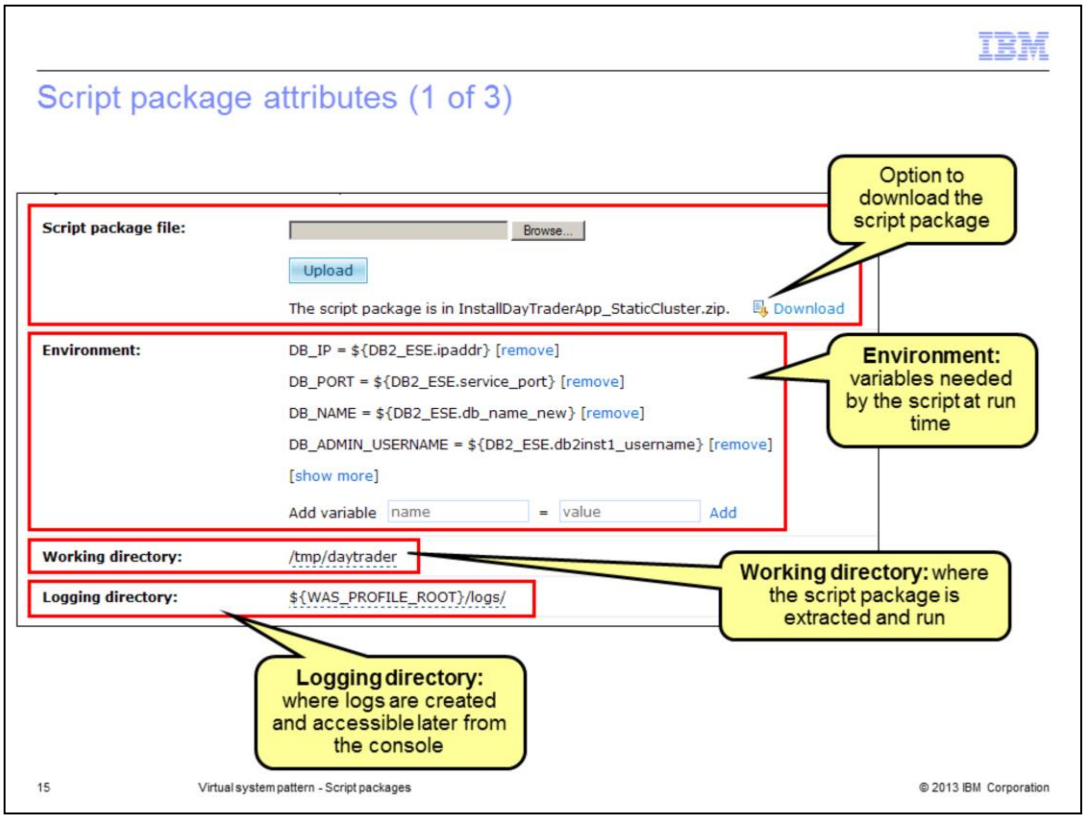
Install DayTrader App

Description:	None provided
Created on:	Nov 21, 2012 11:14:24 AM
Current status:	Draft
Updated on:	Nov 21, 2012 11:14:24 AM
Script package file:	C:\IEA_SVN\Beta_Education\IF Browse... Upload
Environment:	(none) Add variable name = value Add
Working directory:	/tmp
Logging directory:	None provided
Executable:	None provided
Arguments:	None provided
Timeout:	60000000
Executes:	at virtual system creation

14 Virtual system pattern - Script packages © 2013 IBM Corporation

Once you push the Upload button, there is an indicator that the package is being uploaded. You can see this indicator inside the red box with the number one

If the script package contains a cbscript.json file, the rest of the fields are populated with the values that are specified in that file. If the script package does not contain a cbscript.json file, you will have to enter values in those fields manually.



Script package attributes (1 of 3)

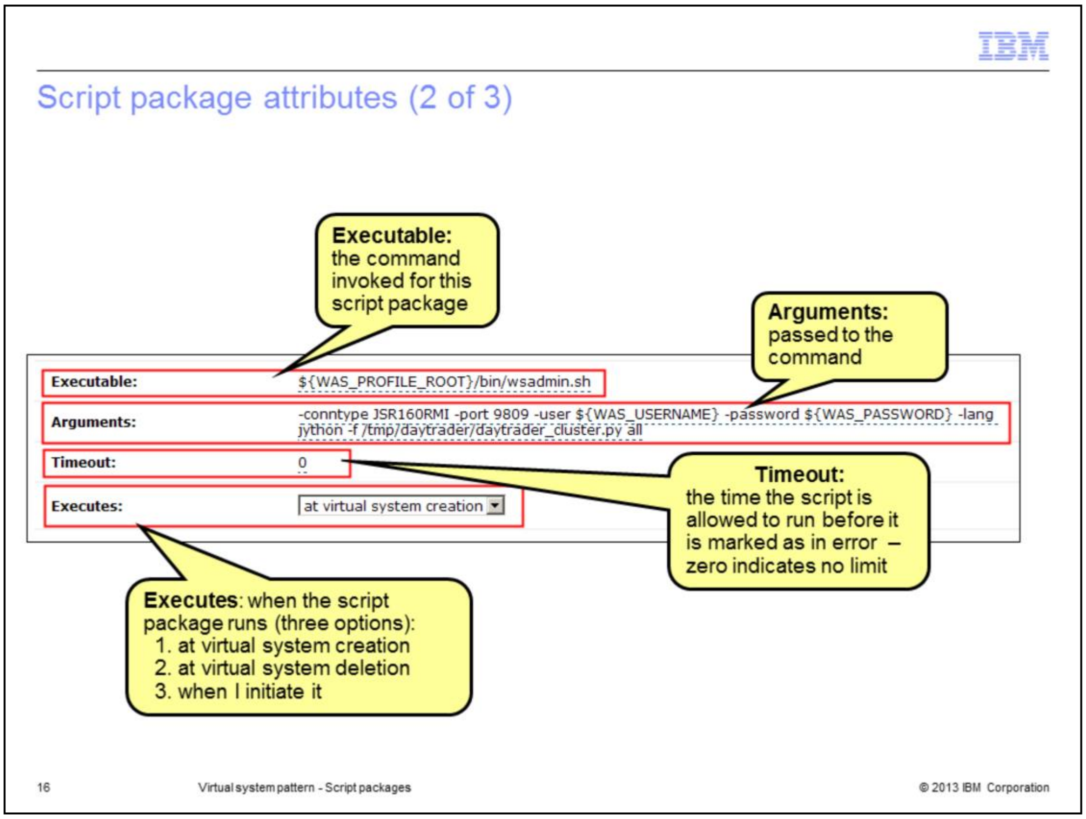
Script package file:	Browse... Upload The script package is in InstallDayTraderApp_StaticCluster.zip. Download
Environment:	DB_IP = ${DB2_ESE.ipaddr} [remove]
	DB_PORT = ${DB2_ESE.service_port} [remove]
	DB_NAME = ${DB2_ESE.db_name_new} [remove]
	DB_ADMIN_USERNAME = ${DB2_ESE.db2inst1_username} [remove]
	[show more]
	Add variable name = value Add
Working directory:	/tmp/daytrader
Logging directory:	${WAS_PROFILE_ROOT}/logs/

Option to download the script package
Environment: variables needed by the script at run time
Working directory: where the script package is extracted and run
Logging directory: where logs are created and accessible later from the console

The Environment section corresponds to the ‘keys’ section of your cbscript.json file. This is where you define any environment variables that to your script package needs at run time. You can provide a default value for each of the environment variables.

The next field is ‘Working directory’. This is where the script package is extracted and run on your deployed virtual machine.

’Logging directory’ specifies the location of the logs that are generated by the scripts.



