IMPS Guide for DAL Installation

S–0049–52

RECORD OF REVISION

S–0049–52 Published March 2014 Supports Direct-Attached Lustre™ on Cray Linux Environment 5.2.UP00 release running on Cray XC30, XE, and XK systems.

S–0049–5101 Published December 2013 Supports Direct-Attached Lustre on Cray Linux Environment 5.1.UP01 release running on Cray XC30 systems.

Contents

Page

Introduction [1]

This technical note is intended as an introduction to the concepts and commands of the Image Management and Provisioning System (IMPS). IMPS is a new set of features that changes how software is installed, managed, provisioned, booted, and configured. In future releases, IMPS will support the installation of other Cray products, but for this release, only the installation and upgrade of DAL are supported by IMPS.

The purpose of IMPS is to create new methods of system management for Cray systems, simplify the install process, broaden the scope of the installation to encompass all the different Cray systems, and provide a base from which Cray can easily adopt new hardware types and configurations. It brings together industry standard methods used for image management and Cray proprietary technologies to produce a coherent image management and provisioning strategy.

1.1 IMPS Command-line Interface (CLI)

`impscli` is the command-line interface for using IMPS and represents a subset of the functionality of the IMPS API. For further information on IMPS CLI, see Chapter 2, IMPS CLI on page 9.

1.2 Repositories

Repositories are collections of RPMs of common type and architecture. Both SLES™ and CentOS™ repositories can exist, but content should never be mixed. By default, newly created repositories have content stored in the directory `/var/opt/cray/imps/repos`; this path, however, is modifiable within the configuration file. IMPS maintains metadata to track repositories, therefore, they can exist anywhere on the SMW.

1.3 Packages

Packages are synonymous with RPM packages. Each RPM is built for a specific architecture and operating system type and is found within a repository.

1.4 Package Collections

Package collections are logically associated packages, sometimes called capabilities, and are extensions of concepts employed by a large number of package managers. Packages within a package collection are typically listed by name and not specifically called out by version. It is possible for a package collection to reference other package collections. For example, a high level `lustre` package collection may reference other package collections such as `lustre_admin`, `lustre_config`, and `infiniband_stack`.

Functionally, package collections serve as organizational containers of packages and by default are located in the `/etc/opt/cray/imps/package_collections.d` directory within JSON (JavaScript Object Notation) files. Cray provides a set of package collections with each release by providing a new JSON file in this directory. Editing these Cray distributed files is not supported. Local changes to package collections must occur in the file `package_collections.local.json`.

1.5 Image Recipes

Image recipes represent the logical marriage of a group of packages and repositories responsible for building an image root. IMPS leverages the package manager, `zypper`, to build image recipes into images roots. Image recipes are, by default, stored in JSON files on the SMW within the `/etc/opt/cray/imps/image_recipes.d` directory. Cray provided image recipes cannot be changed with `impscli`; only those found in the local edits collection file, `image_recipes.local.json`, can be changed.

1.6 Image Roots

An image root, often referred to as an image, is created by building an image recipe. By default, IMPS images roots are located in the `/var/opt/cray/imps/images` directory. Users can `chroot` into the image root to examine its content and the packages that were automatically pulled in to resolve build dependencies. IMPS maintains metadata about an image root such as when it was created, when it was last provisioned, and when it was last updated.

Image roots are primarily obtained from building an image recipe, but can also be obtained from cloning existing image roots or capturing a running system node. An image root must be provisioned, as a separate step, in order to create a bootable image. The provisioning prepares the image root contents in the proper format for deployment.

1.7 Config Sets

IMPS *config sets* are site-specific settings consumed by services throughout a Cray system, typically in the form of JSON files. By default, config sets are located on the SMW in `/etc/opt/cray/share/`*config_set_name*. Currently there is a general association of one config set to one Cray system partition (e.g., `p0`, `p1`, `p2`).

