

Managing CICS Resources in a Unix File System: Best Practices

Chris Hodgins
IBM

Wednesday 14th August
Session 13375



Agenda

- CICS Resources with zFS artefacts
- zFS & security
- Best practices
- Bundles
- Cloud

SIT Parameters that point at zFS resources

- The **USSHOME** system initialization parameter specifies the name and path of the root directory for CICS® Transaction Server files on z/OS® UNIX
- The **JVMPROFILEDIR** system initialization parameter specifies the name (up to 240 characters long) of a z/OS® UNIX directory that contains the JVM profiles for CICS®. CICS searches this directory for the profiles it needs to configure JVMs.
 - The default value of **JVMPROFILEDIR** is relative to **USSHOME**

CICS region HOME directory

- CICS region userid – OMVS segment
 1. Give a z/OS UID to each CICS region user ID.
 2. Set up a *home directory* on zFS for each of your CICS regions (/u/<cicsuser>)
 3. Choose a z/OS UNIX GID for the RACF group, and assign the GID to the RACF group .
 4. Make sure that each CICS region user ID connects to the RACF group that you chose.
- Home directory is available as the \$HOME environment variable in USS

Pipelines

- The **PIPELINE** resource specifies the name of a z/OS® UNIX file containing an XML description of the nodes and their configuration.
 - **CONFIGFILE**
 - Specifies the name of a z/OS® UNIX file that contains information about the processing nodes that will act on a service request, and on the response.
 - **SHELF**
 - An absolute directory used to store the results of a pipeline scan
 - Can be shared by multiple CICS regions
 - Non-absolute paths are relative to `/var/cicsts/<region>`
 - CICS region USERID must have r+w+x permissions to the shelf
 - **WSDIR**
 - *Web service binding directory* (also known as the *pickup directory*) containing wsbind files. Used on a pipeline scan to populate the shelf.
 - CICS region USERID must have read access

Webservices

- WEBSERVICE resources are often dynamically created using a pipeline scan
- The **WEBSERVICE** resource specifies following zFS files:
 - **WSBIND** - Web service binding file.
 - **WSDLFILE** - Web service description (WSDL) file on z/OS UNIX. Used only when runtime validation is active
 - **ARCHIVEFILE** – ZIP archive of wsdl files.

Java

- JVM Profiles
 - zFS file that describe the attributes of the JVM pool (pre CICS TS V5.1) and JVMServers (V5.1 onwards)
- Java .class files
 - From CICS TS V3.2 onwards, the standard class path is constructed in a new way.
 - CICS builds a base standard class path for the JVM using the /lib subdirectories of the directories specified by the **CICS_HOME** and **JAVA_HOME** options in the JVM profile. This standard class path contains the JAR files supplied by CICS and by the JVM. It is not visible in the JVM profile.

Java

- JAR files
 - Archive containing Java class files
 - Traditionally listed on the CLASSPATH but in V4.2 has been superseded by the OSGi deployment mechanism

- OSGi Bundle JARs
 - JAR file with additional OSGi metadata
 - Deployed via a BUNDLE resolves

URIMAP

- The **URIMAP** resource specifies following zFS file:
 - **HFSFILE** - zFS file that forms the body of a static response to a HTTP request from a Web client
 - Fully qualified or relative to home directory

BUNDLES

- The **BUNDLE** resource specifies:
 - **BUNDLEDIR** specifies the root directory for the bundle contents on zFS

Atom

- The ATOMSERVICE resource specifies:
 - **BINDFILE** Specifies the fully qualified (absolute) or relative name of an XML binding stored in z/OS® UNIX System Services. For resource types FILE and TSQUEUE, the XML binding is required, for resource type PROGRAM an XML binding is optional.
 - **CONFIGFILE** The Atom configuration file contains XML that specifies metadata and field names for the Atom document
 - **ALTHOUGH** the Explorer builds and packages all these parts as a bundle for you.

Doctemplates

- The **DOCTEMPLATE** resource specifies:
 - **HFSFILE** - When the template resides in zFS
 - Fully qualified or relative to home directory

zFS & Security

File permissions in zFS

- Use the UNIX permission flags for Owner, Group and All to control access to your CICS resources on zFS

Here's an example of an entry you might see if you listed contents of a zFS directory

- `drwxr-x---` 2 SYSADMIN CICS 8192 May 10 14:52 MyBundle/
- CICS zFS files may require access via 3 classes of user:
 1. CICS regions - region userid
 2. CICS system administrators (humans)
 3. Code management systems (tools)

File permissions in zFS

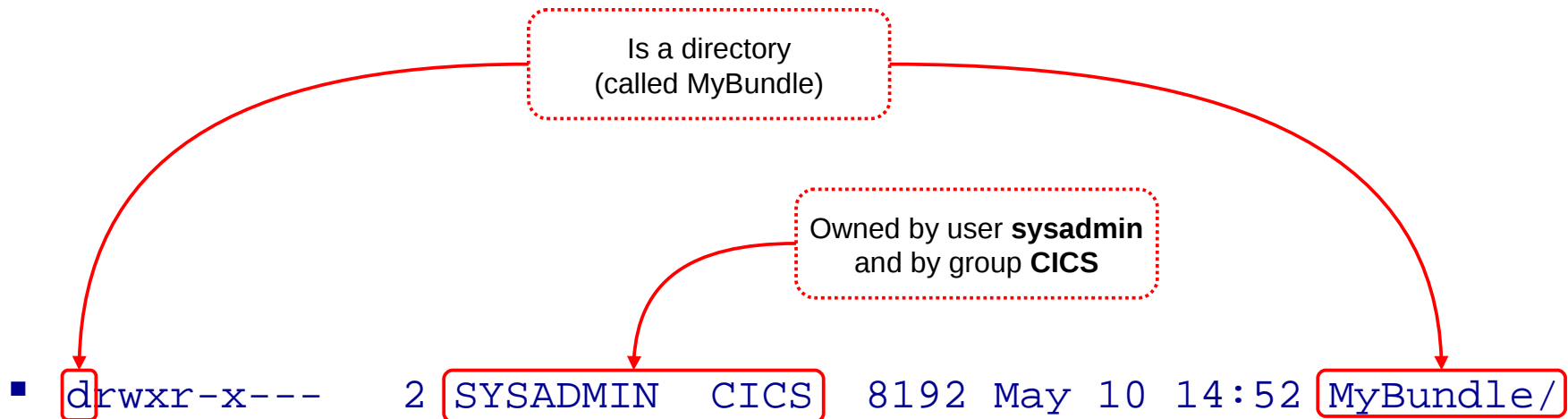
- Use the UNIX permission flags for Owner, Group and All to control access to your CICS resources on zFS

Is a directory
(called MyBundle)

- `drwxr-x---` 2 SYSADMIN CICS 8192 May 10 14:52 `MyBundle/`

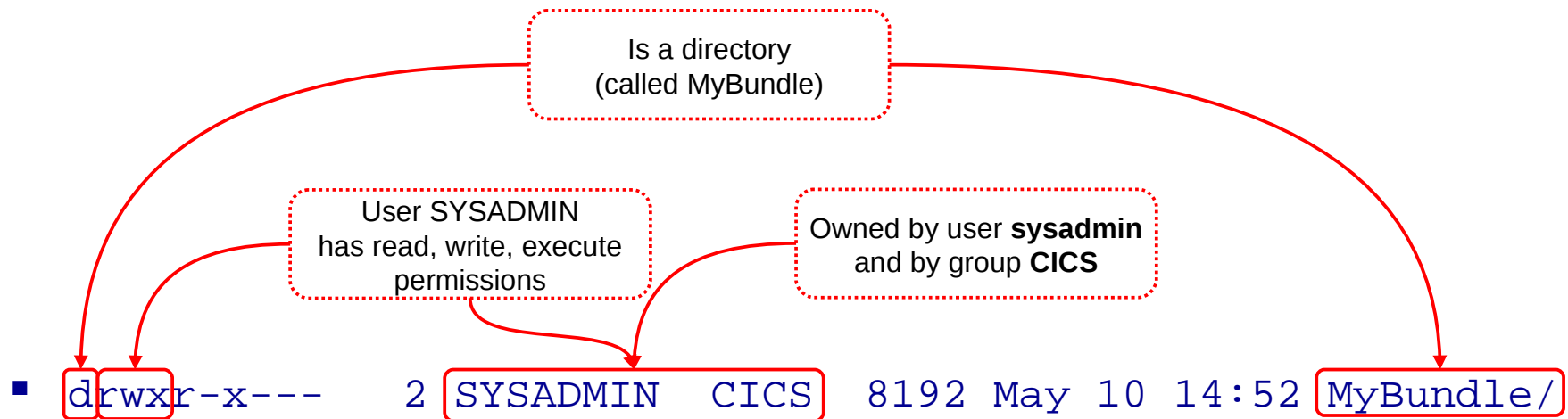
File permissions in zFS

- Use the UNIX permission flags for Owner, Group and All to control access to your CICS resources on zFS