Script package attributes (2 of 3)

Executable: the command invoked for this script package

Arguments: passed to the command

Executable:	${WAS_PROFILE_ROOT}/bin/wsadmin.sh
Arguments:	-conntype JSR160RMI -port 9809 -user ${WAS_USERNAME} -password ${WAS_PASSWORD} -lang jython -f /tmp/daytrader/daytrader_cluster.py all
Timeout:	0
Executes:	at virtual system creation

Timeout: the time the script is allowed to run before it is marked as in error – zero indicates no limit

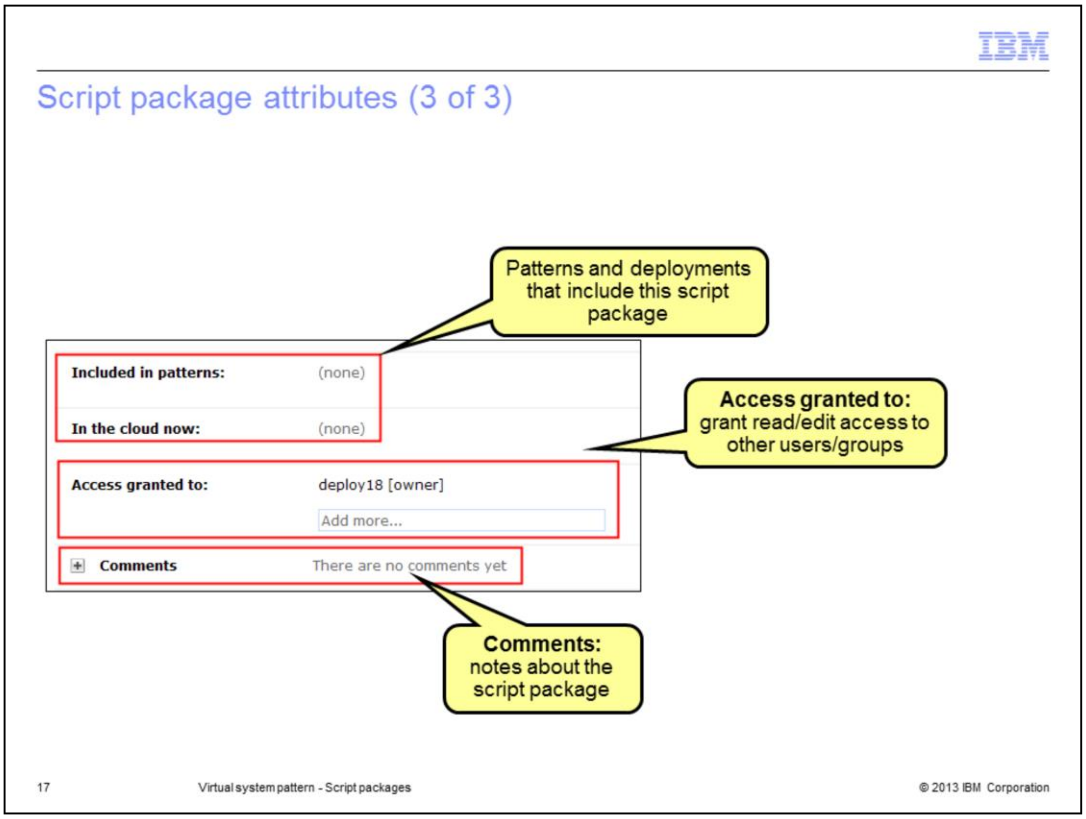
Executes: when the script package runs (three options):
1. at virtual system creation
2. at virtual system deletion
3. when I initiate it

16 Virtual system pattern - Script packages © 2013 IBM Corporation

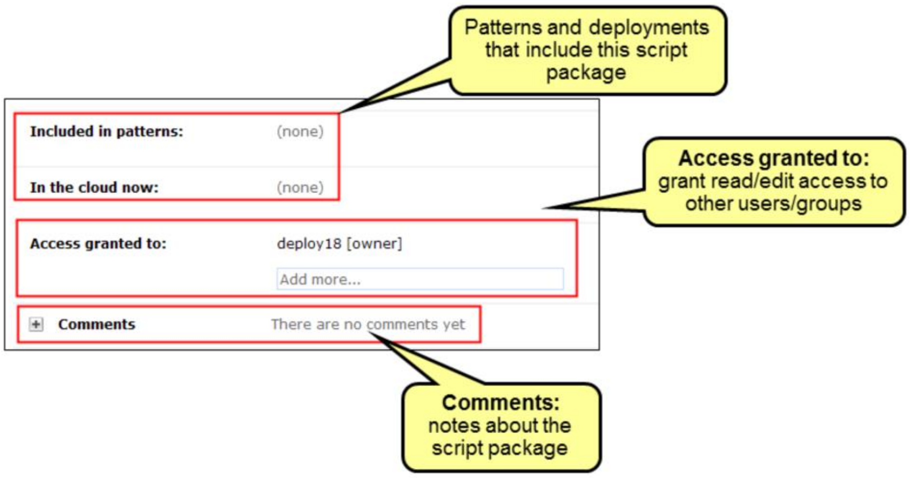
The ‘Executable’ property specifies the command to run when the script package is run. The ‘Arguments’ property lists the arguments to be passed to that command. The example shown here is the wsadmin command. Note that pasting the arguments for the wsadmin command into the Arguments field often leads to translation problems that cause the command to run incorrectly. It is best to either put these parameters in a cbscript.json file or type them here manually.

The ‘Timeout’ property specifies the maximum time that the script should be allowed to run. If script does not complete in the specified time, then PureApplication System will treat it as an error and stop execution. A timeout value of 0 indicates that there is no time limit.

The ‘Executes’ property allows you to specify when the script package is run. The default value is ‘at virtual system creation’. Other options are ‘at virtual system deletion’ and ‘when I initiate it’. An example of a script that might run at virtual system deletion is one that releases resources or locks that the middleware on the virtual machine might have obtained on external resources. In order for this script to run, the virtual machine must be running at the time that the virtual system is deleted. If you select ‘when I initiate it’, then there will be an ‘Execute now’ option in the script package section of the virtual machine. This option is handy for debugging your scripts. There is no limit on the number of times a script can be run using ‘Execute now’.



Script package attributes (3 of 3)



Included in patterns:	(none)
In the cloud now:	(none)
Access granted to:	deploy18 [owner]
	Add more...
Comments	There are no comments yet

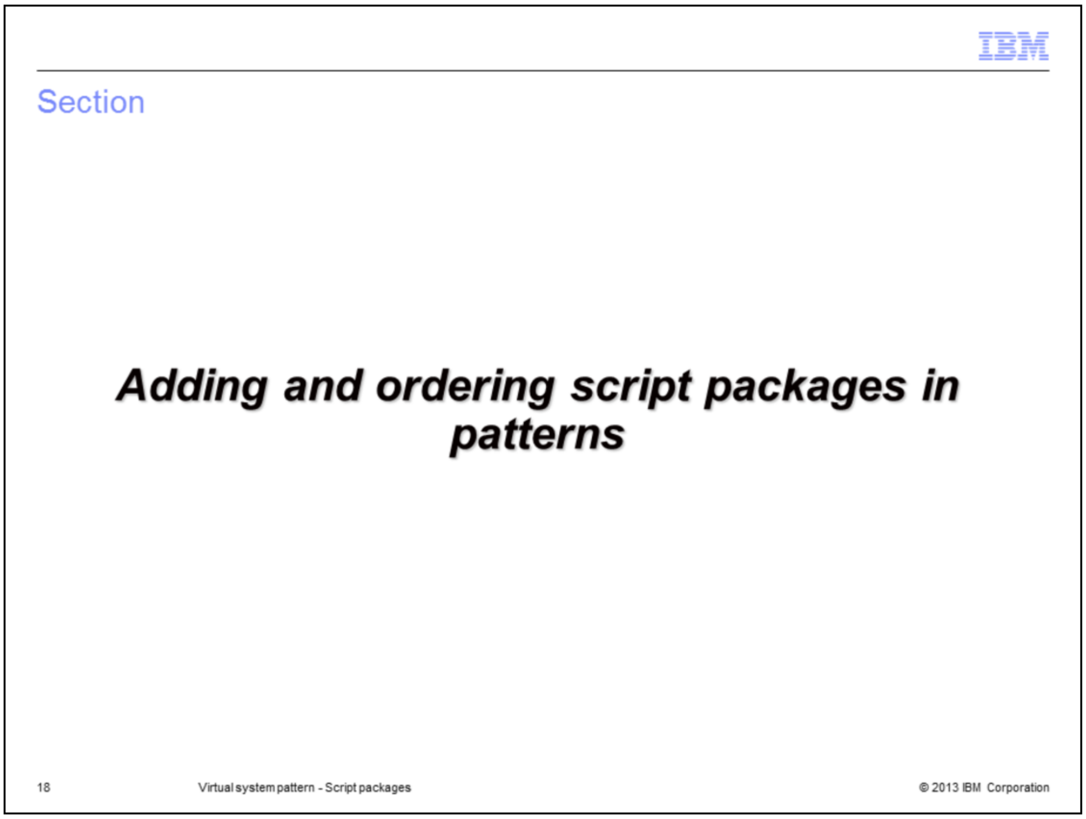
Patterns and deployments that include this script package
Access granted to: grant read/edit access to other users/groups
Comments: notes about the script package

17 Virtual system pattern - Script packages © 2013 IBM Corporation

Any patterns and instance deployments currently in the cloud that include this script package are listed in these sections. A script package cannot be deleted from the catalog if it is included in a pattern or instance.