The IMPS Configurator interactively guides the administrator through the process of providing needed configuration values in order to create a new config set. The administrator is provided a set of descriptions, reasonable default values and guidance on how to answer each question.

The IMPS Configurator can also be utilized to update a config set at any time. Administrators may revise their answers or press return to retain their previous settings.

Config sets are mounted read-only on the boot node and shared from there. All other service nodes in the system mount the config set shared from the boot node. The result is that the configuration information contained in the config set resides on the SMW and is available to all service nodes on the system (read-only). IMPS enabled services consume and act upon this configuration information.

IMPS CLI [2]

The Image Management and Provisioning System (IMPS) command-line interface (CLI), `impscli`, operates in three contexts:

- As a command line tool
- As an interactive session
- Via content piped into it

The most common use case is to invoke simple, single commands, as follows:

`impscli` *action* [*object* [*arg*, *arg*, ..., *arg*]]

IMPS CLI includes a built-in help feature:

```
smw:~ # impscli
Welcome to the Cray Image Management and Provisioning System (IMPS)
Type 'help' for more information
imps> help

Documented commands (type help <topic>)
=======================================
build      configureall  exit    import        remove  strip  validate
clone      create        export  list          set     sync   validateall
configure  diff          extend  provisiondal  show    update

imps> help build

        build image_recipe <name>
        build image_recipes <name><name>...<name>
```

2.1 Example Actions for Repository

The following sections illustrate the command line syntax for common actions.

Repositories can be listed, shown, created, set, removed, and updated. They can be created by hand and are invoked by the CLE installer in order to provide content for creating images.

Example 1. Obtaining a list of defined repositories

```
smw:~ # impscli list repositories
INFO - Repositories:
 centos_6.4_x86-64
 centos_6.4_x86-64_updates
 cle_5.2up00_centos_6.4_x86-64_ari
 cle_5.2up00_centos_6.4_x86-64_ari_updates
 whamcloud-lustre-supplemental_2.4_centos_6.4_x86-64
 whamcloud-lustre-supplemental_2.4_centos_6.4_x86-64_updates
```

Example 2. Showing an individual repository

```
smw:~ # impscli show repository cle_5.2up00_centos_6.4_x86-64_ari
INFO - Repository: cle_5.2up00_centos_6.4_x86-64_ari
 arch: x86_64
 enabled: True
 path: /var/opt/cray/imps/repos/cle_5.2up00_centos_6.4_x86-64_ari
 type: RHEL6
```

Example 3. Creating an individual repository

The following shows the creation of a SLES11 x86-64 based repository without any content.

```
smw:~ # impscli create repository wiki_test with type SLES11 arch x86-64
INFO - Repository 'wiki_test' successfully created.
```

Example 4. Syncing content with a quoted glob path

```
smw:~ # impscli sync repository wiki_test glob_path \
"/home/crayadm/release/0926_cle_0923/install/centos/cray-packages/*.rpm"
INFO - Sync operation on repository 'wiki_test' will add 52 files.
INFO - Generating deep identity for repository 'wiki_test'.
INFO - Deep identity information for repository 'wiki_test' cached.
INFO - Repository 'wiki_test' sync complete.
```

Example 5. Changing a repository attribute

```
smw:~ # impscli set repository test-repo type RHEL6
INFO - Successfully updated repository 'test-repo' key(s) 'type'.
```

Example 6. Removing repositories

```
smw:~ # impscli remove repository wiki_test remove_local true
INFO - Successfully removed local content for repository 'wiki_test'
out of '/var/opt/cray/imps/repos/wiki_test'.
INFO - Successfully removed repository 'wiki_test'.
```

Example 7. Updating repository content

```
smw:~ # impscli update repository wiki_test glob_path '/tmp/bogus_source/*.rpm'
INFO - Successfully copied 3 packages into repository 'wiki_test'.
```

2.2 Example Actions for Package Collections

Package collections are objects defined within JSON files, most typically within files stored in the `/etc/opt/cray/imps/package_collections.d` directory, and are used to group a number of like packages. IMPS CLI can help manage these objects by listing, showing, cloning, creating, removing, extending, and stripping these.