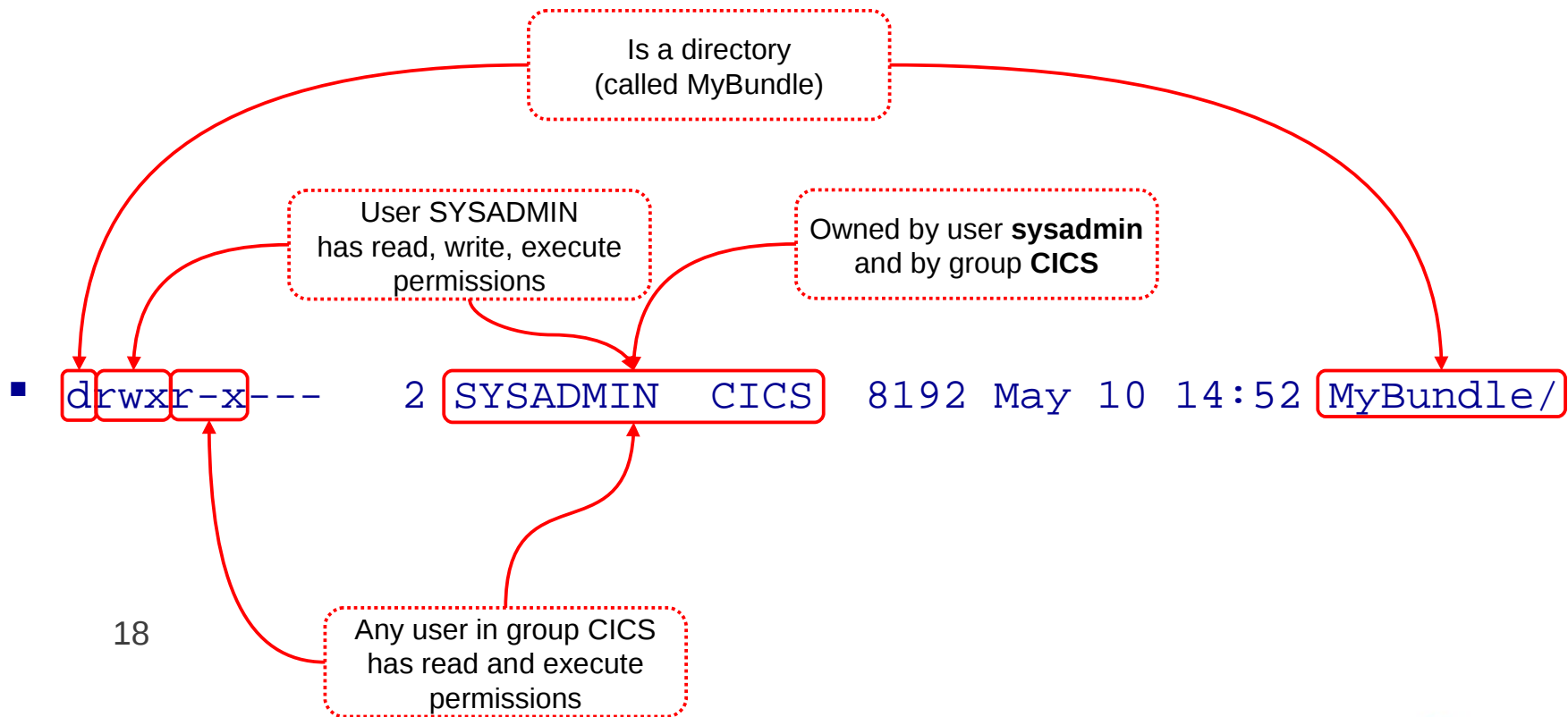
File permissions in zFS

- Use the UNIX permission flags for Owner, Group and All to control access to your CICS resources on zFS



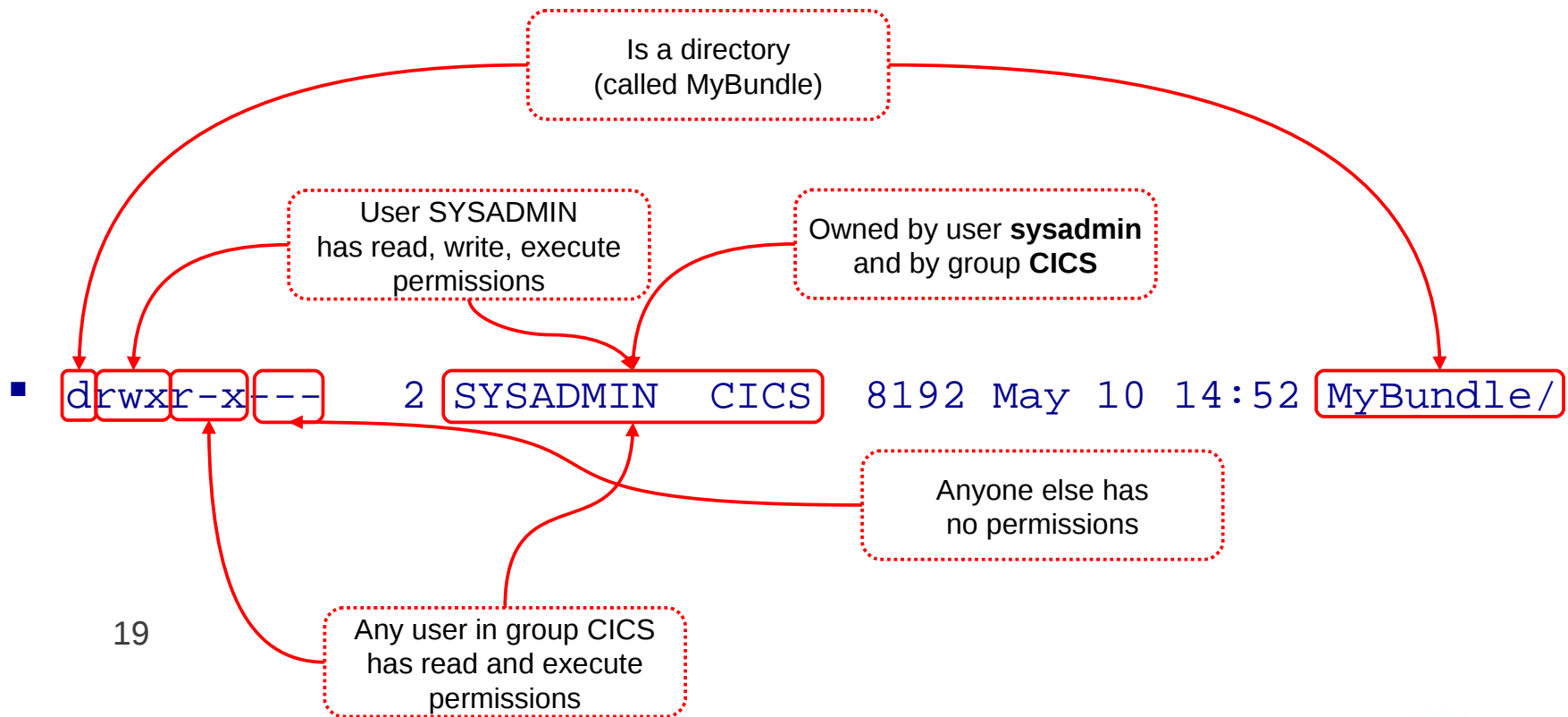
File permissions in zFS

- Use the UNIX permission flags for Owner, Group and All to control access to your CICS resources on zFS



File permissions in zFS

- Use the UNIX permission flags for Owner, Group and All to control access to your CICS resources on zFS



UMASK

- The file permission bits are set using the UMASK of the creating process, which signifies the bits that are not set
 - i.e a umask of 022 causes
 - - Directories to be created with 755 (rwxr-xr-x) permission bits
 - - Files to be created with 644 (rw-r--r--) – as by default x permissions are not given for files

UNIX security rules

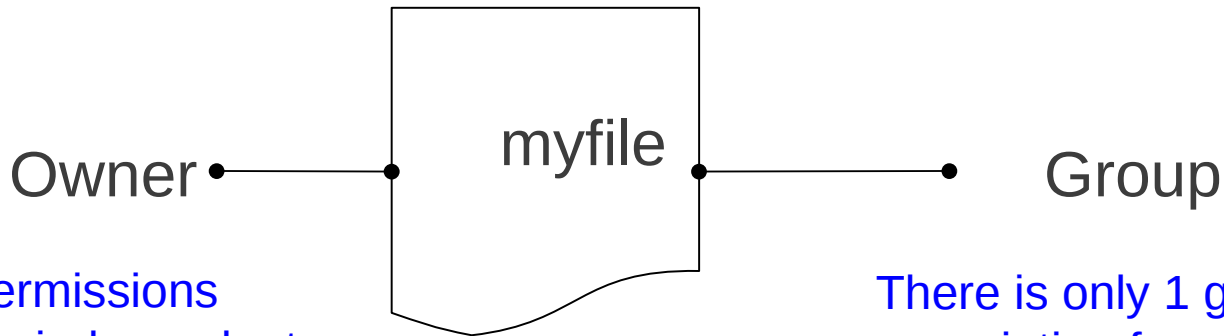
- Permission bits set via 3 bits controlling rwx
- 3 classes of user: *Owner, group, other*
- Octal notation commonly used (i.e 775=rwxrwxr-x)

- A directory must have r permissions for it to be read or traversed
- A directory must have rx permissions for it to be searched
- A file must have r permission for it to be read
- A file must have rw permission for it to be updated
- A file can be deleted by the owner regardless of permissions.

- When creating a file
 - The uid is set to the uid of the creating user
 - The gid is set to the gid of the owning directory (unless the FILE.GROUPOWNER.SETGID profile is active) when this will be the gid of the creating user
 - The file permission bits are set using the UMASK of the creating process

The multiple writers problem...

- A user can be in many groups...
...but a file has only one set of group permission bits



Granted permissions
over myfile, independent
of the group

There is only 1 group
association for a file. You can
not associate 2 logical groups
with the file, using standard
unix permissions.

Extending Security - File ACLs

- ACLs provide a solution to the multiple group writers problem
- They allow a more flexible model
 - Multiple groups can have different permissions over a file
 - ACL inheritance can be controlled
 - Permissions still limited to read/write/execute
- You need to enable ACLs
 - SETROPTS CLASSACT(FSSEC)
 - If the FSSEC class is not active, the standard POSIX permission bit checks are done, even if an access ACL exists
 - Control using `setfacl` USS command
- ACLs Link : **zOS ACLs 1.13**

zFS Best practices

- Define usage for all zFS files
 1. Temporary file system
 - – Not critical for production, but may need to be periodically emptied
 - - Java WORK_DIR or PIPELINE Shelf
 2. Install file system – /usr/lpp/
 - Read only and usually shared with all products
 - Don't store config or temporary files in here
 - USSHOME and JAVA_HOME only
 3. Critical file system – code and configuration files
 - Any files that would cause a production outage if lost
 - Wsdl, wsbind files, JVM profiles, JARs, OSGi bundles, Bundledirs

- Define roles and assign groups as necessary
 1. CICS region userids
 2. System administrators
 3. Tooling/automation

CICS zFS best practice examples

Resource	Attribute	Description	Type	CICS region permissions	Administrator permissions
SIT	USSHOME	CICS zFS install location	Install	r	r
PIPELINE	Configfile	XML configuration file for the pipeline processing	Critical	r	r
PIPELINE	Wsdir	Web service binding directory	Critical	r	r
PIPELINE	Shelf	Shelf directory used on a pipeline scan	Temporary	rw	r
BUNDLE	Bundledir		Critical	r	r
PROGRAM	JVMPROFILE		Critical	r	rw
JVMPROFILE	WORK_DIR	Java logs and dumps	Temporary	rw	rw
JVMPROFILE	JAVA_HOME	Java zFS install location	Install	r	r
JVMPROFILE (Liberty)	WLP_OUTPUT_DIR	Liberty logs	Temporary	rw	rw
JVMPROFILE (Liberty)	WLP_INSTALL_DIR	CICS zFS install location	Install	r	r