The ‘Access granted to’ section shows the owner of the script package and allows the owner to give fine-grained read or edit access to other users and groups.

The last field is the comment field, which allows the owner and editors to add comments for documentation purposes.


Section

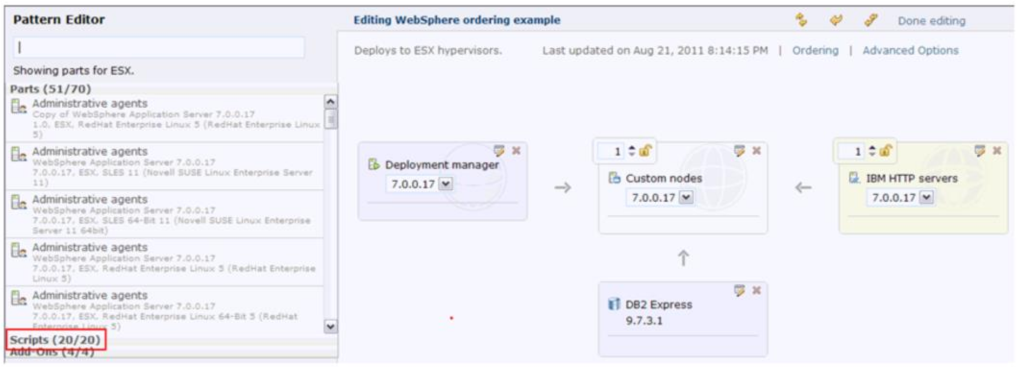
Adding and ordering script packages in patterns

18 Virtual system pattern - Script packages © 2013 IBM Corporation

This section describes how to add script packages to a pattern and specify the order in which they should run.

Adding script packages to a pattern (1 of 2)

- In the **Pattern Editor**, you can drag script packages from the palette and drop them onto parts in your pattern canvas
- To open the editor:
  - Go to **Patterns** in the console
  - In the left navigation, select the pattern you want to edit
  - Click the edit icon on the toolbar
- Script packages are at the bottom of the palette; click to expand



Pattern Editor

Showing parts for ESX.

Parts (51/70)

Administrative agents
Copy of WebSphere Application Server 7.0.0.17
1.0, ESX, RedHat Enterprise Linux 5 (RedHat Enterprise Linux 5)

Administrative agents
WebSphere Application Server 7.0.0.17
7.0.0.17, ESX, SLES 11 (Novell SUSE Linux Enterprise Server 11)

Administrative agents
WebSphere Application Server 7.0.0.17
7.0.0.17, ESX, SLES 64-Bit 11 (Novell SUSE Linux Enterprise Server 11 64bit)

Administrative agents
WebSphere Application Server 7.0.0.17
7.0.0.17, ESX, RedHat Enterprise Linux 5 (RedHat Enterprise Linux 5)

Administrative agents
WebSphere Application Server 7.0.0.17
7.0.0.17, ESX, RedHat Enterprise Linux 64-Bit 5 (RedHat Enterprise Linux 5)

Scripts (20/20)

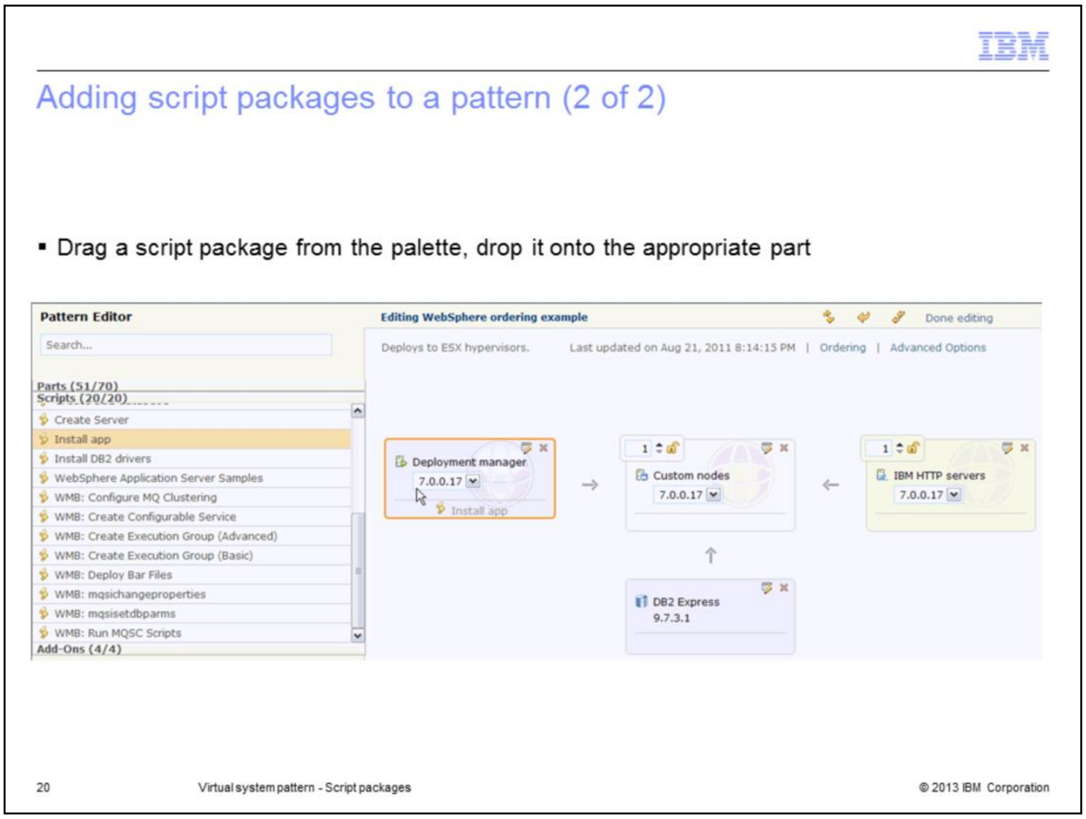
Add-Ons (4/4)

Editing WebSphere ordering example

Done editing

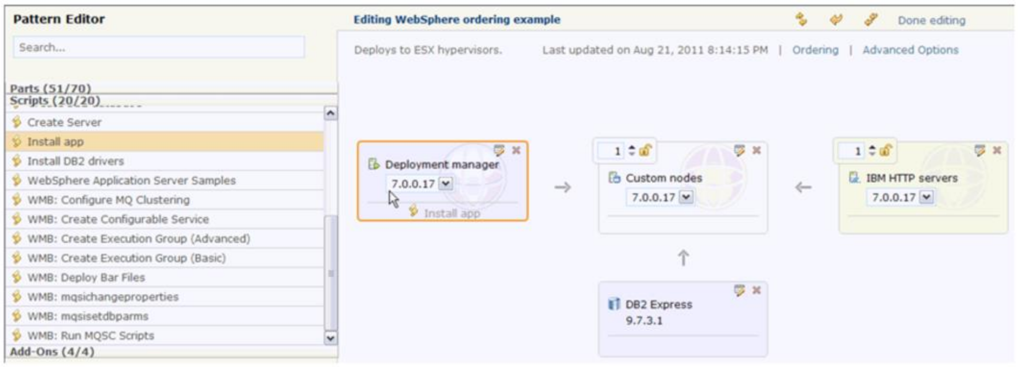
Deploys to ESX hypervisors. Last updated on Aug 21, 2011 8:14:15 PM | Ordering | Advanced Options

Adding a script package to a pattern is a drag-and-drop activity that is done in the pattern editor. There is a ‘Scripts’ category near the bottom of the palette on the left side of the screen. Click ‘Scripts’ to see the list of available script packages.