Example 8. Listing available package collections

```
smw:~ # impscli list package_collections
INFO - Package Collections:
 centos64_admin
 centos64_base
 centos64_debug
 centos64_service
 centos64_storage
 centos64_storage_generic
 centos64_storage_scsi
 cle52_centos64_base
 cle52_centos64_kernel_ari
 cle52_centos64_lustre
 cle52_centos64_lustre_kernel_ari
 cle52_centos64_service
```

Example 9. Showing individual package collections

```
smw:~ # impscli show package_collection cle52_centos64_lustre
INFO - Package Collection 'cle52_centos64_lustre':
 description: Collection of packages needed to support Direct
  Attached Lustre (DAL) server nodes.
 package_collections: centos64_storage, cle52_centos64_service
 packages: cray-lustre-utils, cray-multipath, cray-sysadm,
  e2fsprogs-1.42.7.wc1-7.el6.x86_64, libcom_err-1.42.7.wc1-7.el6,
  mysql-connector-odbc
```

Example 10. Cloning one package collection to another

Use the `clone` action to create a modifiable duplicate version of an existing package collection. The modifiable version can then be edited for testing.

```
smw:~ # impscli clone package_collection centos64_debug to wiki_test
INFO - Package collection 'centos64_debug' successfully cloned to 'wiki_test'.
```

Example 11. Creating a new package collection

```
smw:~ # impscli create package_collection wiki_test2
INFO - Successfully created package collection 'wiki_test2'.
```

User-modifiable package collections can be extended with packages or package collections.

Example 12. Extending a user-modifiable package collection

```
smw:~ # impscli extend package_collection wiki_test2 with package wiki_package
INFO - Package collection 'wiki_test2' successfully extended.
smw:~ # impscli extend package_collection wiki_test2 with package_collection \
wiki_package_collection2
INFO - Package collection 'wiki_test2' successfully extended.
smw:~ # impscli extend package_collection wiki_test2 with package wiki_package2 \
wiki_package3
INFO - Package collection 'wiki_test2' successfully extended.
```

Likewise, modifiable package collections can be stripped of packages and package collections.

Example 13. Stripping packages/package collections from a modifiable package collection

Use the `strip` action to remove a package or package collection from a modifiable package collection, typically to test the modified package collection.

```
smw:~ # impscli strip package_collection wiki_test2 of package wiki_package2 wiki_package3
INFO - Strip operation on 'wiki_test2' successful.
smw:~ # impscli strip package_collection wiki_test2 of package_collection \
wiki_package_collection2
INFO - Strip operation on 'wiki_test2' successful.
```

The modifiable/locally-defined package collections can be removed.

Example 14. Removing a modifiable package collection

```
smw:~ # impscli remove package_collection wiki_test wiki_test2
INFO - Package collection 'wiki_test' successfully removed.
INFO - Package collection 'wiki_test2' successfully removed.
```

2.3 Example Actions for Image Recipes

IMPS CLI can manipulate image recipes without the need to hand edit JSON files. IMPS CLI implements list, show, create, clone, extend, strip, build and remove operations.

Example 15. Listing available image recipes

```
smw:~ # impscli list image_recipe
INFO - Image Recipes:
 cle52_centos64_dal
 wayne_dal_test
```

Example 16. Viewing an image recipe definition

```
smw:~ # impscli show image_recipe cle52_centos64_dal
INFO - Image Recipe 'cle52_centos64_dal'
 Package_collections: cle52_centos64_kernel_ari,
   cle52_centos64_service, cle52_centos64_lustre_kernel_ari,
   cle52_centos64_lustre
 Repositories: lustre, localbuilds64, xc64, centos64_packages
```