FTP access – get the defaults right

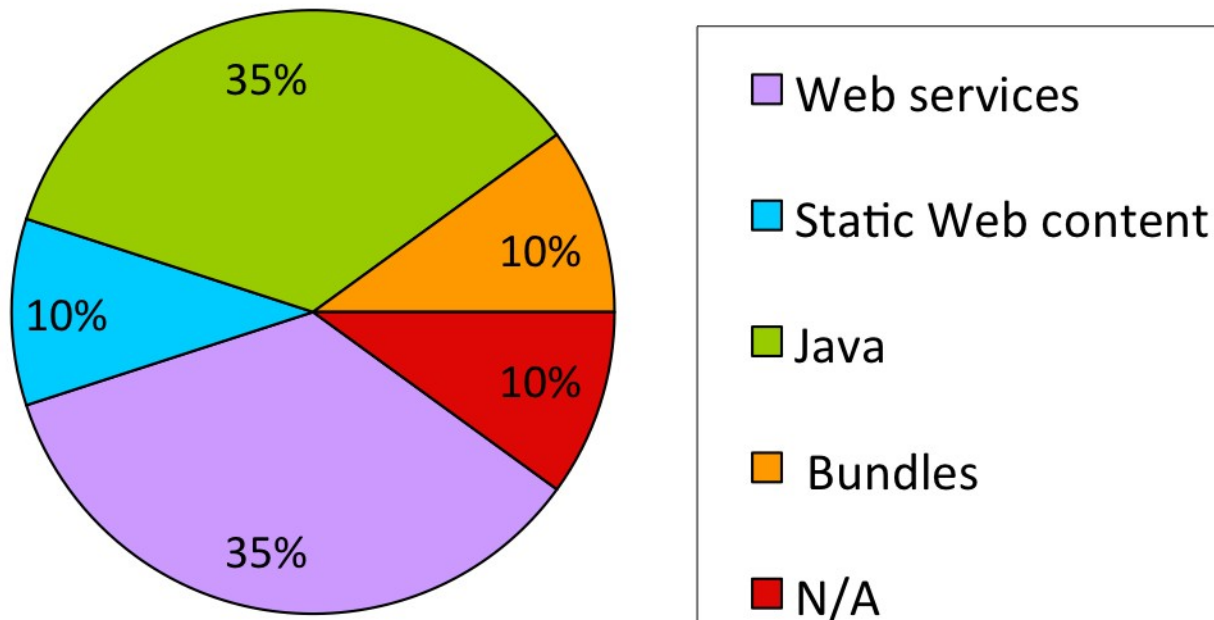
- Default FTP file permission are set using a umask
- i.e UMASK 027 sets 750 ie owner=rwx group=r-x other=---
- Set as a property in the FTP config file
 - see SYS1.TCPPARMS(FTPDATA)

Performance of zFS

- Performance of shared zFS mounted r/w filesystems has been regarded as an issue (in terms of XCF signalling costs) and function shipping of I/O between LPARs
- Solutions:
 - V1R11 provides local read caching – removing overheads for reads
 - V1R13 provides direct I/O for read and write, removing need to function ship these commands to the owning LPAR
 - Or mounting file system locally removes need to function ship I/O
 - JVM class caching provides ability to cache Java byte codes in a shared memory area (i.e within LPAR)
 - -> Requires APAR PM78799 on CICS TS V4.2 to support class caching

zFS usage survey : types of resource

Q2. What CICS application resources do you use today in HFS/zFS?

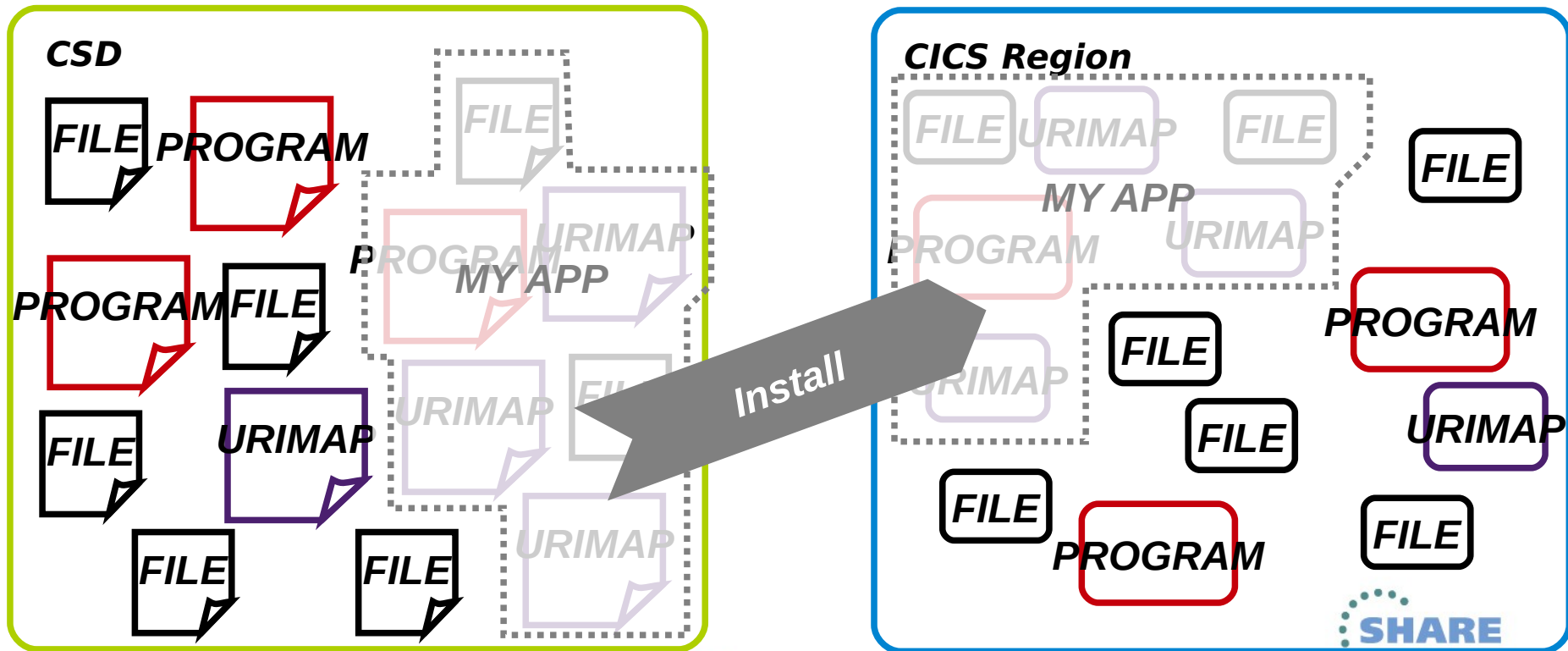


CICS TS 5.1 introduces Application and Platform resources which are packaged as bundles on zFS...

What is a CICS Bundle?

A CICS Bundle is a collection of CICS Resources that can be managed as a logical unit

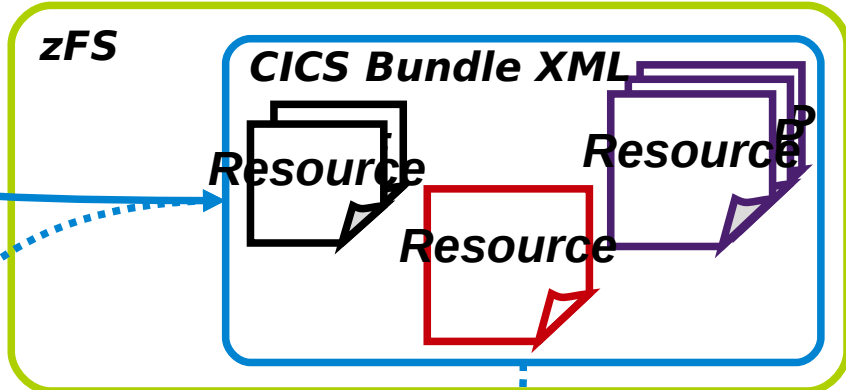
Need a way to manage an application's CICS Resources as a logical group