Adding script packages to a pattern (2 of 2)

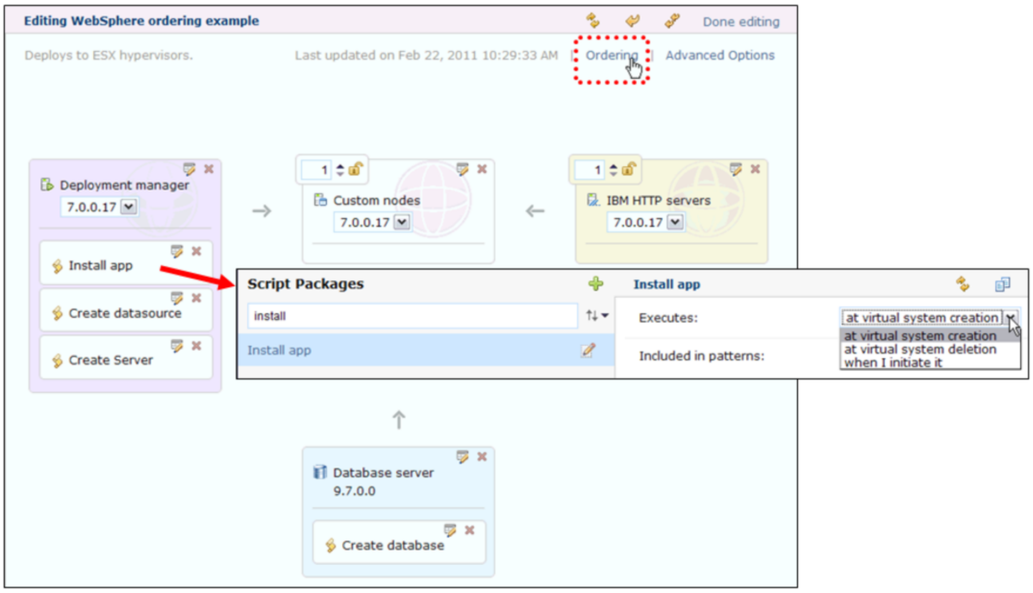
- Drag a script package from the palette, drop it onto the appropriate part


Pattern Editor
Search...
Parts (51/70)
Scripts (20/20)
Create Server
Install app
Install DB2 drivers
WebSphere Application Server Samples
WMB: Configure MQ Clustering
WMB: Create Configurable Service
WMB: Create Execution Group (Advanced)
WMB: Create Execution Group (Basic)
WMB: Deploy Bar Files
WMB: mqsichangeproperties
WMB: mqsisetdbparms
WMB: Run MQSC Scripts
Add-Ons (4/4)
Editing WebSphere ordering example
Done editing
Deploys to ESX hypervisors.
Last updated on Aug 21, 2011 8:14:15 PM | Ordering | Advanced Options
Deployment manager
7.0.0.17
Install app
1
Custom nodes
7.0.0.17
1
IBM HTTP servers
7.0.0.17
DB2 Express
9.7.3.1

Drag a script package from the palette and drop it onto the appropriate virtual image part in the pattern on the right side of the screen. The script package will run on that virtual image part.

In the example shown on this slide, the ‘Install app’ script package is added to the deployment manager part.

Ordering script packages (example)



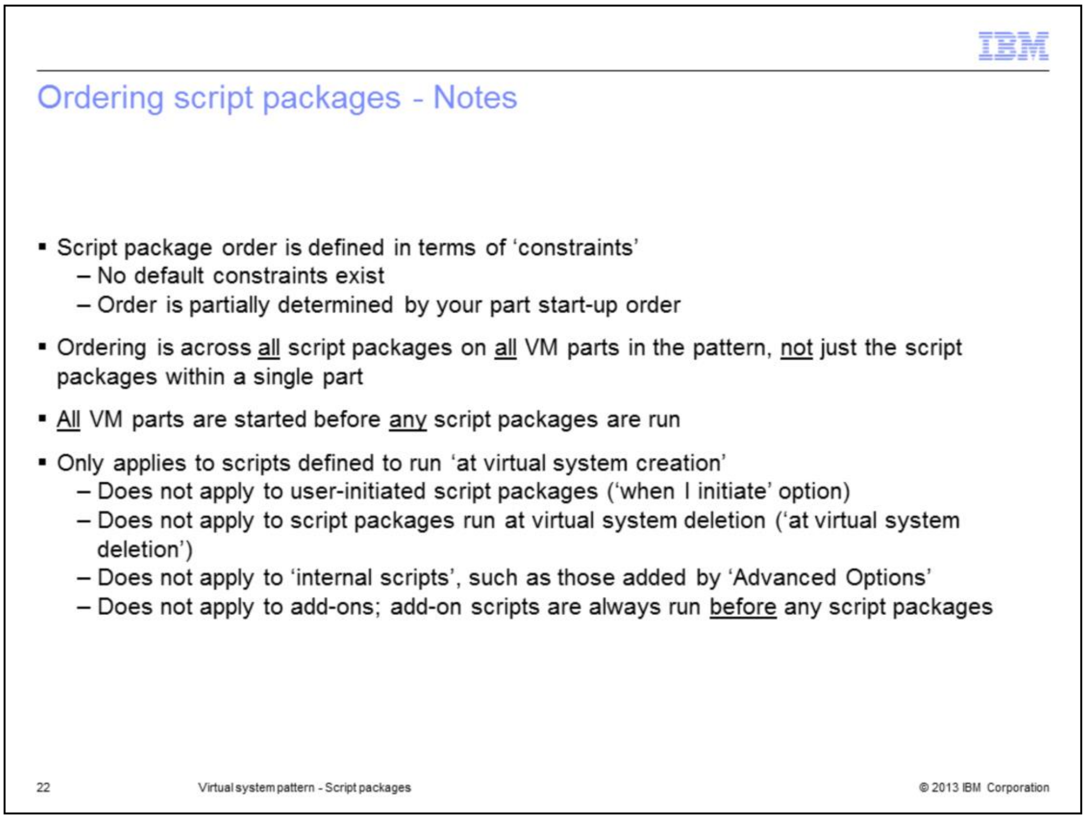
Executes: at virtual system creation

at virtual system creation
at virtual system deletion
when I initiate it

Included in patterns:

21 Virtual system pattern - Script packages © 2013 IBM Corporation

In the example shown on this slide, three script packages have been added to the deployment manager part and one script package has been added to the database server. All of these script packages run ‘at virtual system creation’. You can use the “Ordering” option, which is circled at the top of the picture, to specify the order in which these script run.