Example 17. Cloning an image recipe

It is often helpful to create a clone of an existing image recipe for individual test case work. Cray-provided image recipes cannot be modified. Use the `clone` action to create a user-modifiable duplicate image recipe.

```
smw:~ # impscli clone image_recipe cle52_centos64_dal to wiki_test_recipe
INFO - Successfully cloned image recipe 'cle52_centos64_dal' to image
recipe 'wiki_test_recipe'.
```

Image recipes can be extended by content that does not yet exist. In this example, `test_repo1`, `test_pkg1`, `test_pkg2`, and `test_pcs` have not been defined. These objects **must** be created before an image is built.

Example 18. Extending a modifiable image recipe

```
smw:~ # impscli clone image_recipe cle52_centos64_dal to wiki_test_recipe
INFO - Successfully cloned image recipe 'cle52_centos64_dal' to image
recipe 'wiki_test_recipe'.
smw:~ # impscli extend image_recipe wiki_test_recipe with packages test_pkg1 \
test_pkg2 test_pkg3
INFO - Successfully extended image recipe 'wiki_test_recipe'.
smw:~ # impscli extend image_recipe wiki_test_recipe with package_collections \
test_pc1 test_pc2
INFO - Successfully extended image recipe 'wiki_test_recipe'.
smw:~ # impscli extend image_recipe wiki_test_recipe with repository test_repo1
INFO - Successfully extended image recipe 'wiki_test_recipe'.
```

Stripping image recipes of repositories, packages, and package collections will succeed as long as at least one object is removed (content unable to be removed is reported).

Example 19. Stripping a modifiable image recipe

Use the `strip` action to remove content from a modifiable image recipe, typically to test the modified image recipe.

```
smw:~ # impscli strip image_recipe wiki_test_recipe of package test_pkg2
INFO - Operation to strip image recipe 'wiki_test_recipe' successful.
smw:~ # impscli strip image_recipe wiki_test_recipe of package_collections \
test_pc2 test_pc1
INFO - Operation to strip image recipe 'wiki_test_recipe' successful.
smw:~ # impscli strip image_recipe wiki_test_recipe of repository test_repo1
INFO - Operation to strip image recipe 'wiki_test_recipe' successful.
```

Example 20. Building an image root using image recipes

```
smw:~ # impscli build image_recipe cle52_centos64_dal
INFO - Repository 'lustre' validates.
INFO - Repository 'localbuilds64' validates.
INFO - Repository 'xc64' validates.
INFO - Repository 'centos64_packages' validates.
INFO - Package collection 'cle52_centos64_kernel_ari' validates.
INFO - Package collection 'cle52_centos64_service' validates.
INFO - Package collection 'cle52_centos64_lustre_kernel_ari' validates.
INFO - Package collection 'cle52_centos64_lustre' validates.
INFO - Image recipe 'cle52_centos64_dal' validates.

The following NEW packages are going to be installed:
[...]
312 new packages to install.
Overall download size: 288.8 MiB. After the operation,
additional 1009.2 MiB will be used.
Continue? [y/n/?] (y): y
INFO - Successfully updated image record for 'cle52_centos64_dal'.
INFO - Image recipe 'cle52_centos64_dal' has successfully built image
'cle52_centos64_dal'.
INFO - Rebuilding the RPM database for image 'cle52_centos64_dal'.
INFO - Image 'cle52_centos64_dal' RPM database successfully rebuilt.
```

Example 21. Removing an unnecessary image recipe

```
smw:~ # impscli remove image_recipe wiki_test_recipe
INFO - Successfully removed image recipe 'wiki_test_recipe'.
```

2.4 Example Actions for Image Roots

> **Note:** Image roots are often referred to as images, including within `impscli`. It is important to remember that regardless of the term used, an image root must be provisioned before it is in the proper format for deployment, as described in Image Roots on page 6.