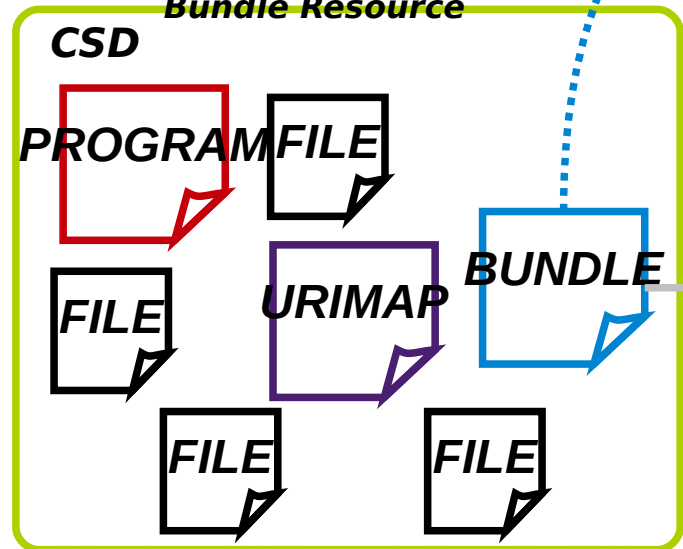
The solution is the CICS Bundle

1. Create CICS Bundle XML using the CICS Explorer

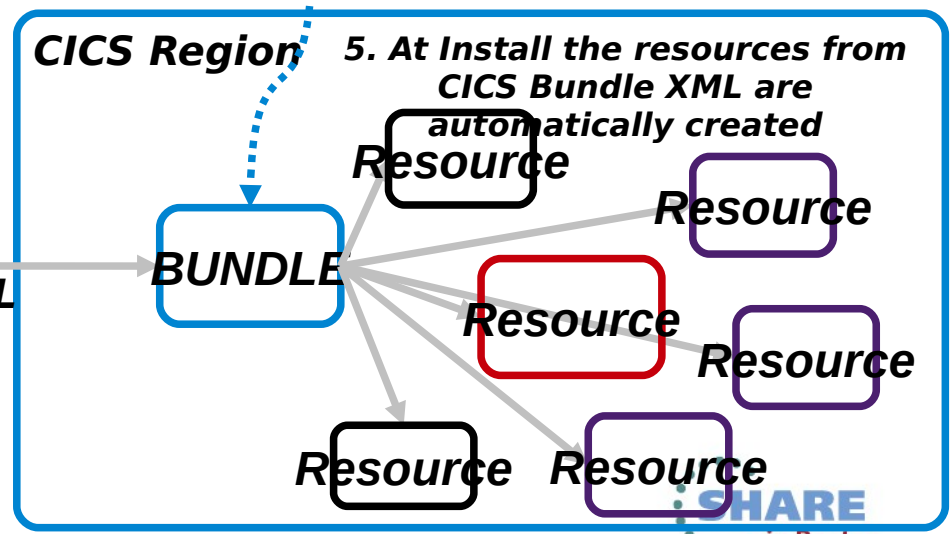
2. Export CICS Bundle XML to zFS



3. Reference the CICS Bundle XML on zFS via BUNDLEDIR attribute on a CICS Bundle Resource



4. INSTALL

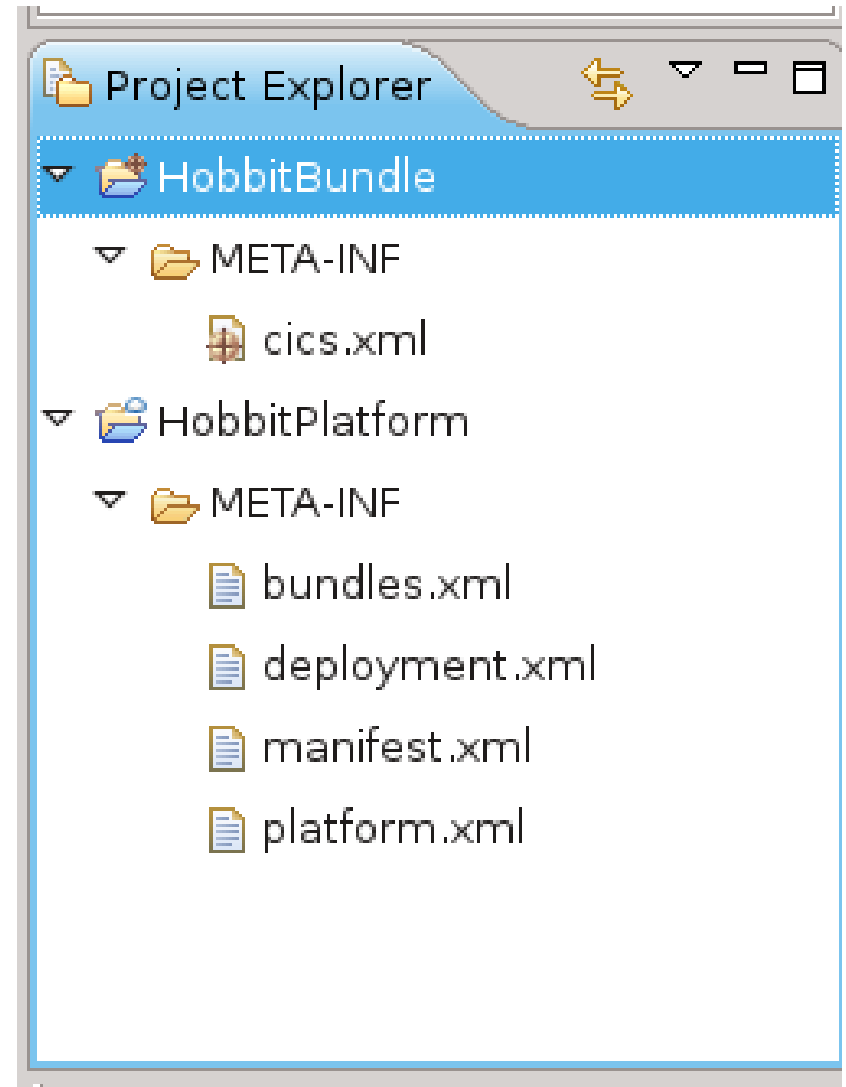


What's in a CICS Bundle?

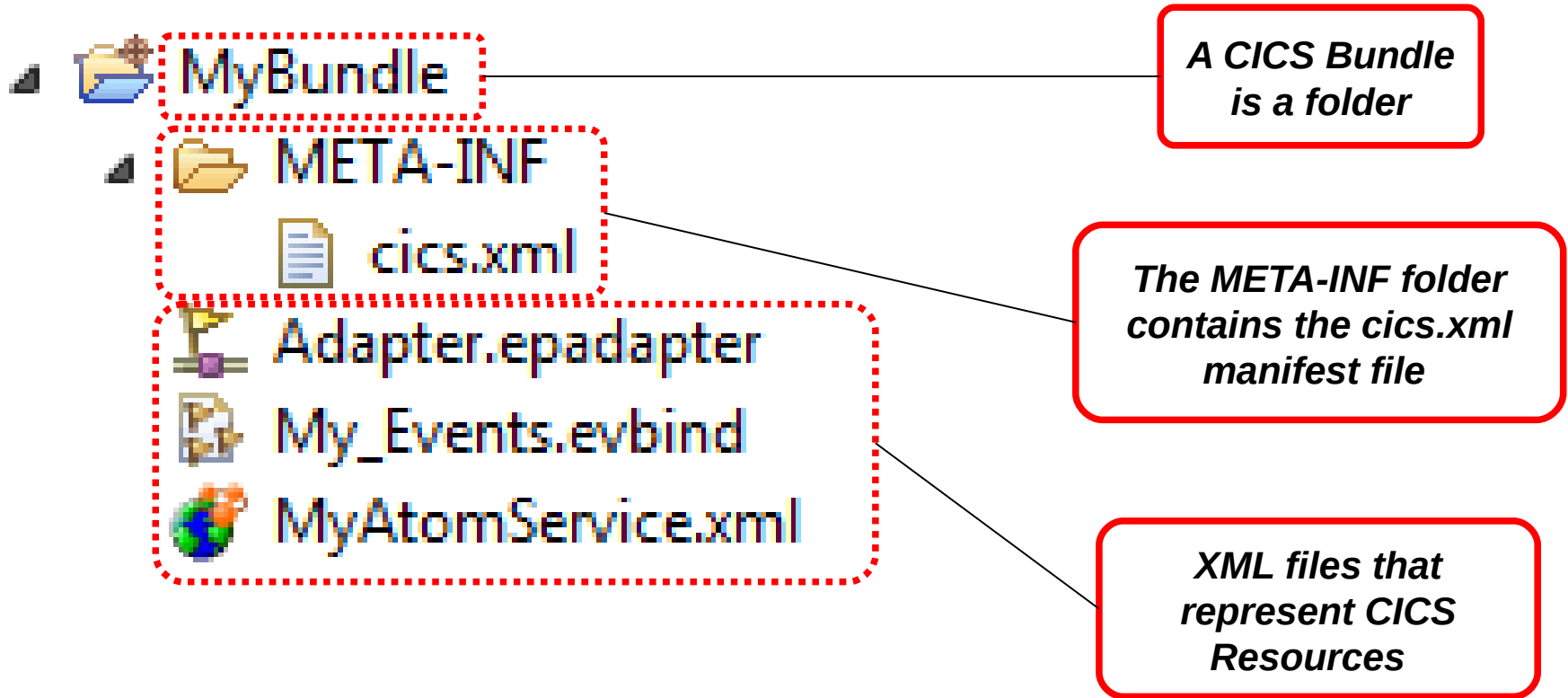
What's in a CICS Bundle?



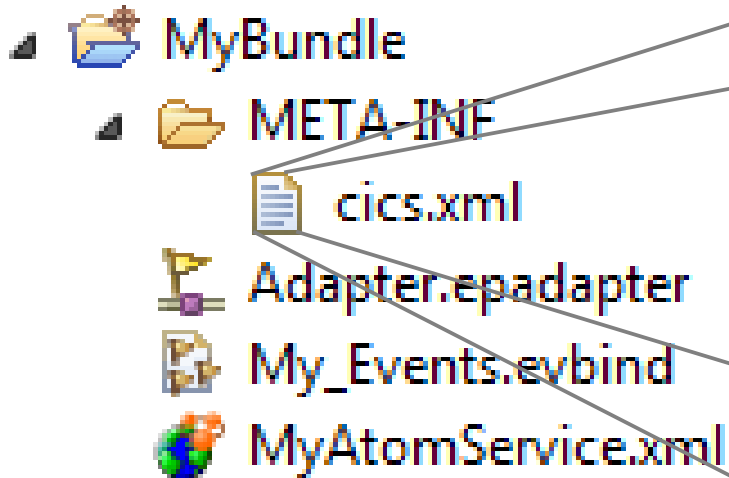
An example bundle and platform bundle as displayed in the CICS Explorer



What's in a CICS Bundle?



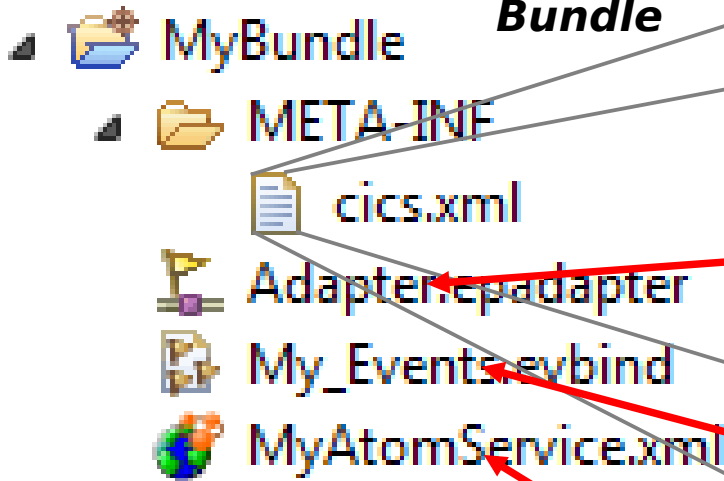
What's in the cics.xml manifest file?



```
<define name="Adapter"  
type="http://www.ibm.com/xmlns/prod/c  
ics/bundle/EPADAPTER"  
path="Adapter.epadapter"/>  
  
<define name="My_Events"  
type="http://www.ibm.com/xmlns/prod/c  
ics/bundle/EVENTBINDING"  
path="My_Events.evbind"/>  
  
<define name="MyAtomService"  
type="http://www.ibm.com/xmlns/prod/c  
ics/bundle/ATOMSERVICE"  
path="MyAtomService.xml"/>  
  
<import name="MYFILE"  
type="http://www.ibm.com/xmlns/prod/c  
ics/bundle/FILE" optional="false"  
warn="true"/>
```

What's in the cics.xml manifest file?

Definitions for all the resources in the CICS Bundle



```

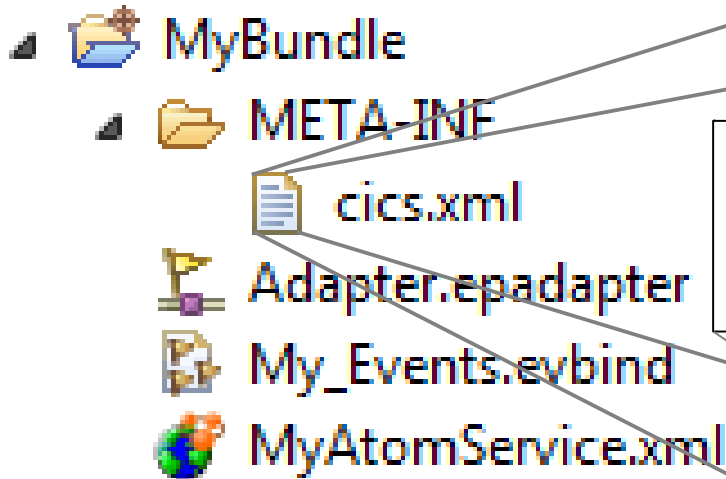
<define
name="Adapter"
<define name="Adapter"
type="http://www.ibm.com/xmlns/
prod/cics/bundle/EPADAPTER"
path="Adapter.epadapter"/>
<define name="My_Events"
type="http://www.ibm.com/xmlns/
prod/cics/bundle/EVENTBINDING"
path="My_Events.evbind"/>
<define name="MyAtomService"
type="http://www.ibm.com/xmlns/
prod/cics/bundle/ATOMSERVICE"
path="MyAtomService.xml"/>

```

The CICS Bundle will not install if any of the defined bundle parts are missing from the bundle folder

What's in the cics.xml manifest file?