Ordering script packages - Notes

- Script package order is defined in terms of 'constraints'
  - No default constraints exist
  - Order is partially determined by your part start-up order
- Ordering is across all script packages on all VM parts in the pattern, not just the script packages within a single part
- All VM parts are started before any script packages are run
- Only applies to scripts defined to run 'at virtual system creation'
  - Does not apply to user-initiated script packages ('when I initiate' option)
  - Does not apply to script packages run at virtual system deletion ('at virtual system deletion')
  - Does not apply to 'internal scripts', such as those added by 'Advanced Options'
  - Does not apply to add-ons; add-on scripts are always run before any script packages

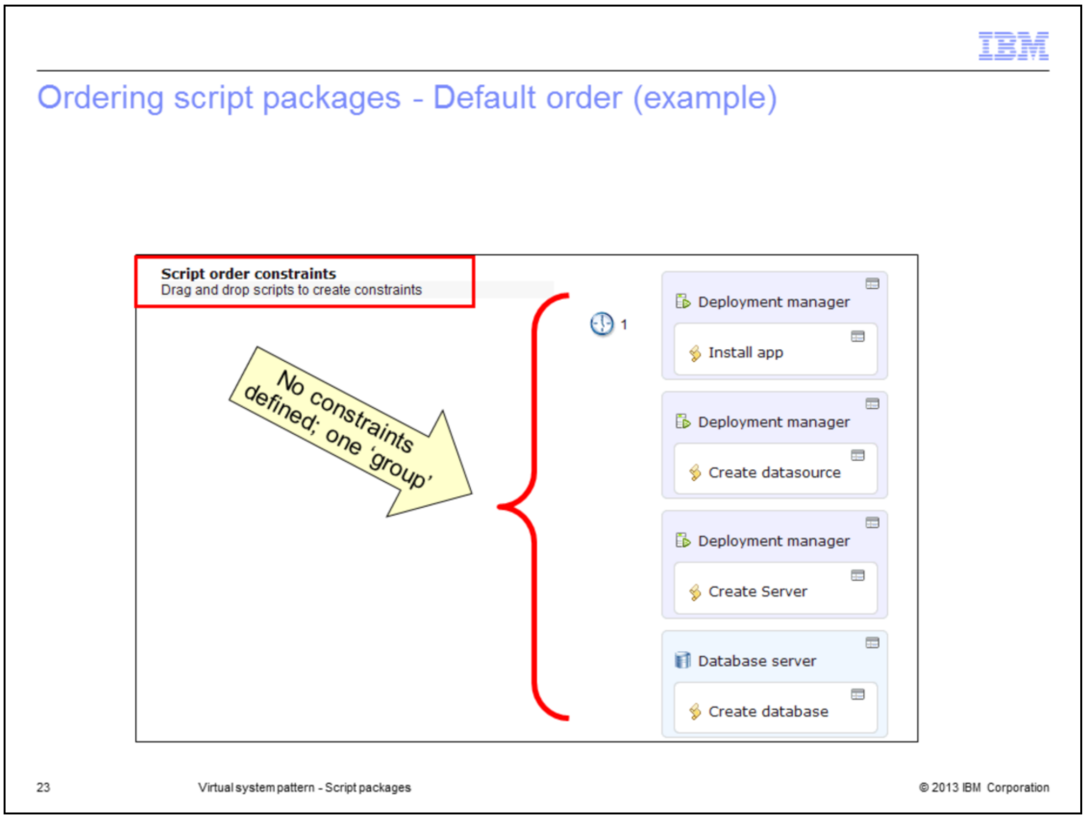
22 Virtual system pattern - Script packages © 2013 IBM Corporation

Here are some important notes about script package ordering.

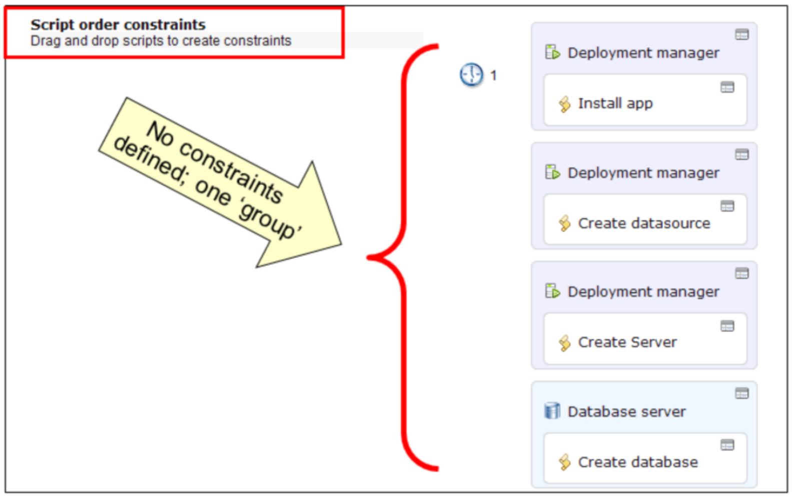
The script packages run after all of the virtual machines are started. You cannot specify that PureApplication System should run a package script on one part before another part is started.

The order that the script packages will run is defined in terms of constraints. An example of a constraint is "script package A runs after script package B". This is similar to the way you define the order that parts are started. However, there are no default constraints for script packages, like there are for parts. There is no restriction that all of the script packages for one part have to run before any script packages for other parts. There is just one set of constraints that is used to order all of the script packages in the pattern.

Script package ordering only applies, however, to script packages that were added to the catalog manually. You cannot specify when the internal scripts run. An example of an internal script is one that was added using 'Advanced Options'. The ordering also does not apply to scripts that are defined to be run 'at virtual system deletion' or 'when I initiate it'. And finally it does not apply to add-ons. Add-on scripts are always run before any script packages.



Ordering script packages - Default order (example)

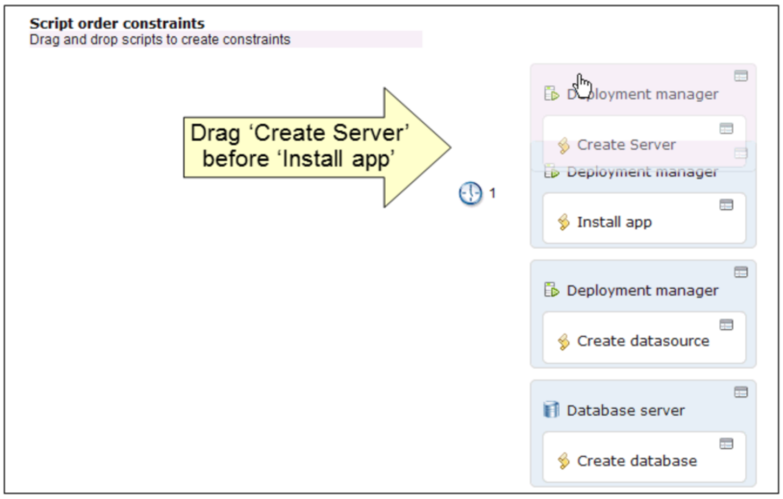

Script order constraints
Drag and drop scripts to create constraints

23 Virtual system pattern - Script packages © 2013 IBM Corporation

This slide is a continuation of the example and discusses the default script package order, which is partially based on the part start-up order. If the part start-up order specifies that the deployment manager part starts before the database server part, then the default script package order is as shown on this slide. If, instead, the part start-up order specifies that the database server part starts before the deployment manager part, then the 'Create database' script is first in this order.

There are currently no constraints defined. You will notice that there is just one number, the number one, next to the 'Install app' script package. This indicates that all of the script packages are considered to be in the same 'group'. Since there are no constraints, the script packages in the group can start in any order.

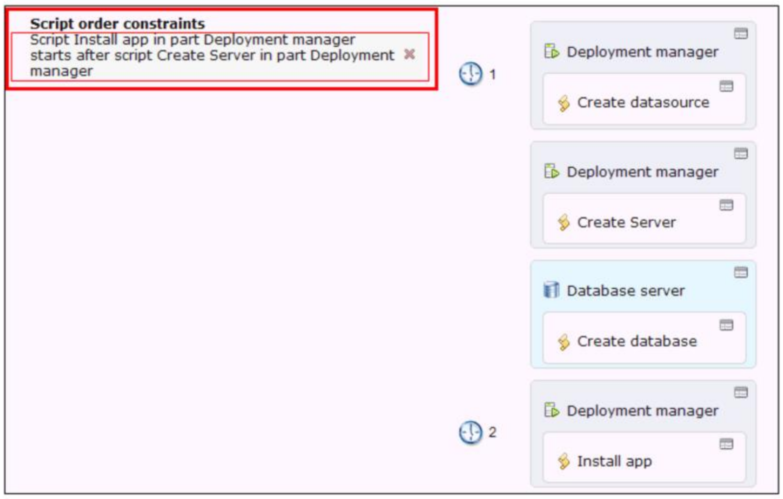
Ordering script packages – Changing the order



You can use drag-and-drop functionality to create constraints that determine the order in which the script packages are run.

In this example, the server needs to be defined before the application is installed. Therefore, you need to drag the 'Create Server' script package and drop it above the 'Install app' script package.