Image roots are the outcome of image recipe build operations, existing image clones or live node captures. IMPS CLI is used to manage these images, and implements list, show, clone, remove, and a number of provisioning methods.

Example 22. Listing images

```
smw:~ # impscli list images
INFO - Images:
 cle52_centos64_dal
 cle52_centos64_dal_ofed
 dev_centos64_dal
 haasken_dal_test
 imps-jasonm
 wayne_dal_test
```

Example 23. Cloning an image

An image root can be cloned to create a duplicate; however, the content of an image root can only be modified by changing the image recipe and recreating the image root.

```
smw:~ # impscli clone image imps-jasonm to imps-jsl
INFO - Copying image root of image 'imps-jasonm'...
INFO - Successfully cloned image 'imps-jasonm' to 'imps-jsl'.
```

Example 24. Removing an image

```
smw:~ # impscli remove image imps-jsl
INFO - Successfully removed image 'imps-jsl'.
```

Example 25. Provisioning a DAL image for booting

```
smw:~ # impscli provisiondal image cle52_centos64_dal to \
/opt/xt-images/opal-p3-trunk.201307160231.SP2-DEVSP2-P3-DAL
INFO - Removing existing instance at
'/opt/xt-images/opal-p3-trunk.201307160231.SP2-DEVSP2-P3-DAL/cle52_centos64_dal'.
INFO - Copying 'cle52_centos64_dal' to
'/opt/xt-images/opal-p3-trunk.201307160231.SP2-DEVSP2-P3-DAL'.
INFO - Using existing service node boot param file
'/opt/xt-images/opal-p3-trunk.201307160231.SP2-DEVSP2-P3-DAL/
service/boot/parameters-snl' with image 'cle52_centos64_dal'.
INFO - Calling xtpackage on
'/opt/xt-images/opal-p3-trunk.201307160231.SP2-DEVSP2-P3-DAL/cle52_centos64_dal'.
292+1 records in
292+1 records out
3924184 bytes (3.9 MB) copied, 0.288043 s, 13.6 MB/s
copying image
copying modules
running depmod
creating load file:
/opt/xt-images/opal-p3-trunk.201307160231.SP2-DEVSP2-P3-DAL/
cle52_centos64_dal/cle52_centos64_dal.load
compressing initramfs
INFO - Provisioning of DAL image 'cle52_centos64_dal' successful.
```

Example 26. Determining the Package or File-level differences of an image

```
smw:~ # impscli diff image imps-jsl dal_cle52_centos_6.4
INFO - Hashed Difference:
 -var/lib/rpm/__db.001
 -var/lib/rpm/__db.002
 -var/lib/rpm/__db.004
```

2.5 Example Actions for Config Sets

IMPS CLI offers a number of routines that can be utilized to list, show, create, clone, configure, configure all, update and remove config sets. In this release, configuration sets share the same name as the partition being configured.

Example 27. Listing available config sets

```
smw:~ # impscli list config_sets
INFO - Configuration Sets:
 bhergert_test
 imps-jasonm
 imps-jasonm2
 lustre
 p3
 p3-install-testing
 p3.prototype
```

Example 28. Showing details of a given config set

```
smw:~ # impscli show config_set p3
INFO - Configuration set p3:
 images: cle52_centos64_dal
 initialized: 06.07.13 09:56:56 AM
 path: /etc/opt/cray/share/p3
```

Example 29. Creating a new config set

In this example, `images` is a collection of images that were originally used to create the config set. The `create config_set` action launches the IMPS Configurator, which interactively guides the user through the process of providing the configuration details that are included in the config set.

```
smw:~ # impscli create config_set p3 with images cle52_centos64_dal
[...]
INFO - Created configuration set 'p3'.
```

Example 30. Cloning a config set

A config set can be cloned to create a duplicate; however, the content of a config set is fixed and cannot be modified, only reconfigured as in Example 31. A clone of a config set provides a backup in the event that the reconfiguration does not produce the expected results.

```
smw:~ # impscli clone config_set p3 to p3-backup
INFO - Successfully cloned config_set 'p3' to 'p3-backup'.
```