Bundle Imports



```
<import name="MYFILE"
type="http://www.ibm.com/xmlns/
prod/cics/bundle/FILE"
optional="false" warn="true"/>
```

```
<define
name="Adapter"
type="http://www.
```

```
path="Adapter.epa
dapter"/>
```

```
<define
name="My_Events"
type="http://www.
```

```
ibm.com/xmlns/pro
d/cics/bundle/EVE
NTBINDING"
path="My_Events.e
vbind"/>
```

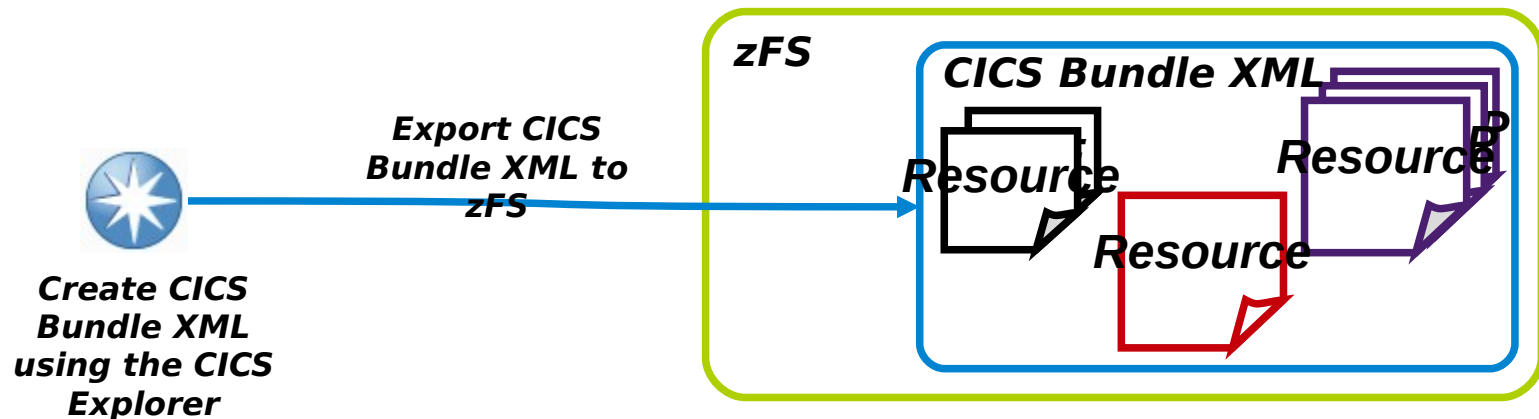
Used to articulate dependencies upon CICS Resources that are not in this bundle

Bundle Lifecycle

- Creating/Editing CICS Bundles and exporting to zFS
- Install CICS Bundles into CICS
- Enabling, Disabling and Discarding

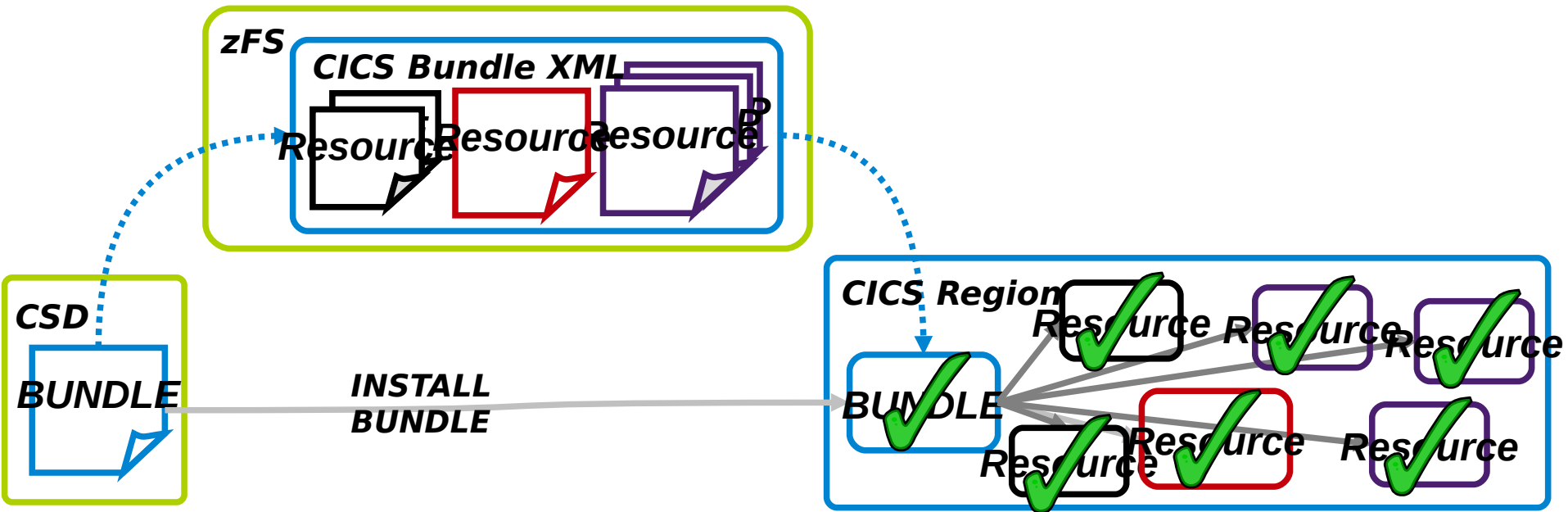
Creating/Editing CICS Bundles and Exporting to zFS

- The CICS Bundle XML is created and edited in the CICS Explorer
- The CICS Explorer **MUST** be used to export the CICS Bundle XML to zFS
- The export to zFS operation is **ONE WAY** (like compiling) you cannot import from zFS
- CICS Bundle XML should **NOT** be edited on zFS directly
- If you need to change the the CICS Bundle XML it **MUST** be re-exported using the CICS Explorer



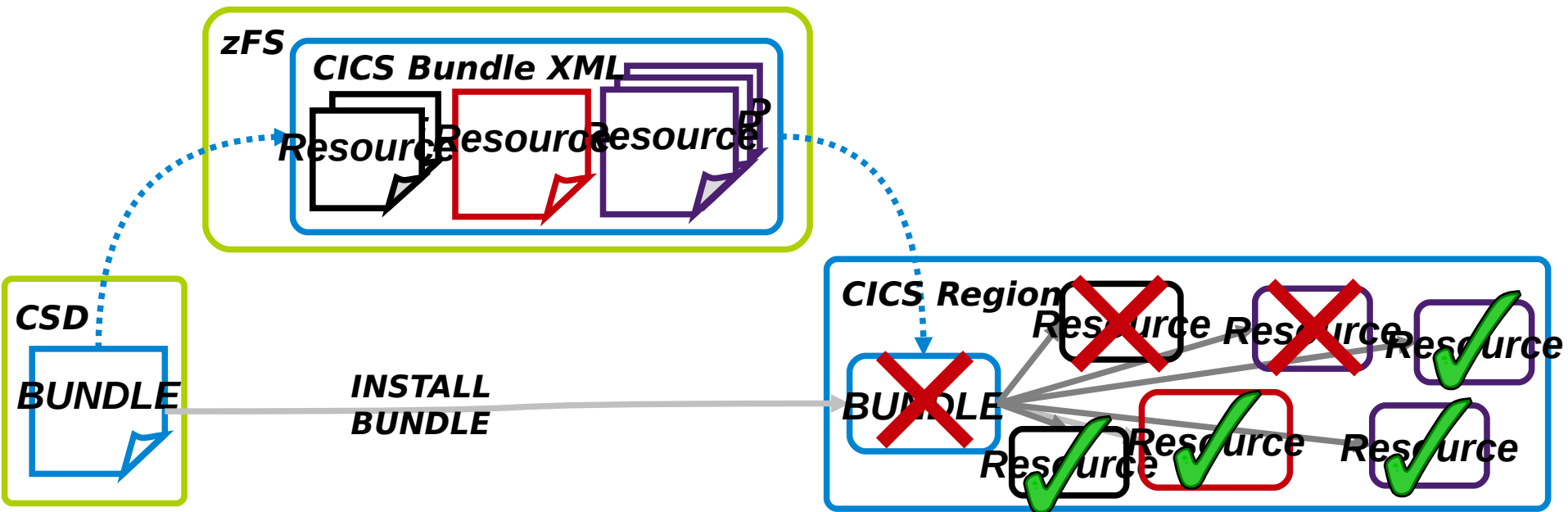
Installing CICS Bundles into CICS

- Create a BUNDLE resource definition which references the CICS Bundle XML in it's BUNDLEDIR attribute
- When the BUNDLE is installed all the resources from CICS Bundle XML are automatically created



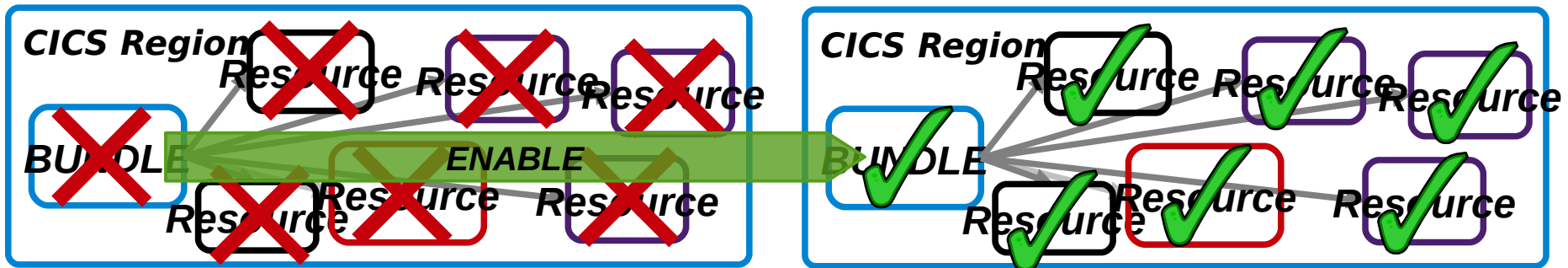
Installing CICS Bundles into CICS

- The CICS Bundle will be implicitly disabled at install if:
 - Any of it's resources fail to install
 - Any of it's resources are installed disabled
 - Any of it's imports are disabled or not present

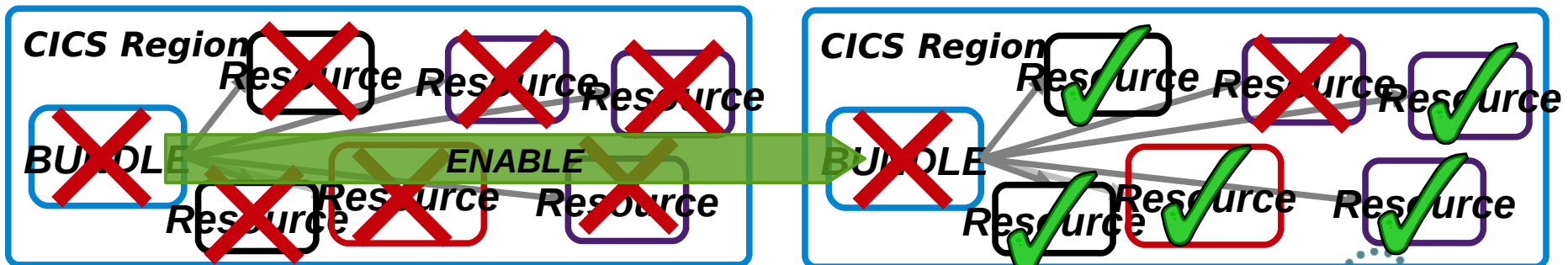


Enabling a CICS Bundle

- When a CICS Bundle is enabled explicitly it will try to enable all its Resources