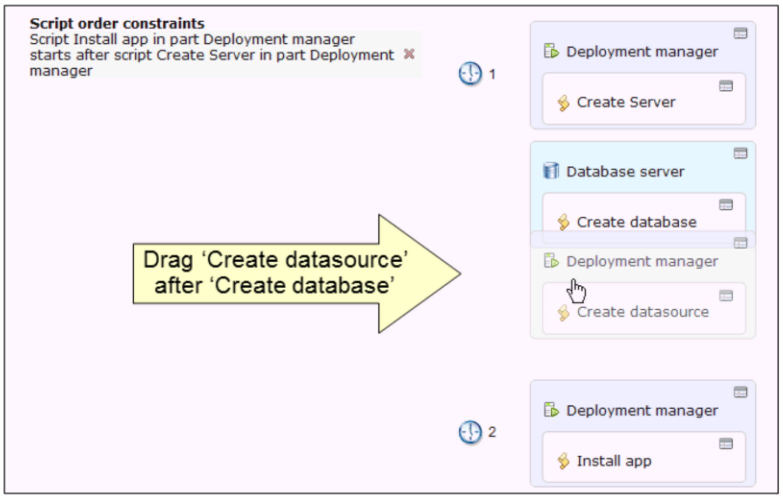
Ordering script packages – One constraint


Script order constraints

Script Install app in part Deployment manager starts after script Create Server in part Deployment manager

That creates a script package order constraint, as seen in the top left corner of the picture. Notice the wording of the constraint. The constraint will ensure that the ‘Install app’ script package runs after the ‘Create server’ script. This constraint is implemented through the creation of a new group, indicated by the number two next to the ‘Install app’ script package. The other script packages remain in group one and can still run in any order.

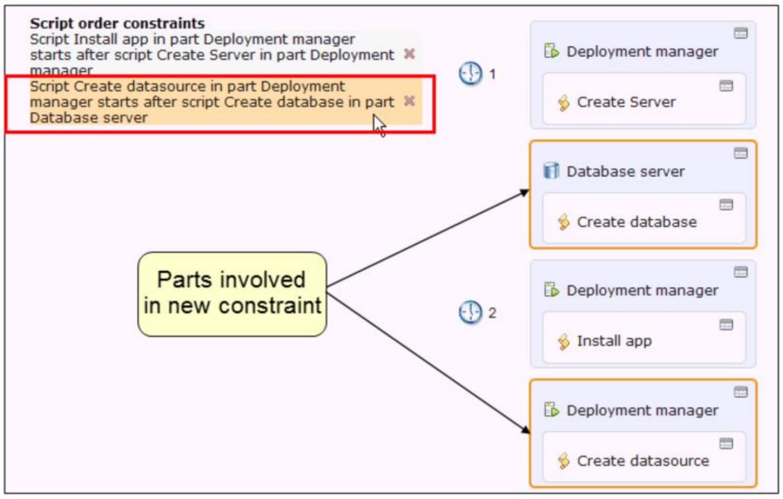
Ordering script packages – Adding another constraint



Script order constraints
Script Install app in part Deployment manager starts after script Create Server in part Deployment manager

Next, the ‘Created datasource” script package is moved after the ‘Create database’ script package.

Ordering script packages – Script package added to existing group

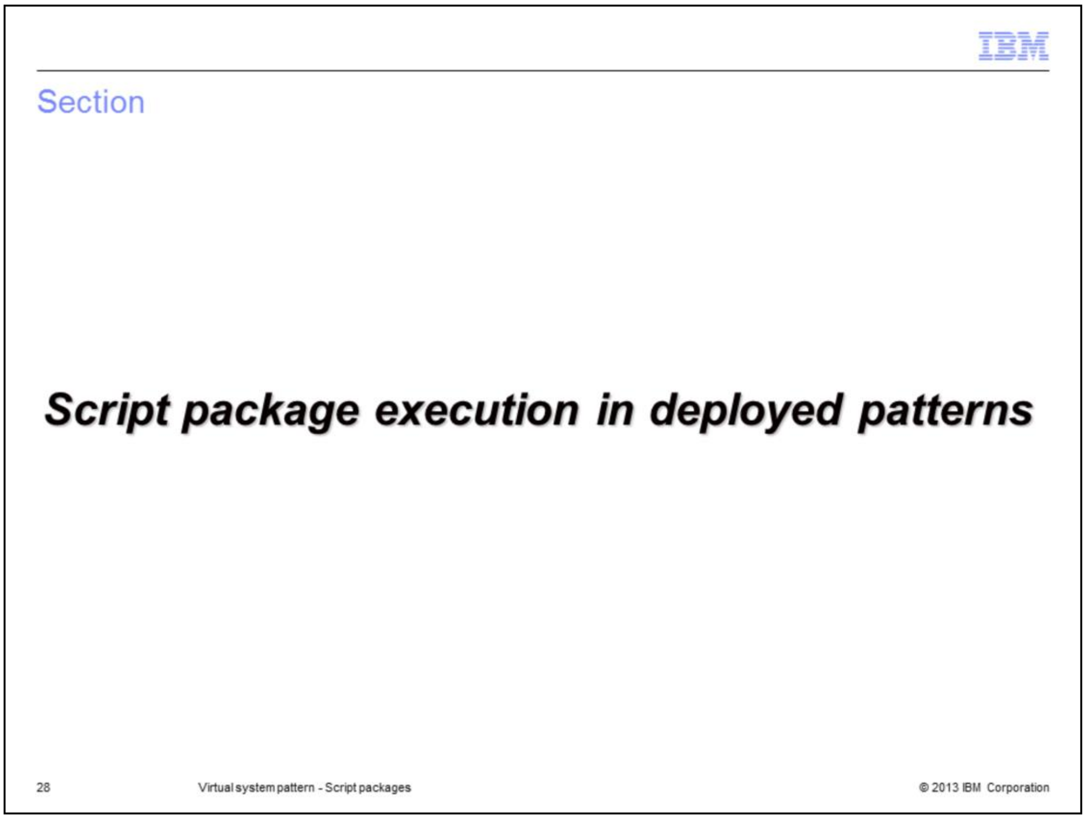

Script order constraints

Script Install app in part Deployment manager starts after script Create Server in part Deployment manager

Script Create datasource in part Deployment manager starts after script Create database in part Database server

27 Virtual system pattern - Script packages © 2013 IBM Corporation

This creates a second constraint that says that ‘Create datasource’ will run after ‘Create database’. But notice that no new group is created. There is no need for a third group because the first constraint does not involve either of the parts in the second constraint.