Example 31. Reconfigure a config set

Modifying a configuration value is done through the `update config_set` action. The IMPS Configurator is launched, and again prompts the user for the configuration data, providing the user the opportunity to modify, or reconfigure, the values. This action honors previously set values.

```
smw:~ # impscli update config_set p3
INFO - Reconfiguring existing services: 'classes, system'.
INFO - Existing settings for service 'classes' read in.
[..Configurator prompts..]
INFO - Configuration set 'p3' config for service 'system'
complete.
INFO - Successfully updated config set 'p3'.
```

Example 32. Populating a config set with templates and distribution content from an additional image

Only new content is added; if new templates are found, the Configurator will process them and prompt the user for configuration data.

```
smw:~ # impscli update config_set wiki_test with image imps-jasonm
INFO - Successfully updated config set 'wiki_test'.
smw:~ # impscli show config_set wiki_test
INFO - Configuration set wiki_test:
 images: [u'cle52_centos64_dal', u'imps-jasonm']
 initialized: 07.22.13 04:45:06 PM
 path: /etc/opt/cray/share/wiki_test
```

Example 33. Updating individual config set services

```
smw:~ # impscli configure config_set wiki_test classes
INFO - Existing settings for service 'classes' read in.
[...]
INFO - Configuration set 'wiki_test' config for service
'classes' complete.
```

Example 34. Removing a config set

```
smw:~ # impscli remove config_set wiki_test
INFO - Successfully removed config set 'wiki_test'.
```

Use Cases [3]

The following tasks include combinations of IMPS CLI actions that administrators or developers may need to know. These steps are derived from the above information into small, easy to digest commands with a brief narrative about how and why these steps are needed, and what they do.

3.1 Testing a Single RPM Installation

As a developer, testing the functionality of a packaged RPM with no other dependencies is straight forward. This works well for testing new versions of libraries and software that do not require any startup configuration action.

Procedure 1. Testing a single RPM installation

1. Distribute the content of the RPM to a running IMPS node of interest.
   a. To the boot node.
   b. To the node of interest.
2. `ssh` to the node.
3. Install or update the RPM.
4. Test new RPM functionality.
5. Reboot the node to restore node to original state.

Example 35. Testing a single RPM install

1. Distribute the content of the RPM to a running IMPS node of interest.

```
smw:/home/crayadm/user # scp \
cray-config-service-1.0-1.0000.44547.0.0.ari.x86_64.rpm root@boot:/tmp/.

smw:/home/crayadm/user # ssh root@boot
boot:~ # scp \
/tmp/cray-config-service-1.0-1.0000.44547.0.0.ari.x86_64.rpm root@nid00013:/tmp/.
```

2. `ssh` to the node.

```
boot:~ # ssh root@nid00013
```

3. Install or update the RPM.

```
nid00013:~ # rpm -ivf /tmp/cray-config-service-1.0-1.0000.44547.0.0.ari.x86_64.rpm
```

Note: Steps 4 and 5 are omitted as they are site specific.

It is important to note that these changes do not persist between reboots. As soon as the node is bounced or rebooted, the modified RAM root will be recreated from its IMPS image.

3.2 Testing Advanced Image Root Changes

It may be necessary to create a custom image. This can be the case when testing modified packages or packages that need to be added to an existing image.

Procedure 2. Testing advanced image root changes

1. Identify the image recipe of interest.
2. Clone the image recipe.
3. Create custom repository for testing changes.
4. Populate custom repository.
5. Modify the cloned image recipe.
6. Build and provision a new image.
7. Boot as many nodes as necessary to test the changes.
8. Clean up image recipes, package collections, repositories and images that are no longer needed.