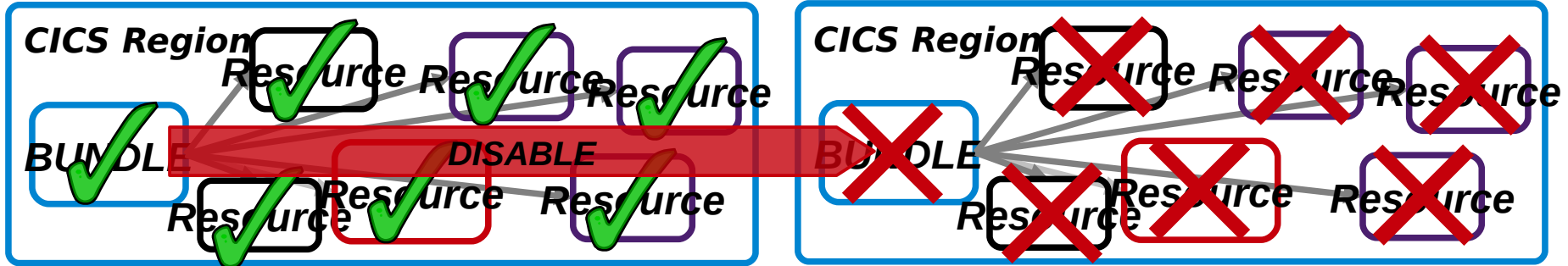


- If there are any resources that cannot be enabled then it will remain disabled

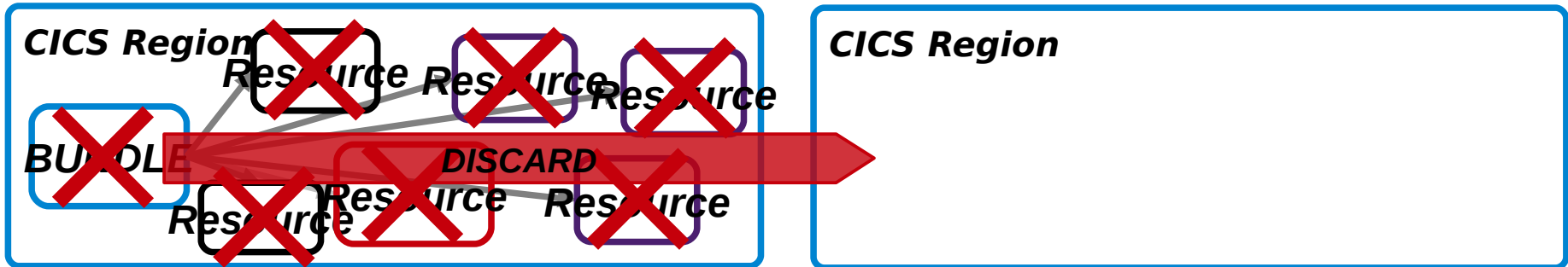


Disabling and Discarding a CICS Bundle

- When a CICS Bundle is disabled explicitly it will disable all its Resources

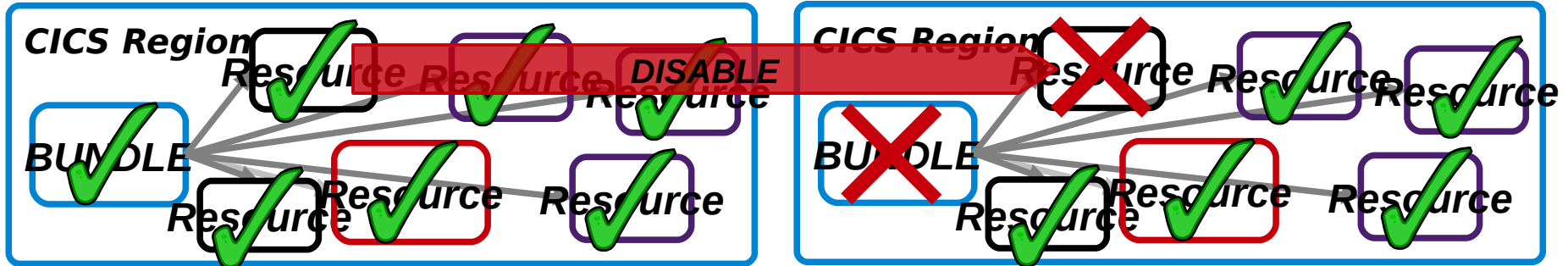


- When a CICS Bundle is discarded it will discard all of its resources

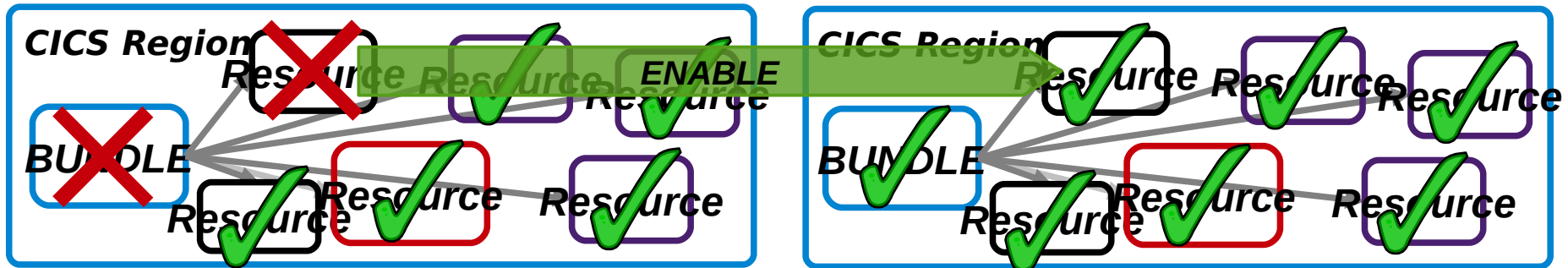


Disabling and Enabling a CICS Resource

- When a CICS Resource is disabled, any Bundles which own it or import it will be implicitly disabled



- When a CICS Resource is enabled, any Bundles which own it or import it will be enabled if they were implicitly disabled and this is the last resource stopping it from being enabled

















CICS TS V5.1 – Cloud and zFS

- Platform Definitions (CPSM resource) contains a “Platform Home”
- The Platform Home is the root directory in zFS for CICS Platforms
 - This contains deployed applications, policies, and bindings (deployed bundles – more later)
- **Best Practice for Platform Home**
 - `/var/cicsts/<cics_plex>/<platform_name>`
 - Using this means the CICS Explorer can “add value” because it knows where to find things when you’re browsing

Deploying Platforms

- One way street (same rules apply as for bundles, because it's just a different type of bundle.)
- All the MAS & CMAS that participate in the platform need to be able to read the platform home and it's subdirectories
- When deployed, the ID & version information stored in the bundle becomes part of the directory name.

Deployed platforms in zFS

- ▼  Platform1
 - ▼  applications
 - ▶  svt.application.a_1.1.1
 - ▼  bindings
 - ▶  binding_appa_plat1_1.0.0
 - ▼  bundles
 - ▶  svt.application.a.aor.binding_1.0.0
 - ▶  svt.application.a.bundle.a_1.1.0
 - ▶  svt.application.a.bundle.b_1.0.0
 - ▶  svt.application.a.bundle.c_1.0.0
 - ▶  svt.application.a.bundle.d_1.0.0
 - ▶  svt.application.a.tor.binding_1.0.0
 - ▼  platform
 - ▶  Platform1

- If you want to backup the entire deployed platform and applications, copy everything under the “platform home directory”.
- If your platform spans LPARs then use shared zFS

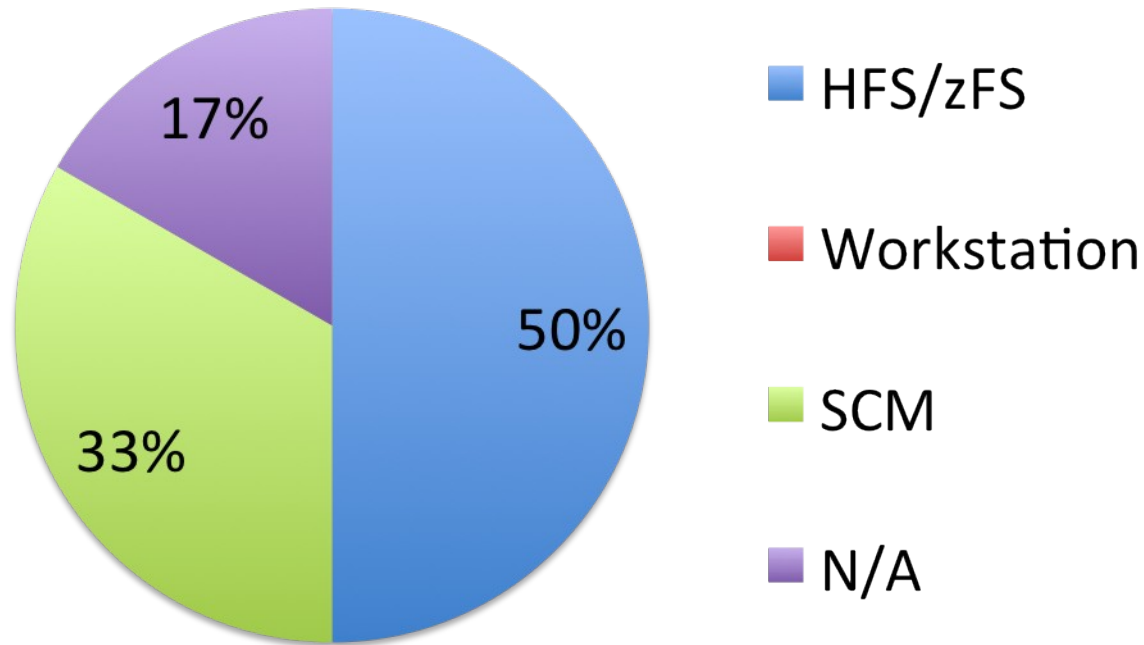
Security – deploying platforms

- The user ID signed into the zOS Explorer, doing the deploy, needs access to the FTP service and write authority to the “platform home directory”