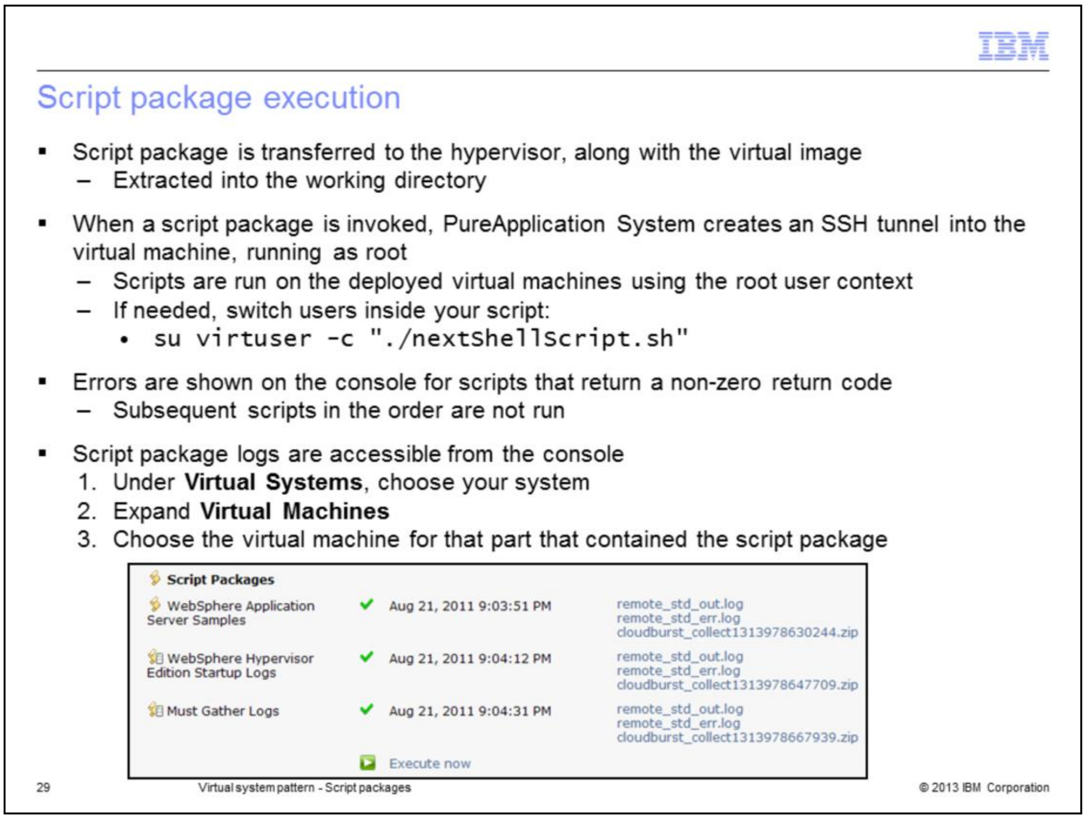


Section

Script package execution in deployed patterns

28 Virtual system pattern - Script packages © 2013 IBM Corporation

The next slide describes how script packages are run in deployed patterns.



Script package execution

- Script package is transferred to the hypervisor, along with the virtual image
  - Extracted into the working directory
- When a script package is invoked, PureApplication System creates an SSH tunnel into the virtual machine, running as root
  - Scripts are run on the deployed virtual machines using the root user context
  - If needed, switch users inside your script:
    - `su virtuser -c "./nextShellScript.sh"`
- Errors are shown on the console for scripts that return a non-zero return code
  - Subsequent scripts in the order are not run
- Script package logs are accessible from the console
  1. Under **Virtual Systems**, choose your system
  2. Expand **Virtual Machines**
  3. Choose the virtual machine for that part that contained the script package

Script Packages

WebSphere Application Server Samples	✓	Aug 21, 2011 9:03:51 PM	remote_std_out.log remote_std_err.log cloudburst_collect1313978630244.zip
WebSphere Hypervisor Edition Startup Logs	✓	Aug 21, 2011 9:04:12 PM	remote_std_out.log remote_std_err.log cloudburst_collect1313978647709.zip
Must Gather Logs	✓	Aug 21, 2011 9:04:31 PM	remote_std_out.log remote_std_err.log cloudburst_collect1313978667939.zip

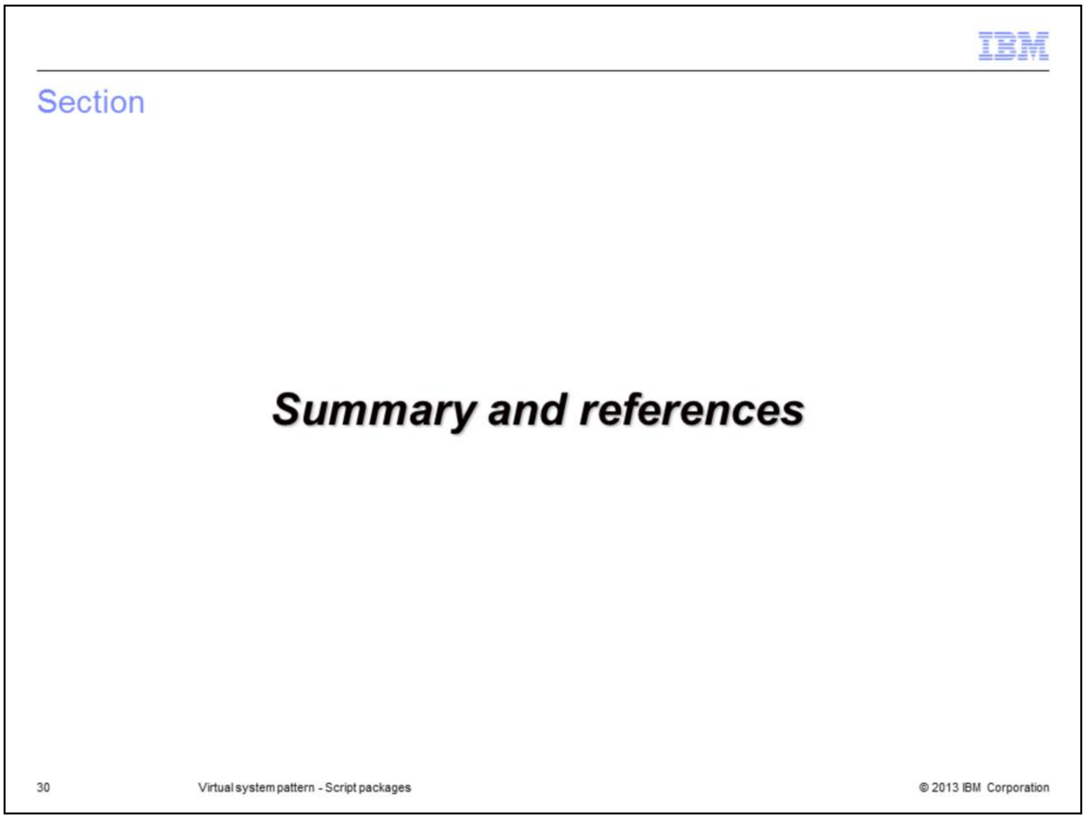
Execute now

29 Virtual system pattern - Script packages © 2013 IBM Corporation

The script package is transferred to the hypervisor, along with the virtual image, and extracted into the working directory that you specified. When it is time for the script package to run, PureApplication System creates an SSH tunnel into the virtual machine and logs in as the root user. If you want the commands in your script package to run as a different user, you should switch to that user inside your script using a command like the one shown on this slide.

For scripts that return an error, which is indicated by a non-zero return code, the error is displayed in the console and subsequent scripts in the order are not run.

After all your script packages have run, the logs for them are available on the Virtual Systems page, under the Virtual Machines section. An example is shown on this slide.

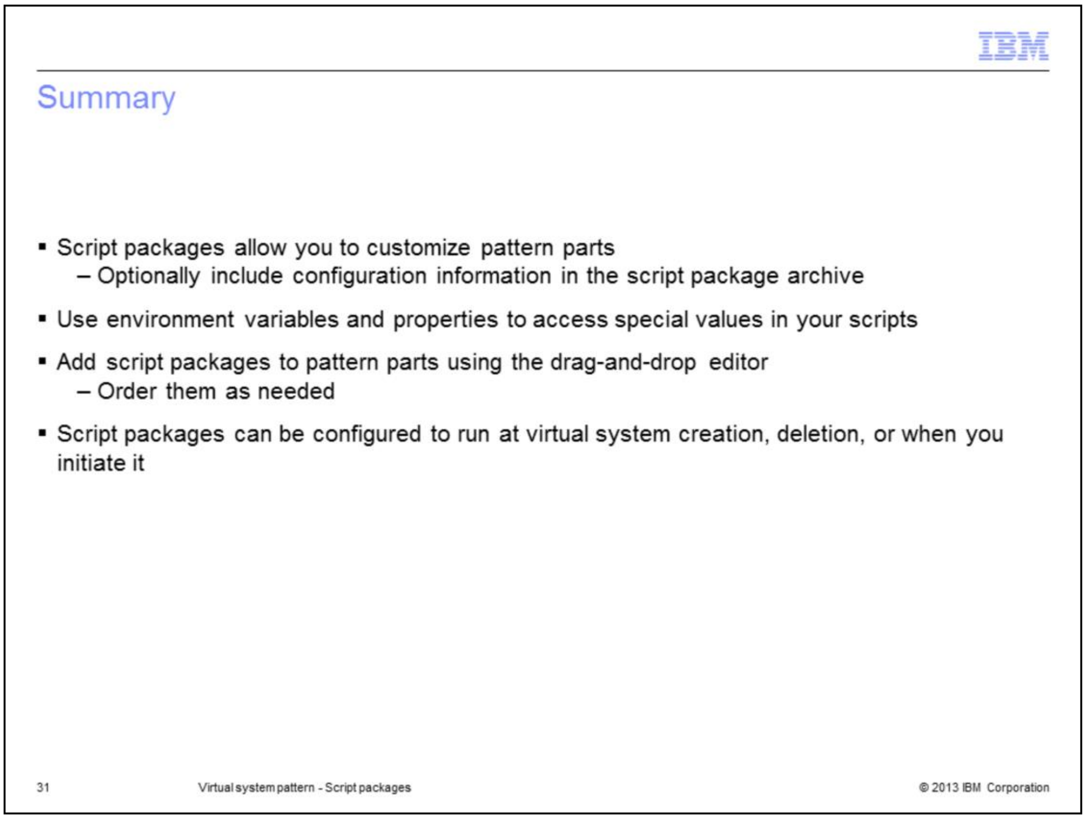


Section

Summary and references

30 Virtual system pattern - Script packages © 2013 IBM Corporation

This section provides a summary and some additional references.