Example 36. Testing advanced image root changes

1. Identify the image recipe of interest.

```
smw:~ # impscli list image_recipes
 INFO - Image Recipes:
dal_cle_5.2up00_centos_6.4_x86-64_ari
dal_cle_trunk_centos_6.4_x86-64_ari
```

2. Clone the image recipe.

```
smw:~ # impscli clone image_recipe dal_cle_trunk_centos_6.4_x86-64_ari \
to jsl_dal_wiki_test
 INFO - Successfully cloned image recipe
'dal_cle_trunk_centos_6.4_x86-64_ari' to image recipe
'jsl_dal_wiki_test'.
```

3. Create custom repository for testing changes.

```
smw:~ # impscli create repository cle_jsl_centos_6.4_x86-64_ari \
with type RHEL6 arch x86-64
 INFO - Generating deep identity for repository
 'cle_jsl_centos_6.4_x86-64_ari'.
 INFO - Deep identity information for repository
 'cle_jsl_centos_6.4_x86-64_ari' cached.
 INFO - Repository 'cle_jsl_centos_6.4_x86-64_ari' successfully created.
```

In order to determine repository type and architecture, it is sometimes useful to show the type and architecture for the like repository that is to be temporarily replaced. This can be done using:

```
smw:~ # impscli show repository cle_trunk_centos_6.4_x86-64_ari
 INFO - Repository: cle_trunk_centos_6.4_x86-64_ari
  arch: x86-64
  enabled: True
  path: /var/opt/cray/imps/repos/cle_trunk_centos_6.4_x86-64_ari
  type: RHEL6
```

4. Populate custom repository.

   Whenever adding or removing content from a repository, it is important to regenerate the repository identity so that IMPS can validate it. The default configuration location for IMPS repository stores is `/var/opt/cray/imps/repos/`*repo_name*.

   To add or remove content from a repository, copy content from existing IMPS repositories or from the OBS build service into the associated repository folder (`/var/opt/cray/imps/repos/cle_jsl_centos_6.4_x86-64_ari` for this example).

   Moving content to these repositories alone is not enough; when the repository content is complete, tell IMPS that the content in the repository is valid by updating it:

```
smw:~ # impscli update repository cle_jsl_centos_6.4_x86-64_ari
 INFO - Generating deep identity for repository
 'cle_jsl_centos_6.4_x86-64_ari'.
 INFO - Deep identity information for repository
 'cle_jsl_centos_6.4_x86-64_ari' cached.
```

5. Modify the cloned image recipe, as desired.

   It is then necessary to direct the image recipe to the newly created repository:

```
smw:~ # impscli extend image_recipe jsl_dal_wiki_test with repository \
cle_jsl_centos_6.4_x86-64_ari
 INFO - Successfully extended image recipe 'jsl_dal_wiki_test'.
```

The image recipe also points at the old associated repositories and must be redirected by stripping them out:

```
smw:~ # impscli strip image_recipe jsl_dal_wiki_test of repository \
cle_trunk_centos_6.4_x86-64_ari
 INFO - Operation to strip image recipe 'jsl_dal_wiki_test'
 successful.
 smw:~ # impscli strip image_recipe jsl_dal_wiki_test of repository \
cle_trunk_centos_6.4_x86-64_ari_updates
 INFO - Operation to strip image recipe 'jsl_dal_wiki_test'
 successful.
```

To install a specific package into an image from a repository:

```
smw:~ # impscli extend image_recipe jsl_dal_wiki_test with \
package jsl_test_package
 INFO - Successfully extended image recipe 'jsl_dal_wiki_test'.
```