Best practices

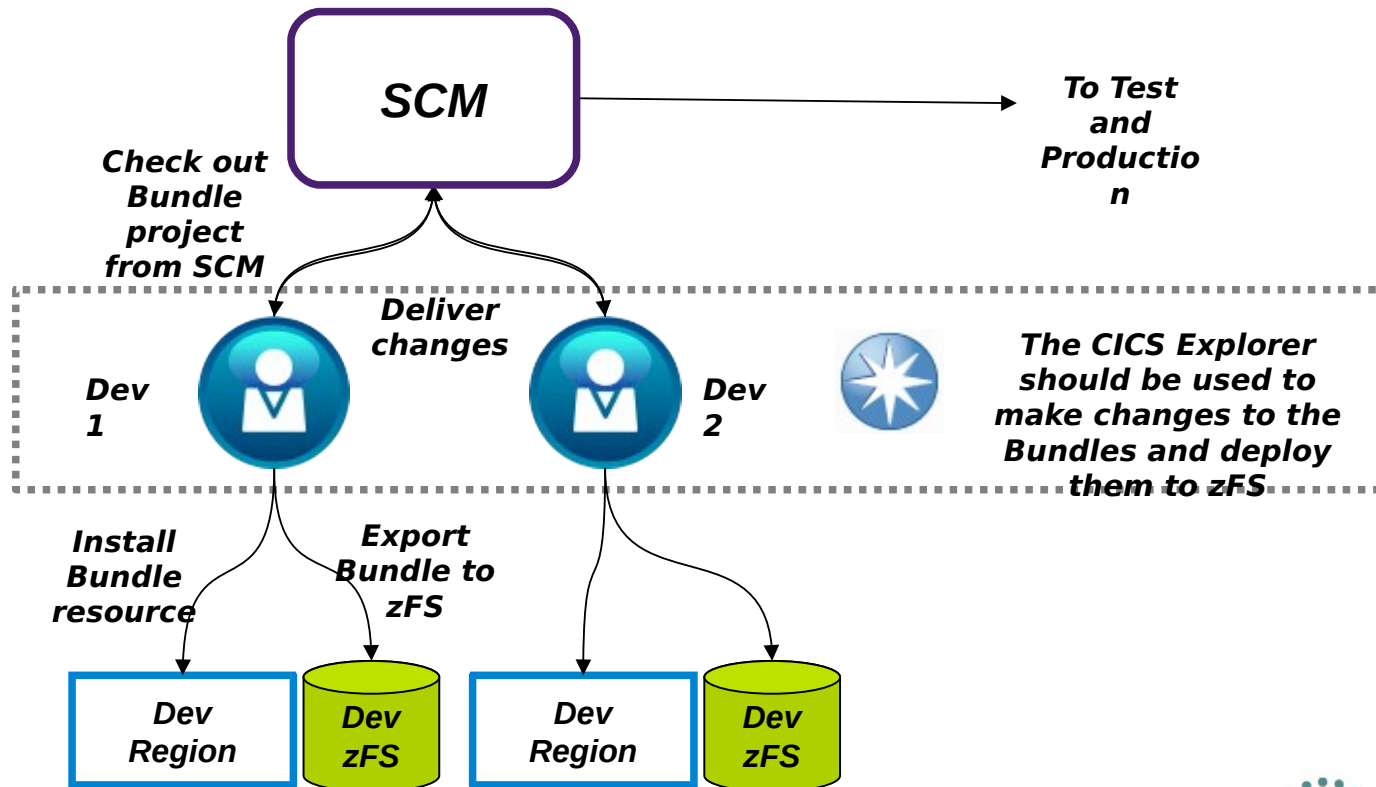
zFS usage survey...

Where do you store the master copy of CICS USS files?



Managing changes to CICS Bundles

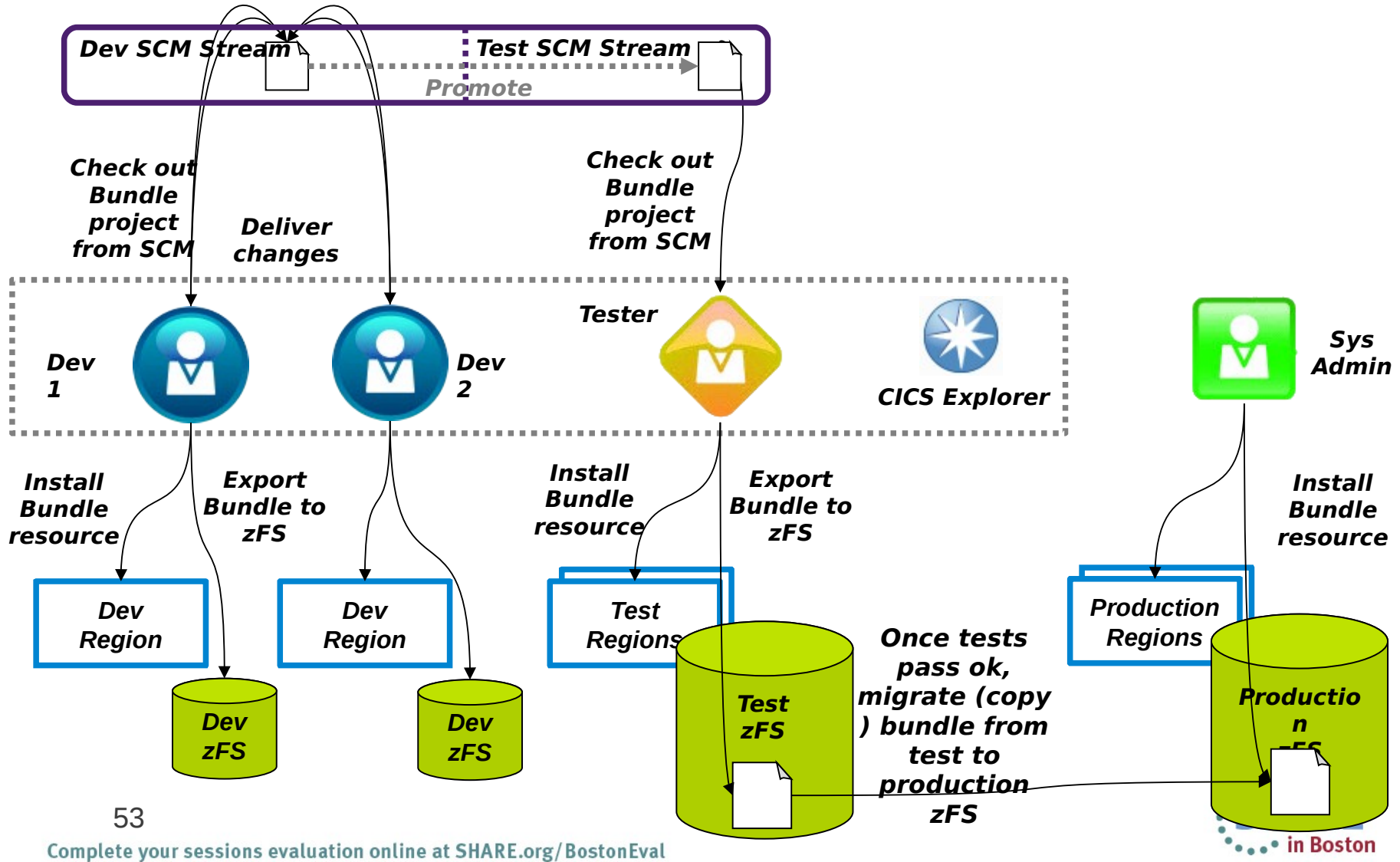
- CICS Bundle XML should be treated as **source code**
- Changes should be managed and shared using a source code management (SCM) repository



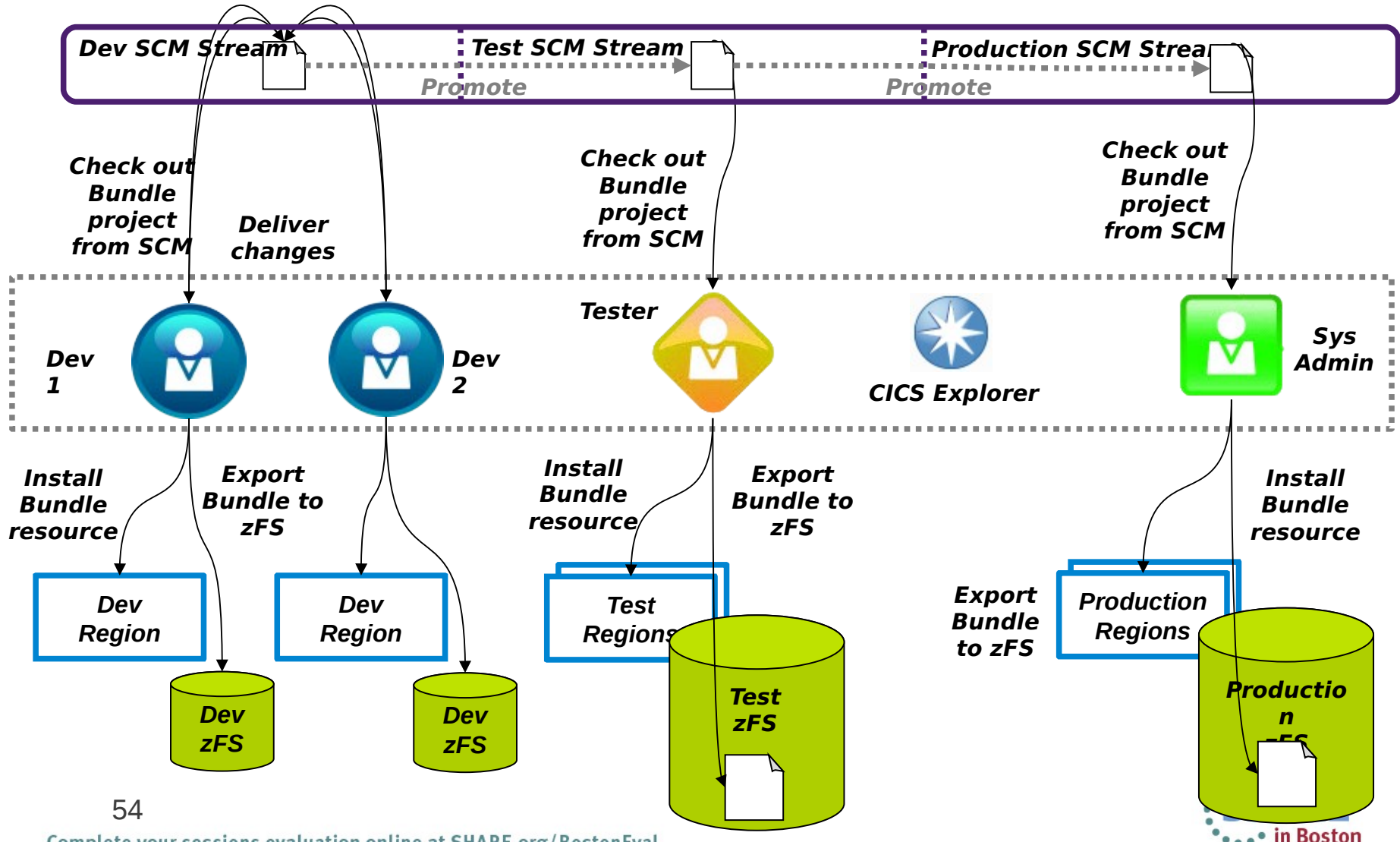
Migrating CICS Bundles from Dev to Test to Production

- BUNDLES should be treated like any other CICS resource that has a reference to an artefact that lies outside the CSD eg:
 - PROGRAMS have load modules/java classes
 - WEBSERVICES have wsbind files
- You should migrate the CICS Bundle XML **before** the BUNDLE resource
 - You wouldn't migrate a new PROGRAM resource before you migrated the load module for it!