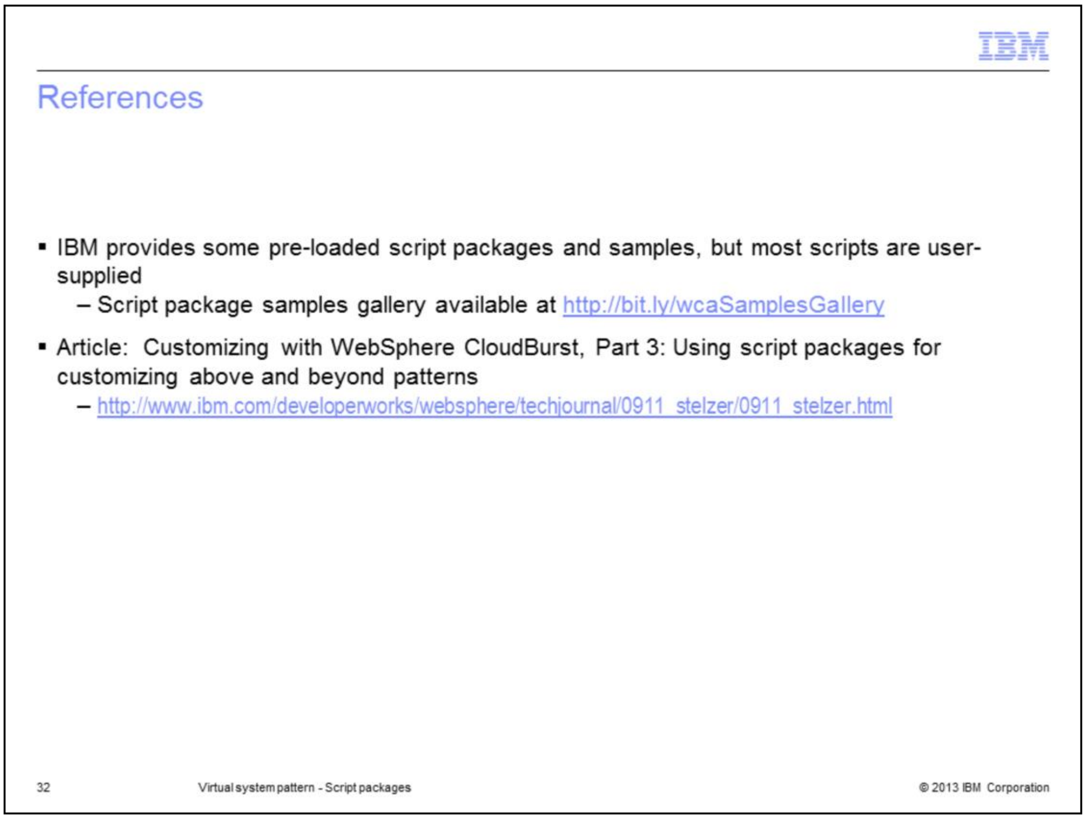


Summary

- Script packages allow you to customize pattern parts
  - Optionally include configuration information in the script package archive
- Use environment variables and properties to access special values in your scripts
- Add script packages to pattern parts using the drag-and-drop editor
  - Order them as needed
- Script packages can be configured to run at virtual system creation, deletion, or when you initiate it

31 Virtual system pattern - Script packages © 2013 IBM Corporation

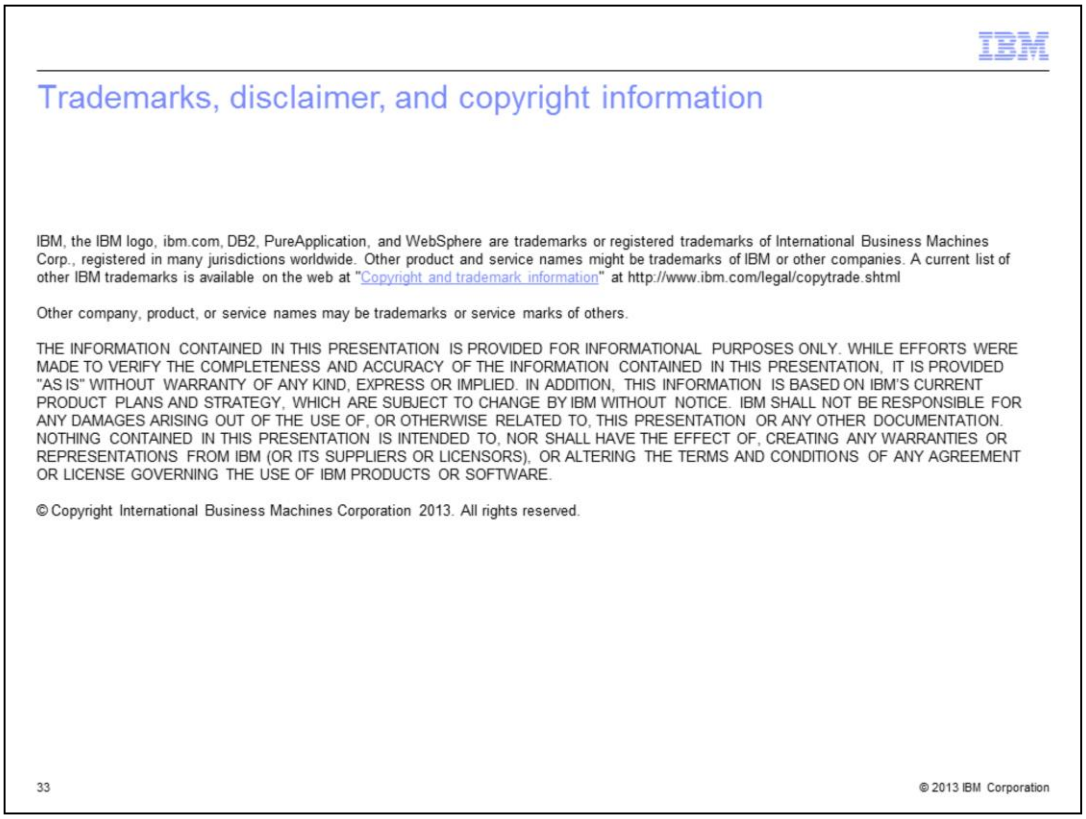
This presentation covered the complete life cycle of a script package. You were shown how to create a script package and some of the predefined variables that you can use in your script packages. You learned how to add the script package to the catalog and specify whether it should run at virtual system creation, deletion or only when you explicitly initiate it. You were shown how to include your script package in an existing pattern and specify the order that the script packages should run. Finally you learned how the script package runs at pattern deployment time.


References

- IBM provides some pre-loaded script packages and samples, but most scripts are user-supplied
  - Script package samples gallery available at http://bit.ly/wcaSamplesGallery
- Article: Customizing with WebSphere CloudBurst, Part 3: Using script packages for customizing above and beyond patterns
  - http://www.ibm.com/developerworks/websphere/techjournal/0911_stelzer/0911_stelzer.html

32 Virtual system pattern - Script packages © 2013 IBM Corporation

The samples gallery contains examples that can help you get started creating your own script packages. The URL is shown on this slide.


Trademarks, disclaimer, and copyright information