6. Build and provision the new image.

```
smw:~ # impscli build image_recipe jsl_dal_wiki_test
 INFO - Repository 'centos_6.4_x86-64' validates.
 INFO - Repository 'whamcloud-lustre-supplemental_2.5_centos_6.4_x86-64' validates.
 INFO - Repository 'whamcloud-lustre-supplemental_2.5_centos_6.4_x86-64_updates' validates.
 INFO - Repository 'cle_jsl_centos_6.4_x86-64_ari' validates.
 INFO - Repository 'centos_6.4_x86-64_updates' validates.
 INFO - Package collection 'cle_trunk_centos_6.4_lustre_kernel_ari' validates.
 INFO - Package collection 'cle_trunk_centos_6.4_kernel_ari' validates.
 INFO - Package collection 'cle_trunk_centos_6.4_lustre' validates.
 INFO - Package collection 'cle_trunk_centos_6.4_service' validates.
 INFO - Image recipe 'jsl_dal_wiki_test' validates.
 INFO - Calling Package manager to build new image root; this will take a few minutes.

 The following NEW packages are going to be installed:
 [...]
 313 new packages to install.
 Overall download size: 291.0 MiB. After the operation, additional 1016.6 MiB will be used.
 Continue? [y/n/?] (y): y
 INFO - Successfully updated image record for 'jsl_dal_wiki_test'.
 INFO - Image recipe 'jsl_dal_wiki_test' has successfully built image 'jsl_dal_wiki_test'.
 INFO - Rebuilding the RPM database for image 'jsl_dal_wiki_test'.
 INFO - Image 'jsl_dal_wiki_test' RPM database successfully rebuilt.
```

Provision the newly created IMPS image with other existing images.

```
smw:~ # impscli provisiondal image jsl_dal_wiki_test to \
/opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2
 INFO - Copying 'jsl_dal_wiki_test' to
'/opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2'.
 INFO - Using existing service node boot param file
'/opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/service/boot/parameters-snl'
with image 'jsl_dal_wiki_test'.
INFO - Calling xtpackage on
'/opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/jsl_dal_wiki_test'.
 293+1 records in
 293+1 records out
 3926840 bytes (3.9 MB) copied, 0.307402 s, 12.8 MB/s
 copying image
 creating load file: /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/
jsl_dal_wiki_test/jsl_dal_wiki_test.load
 compressing initramfs
 INFO - Provisioning of DAL image 'jsl_dal_wiki_test' successful.
```

Build the CPIO file containing all of your images.

```
opal-smw:~ # xtbootimg -L \
/opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/service/SNL0.load -L
/opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/compute/CNL0.load -L
/opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/\
jsl_dal_wiki_test/jsl_dal_wiki_test.load -c jsl_dal_wiki_test.cpio
 processing /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/service/SNL0.load
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/service/SNL0/size-initramfs
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/service/SNL0/
bzImage-3.0.80-0.5.2_1.0000.7629-cray_ari_s.bin
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/service/SNL0/parameters
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/service/SNL0/initramfs.gz
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/service/SNL0/debug/...
 processing /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/compute/CNL0.load
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/compute/CNL0/size-initramfs
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/compute/CNL0/
bzImage-3.0.80-0.5.2_1.0000.7629-cray_ari_c.bin
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/compute/CNL0/parameters
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/compute/CNL0/initramfs.gz
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/compute/CNL0/debug/...
 processing /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/jsl_dal_wiki_test/
jsl_dal_wiki_test.load
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/jsl_dal_wiki_test/
jsl_dal_wiki_test/size-initramfs
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/jsl_dal_wiki_test/
jsl_dal_wiki_test/bzImage-2.6.32-358.el6_1.0000.7630-cray_ari_s_cos.bin
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/jsl_dal_wiki_test/
jsl_dal_wiki_test/parameters
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/jsl_dal_wiki_test/
jsl_dal_wiki_test/initramfs.gz
 adding /opt/xt-images/opal-p2-trunk.201310290401.SP2-DEVSP2-P2/jsl_dal_wiki_test/
jsl_dal_wiki_test/debug/...
 2942375 blocks
 /root
 "jsl_dal_wiki_test.cpio" successfully created.
```

Note: Steps 7 and 8 are omitted as they are site specific.