Migrating CICS Bundle XML from Dev to Test to Production

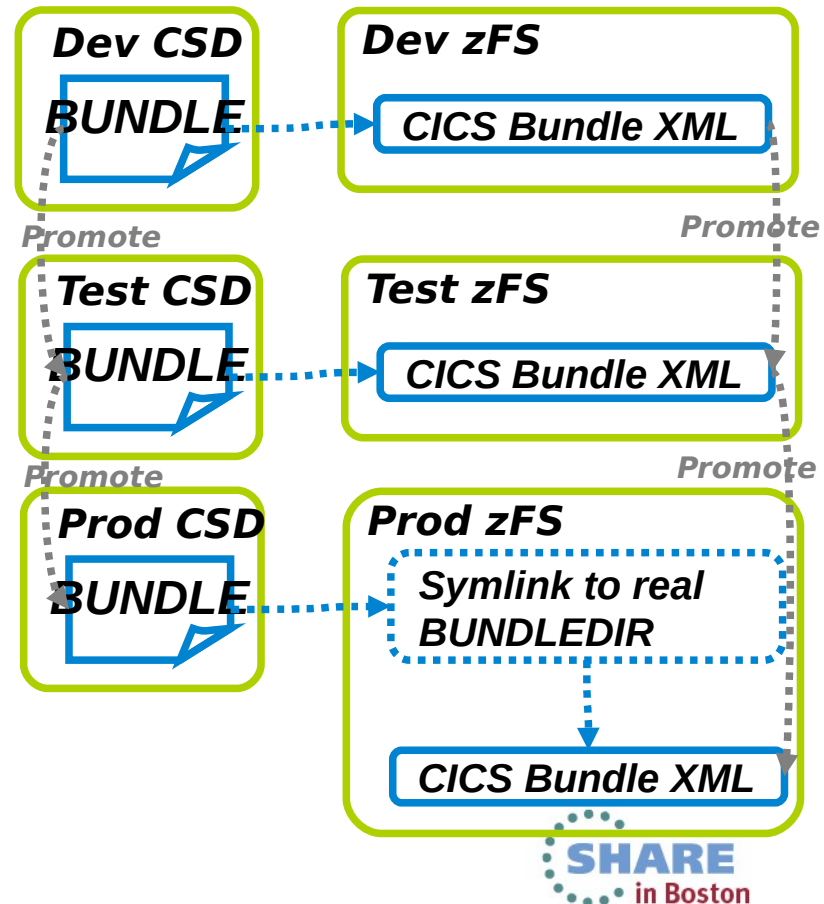


Migrating CICS Bundles XML from Dev to Test to Production



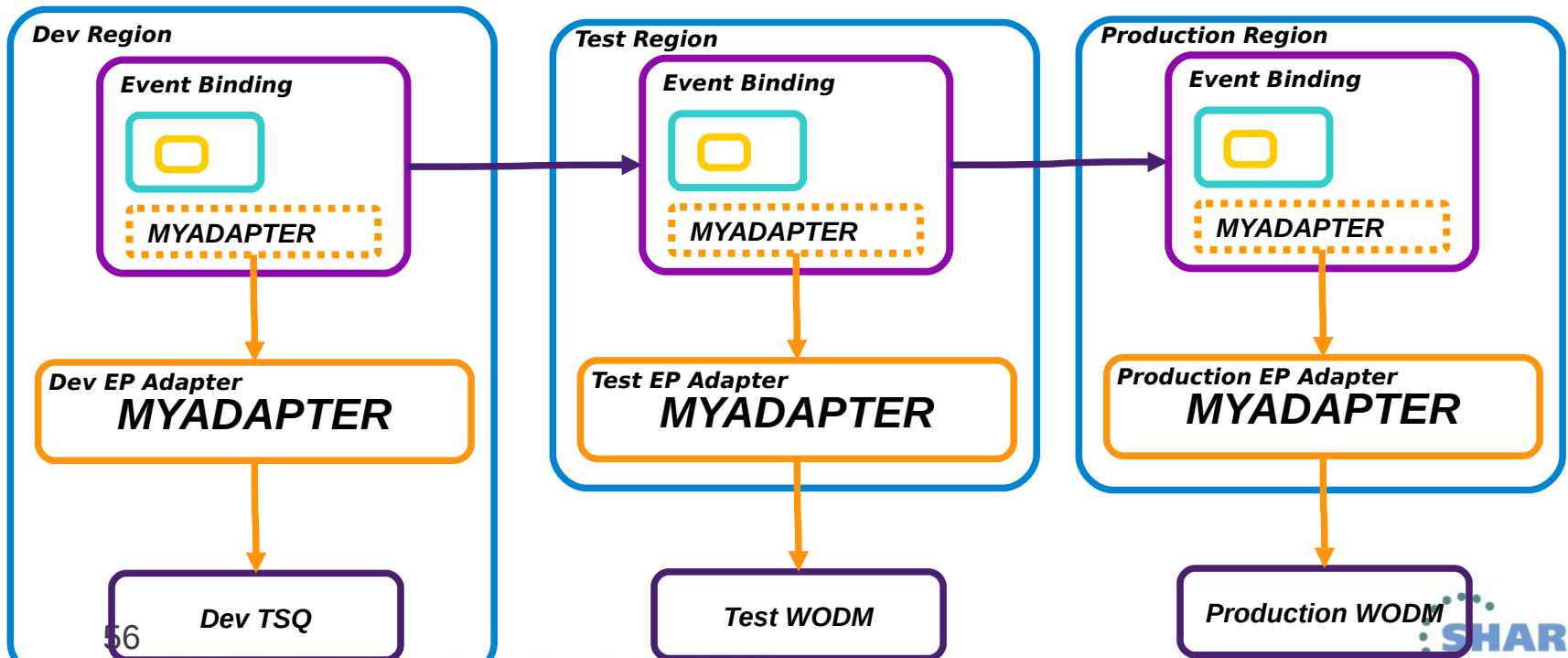
Top Tip 1: Avoid having to change BUNDLEDIR

- Changing the BUNDLE resource's BUNDLEDIR is undesirable because you aren't promoting the same resource that you tested
- Option 1: Put your CICS Bundle XML in the same directories on each zFS
- Option 2: Use Symlinks to point to the real bundle location



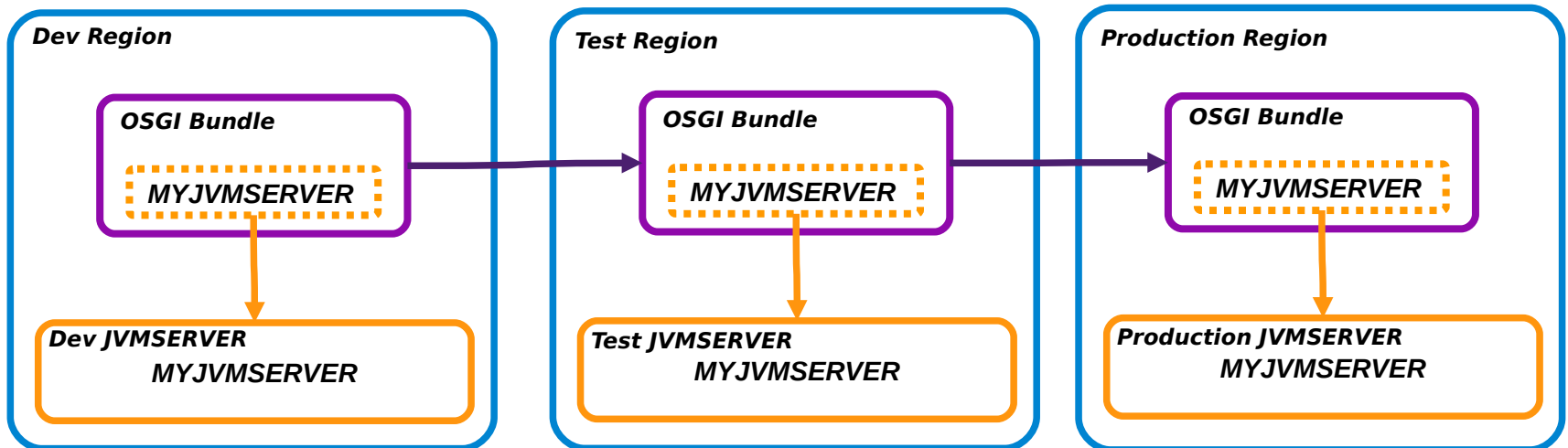
Top Tip 2: Use separate EPADAPTERs on all EVENTBINDINGS

- Use Separate EP Adapters in CICS TS V4.2 to ensure no changes are needed to CICS Bundles containing EVENTBINDINGS during migration



Top Tip 3: Use the same JVMSERVER names in all regions

- For CICS TS V4.2 have the same JVMSERVERs in all regions to ensure no changes are needed to CICS Bundles containing OSGI Bundles during migration



zFS setup - Best Practice

- 1. Create data set for usage as /var/cicsts zFS
- 2a. If using shared zFS across a sysplex
 - Mount data set at root (/cicsts) as a read/write file system
 - For each LPAR create symbolic links ...
 - link /var/cicsts to /cicsts (/var is always a symlink to /<LPAR>/var)
 - > ln -s /cicsts /var/cicsts
- 2b. If using non-shared zFS,
 - Mount data set onto /var as /var/cicsts

zFS setup - Best Practice

- 3. Set permissions of /var/cicsts to allow access by multiple readers (CICS regions) and a common writer (administrator)
 - 1. Set the owner (admin) to have read/write/execute
 - **this will be the userid required by zFS to export files into zFS**
 - 2. Set the readers to have read/execute access
 - > chgrp -R <group> /cicsts
 - > chmod -R 750 /cicsts
- 4. Set default file permission for the FTP daemon
 - give owners read/write permission
 - Give groups read permission
 - i.e Set UMASK for FTP to 027

zFS Cloud setup - Best Practice

- 5. If write access is required by multiple groups of writers (administrators) then you can either
 - 5a.
 - Set the group ownership to a common group in which all the writers are members
 - Set the FTP UMASK to 207 to give write permission to the group.
 - Set all CICS regions to run under the same uid which is limited to read access
 - Or 5b.
 - Use ACLs to add additional group permissions.
 - This can be achieved by activating the FSSEC resource class and using the setfacl command