XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide

(CLE 6.0.UP07)

S-2583

Contents

1 About XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide

ATTENTION: DataWarp API Access from non-Cray Environments will only be available via a patch for CLE 6.0.UP07 and later releases. To request access to this patch contact Cray Service.

Scope and Audience

The *XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide* (S-2583) provides detailed procedures to install and configure DataWarp RESTful API access tools on a commercial off-the-shelf (COTS), non-Cray node. This allows DataWarp clients on the COTS node, such as `dwstat` and `dwcli`, to access DataWarp on an XC Series system. By default, the DataWarp RESTful API and API clients are only accessible on Cray platforms.

This publication applies to an initial configuration, an update following a DataWarp patch or configuration change, and an upgrade following a CLE update.

This publication is intended for system installers, administrators, and anyone who installs and configures software on a Cray XC™ Series system. It assumes some familiarity with standard Linux and open source tools (e.g., zypper/yum for RPMs).

Release Information

XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide (CLE 6.0.UP07) S-2583 supports Cray software release CLE 6.0.UP07 for Cray XC™ Series systems.

Table 1. Record of Revision

Publication Title	Release Date	Updates
XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide (CLE 6.0.UP07) S-2583	12 July 2018	
XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide (CLE 6.0.UP06 Rev A) S-2583	30 Apr 2018	Bug fixes
XC™ Series DataWarp™ API Access from non-Cray Environments Installation Guide (CLE 6.0.UP06) S-2583	15 Mar 2018	Initial release

2 Configure DataWarp RESTful API Access from non-Cray Environments

ATTENTION: DataWarp API Access from non-Cray Environments will only be available via a patch for CLE 6.0.UP07 and later releases. To request access to this patch contact Cray Service.

Some workload managers (WLMs) support running subsets of their product on commercial off-the-shelf (COTS), non-Cray platforms that do not have direct access to the Cray HSN. This is useful for example to dedicate a whole node to a resource intensive WLM process. Depending on how the WLM has implemented DataWarp support, the entity running on the COTS box may need to interact with the DataWarp RESTful API. By default, the DataWarp RESTful API and API clients are only accessible on Cray platforms.

The procedures for configuring a COTS node so that DataWarp API clients running on it can both execute and access the DataWarp RESTful API involve copying over system configuration data and security credentials, starting up a MUNGE daemon, and installing DataWarp software in the COTS node. These steps are automated through the use of RPM. The procedures are:

- Initial configuration - *Configure a non-Cray Node for DataWarp API Access* on page 4
- Update following a patch or DataWarp configuration change - *Update a non-Cray Node for DataWarp API Access Following a DataWarp Patch or Configuration Change* on page 9
- Upgrade following a CLE update or upgrade - *Upgrade a non-Cray Node for DataWarp API Access Following a CLE Update or Upgrade* on page 13

2.1 Configure a non-Cray Node for DataWarp API Access

Prerequisites

- The XC Series system (XC) must have LiveUpdates enabled.
- External API gateway nodes were configured during DataWarp installation; if not, see *XC™ Series DataWarp™ Installation and Administration Guide* (S-2564) for the initial DataWarp installation procedure.
- The commercial off-the-shelf (COTS), non-Cray node must run a SLES release and hardware architecture equivalent to the CLE environment internal login or MOM nodes.
- The COTS node must have a network route to the XC Series SMW (SMW) and be able to access port 2526.
- Required by MUNGE:
  - COTS node and XC must have synchronized time
  - COTS node and XC must have an identical UID and GID namespace

About this task

This procedure is for the initial configuration of a COTS node for DataWarp API access. See *Update a non-Cray Node for DataWarp API Access Following a DataWarp Patch or Configuration Change* on page 9 to perform an update following a DataWarp patch or configuration change. See *Upgrade a non-Cray Node for DataWarp API Access Following a CLE Update or Upgrade* on page 13 to perform an upgrade of the COTS configuration following an update or upgrade of CLE on the XC.

This procedure enables native clients of the DataWarp RESTful API to run from non-XC environments. It comprises the following subtasks:

A. Configure the COTS node to use an SMW as a software repository source
B. Configure the COTS node to use a local directory as a software repository source
C. Build the cray-datawarp-cots-api-access RPM
D. Update the COTS node local RPM repository with the new `cray-datawarp-cots-api-access` RPM
E. Complete the initial configuration of a COTS node for DataWarp API access
F. Test DataWarp API access from the COTS node

These subtasks are highlighted within the procedure.

Procedure

A. Configure a COTS node to use an XC Series SMW as a software repository source

This part of the procedure enables a COTS node to install RPMs hosted and mirrored by an SMW. The SMW hosting list includes SLES ISO RPM and SLES RPM update repositories, therefore, it is not necessary to maintain a separate duplicate source for these RPMs. The first step, which is optional but recommended, suggests ways to find and disable or remove these duplicate repositories.

> **TIP:** Although this procedure enables a COTS node to install any XC Series software to it via `zypper`, not all software is supported on COTS nodes. Only RPMs that come from SLES and RPMs associated with DataWarp are supported.

1. (Optional) Log on to a COTS node, then disable or remove software repositories that correspond to SLES ISO RPM material or update repositories. The proceeding steps re-establish these repositories using your SMW as a host.

a. Review repositories.

```
cots# zypper repos
Repository priorities are without effect. All enabled repositories share the
same priority.

# | Alias              | Name               | Enabled | GPG Check | Refresh
--+--------------------+--------------------+---------+-----------+--------
1 | SLES12-SP3-12.3-0 | SLES12-SP3-12.3-0 | Yes     | (r ) Yes  | No

...
```

b. Disable repositories.

```
cots# zypper modifyrepo --disable repo
Repository 'repo' has been successfully disabled.
```

c. Permanently remove repositories.

```
non-XC# zypper removerepo repo
```

2. Log on to an XC internal login node, then capture the current LiveUpdates repository configuration.

   Remember to define SYSTEM_NAME appropriately.

```
login# SYSTEM_NAME=system-name
login# CLE_RELEASE=$(grep RELEASE /etc/opt/cray/release/cle-release | cut -d = -f 2)
login# SLES_RELEASE=SLES$(lsb-release -rs)
login# SNAPSHOT=$(date +%Y%m%d%H%M%S | tr -cd '[0-9]')
login# XC_REPO_FILE=$SYSTEM_NAME-$CLE_RELEASE-$SLES_RELEASE-$SNAPSHOT.repo
login# zypper repos --export $XC_REPO_FILE
Repositories have been successfully exported to ./system-name-6.0.UP05-
SLES12.3-20170911094727.repo.
```

3. Modify the *baseurl* attribute in $XC_REPO_FILE to point to the single SMW hostname rather than multiple internal XC hostnames.

   Remember to define SMW_HOSTNAME appropriately.

```
login# SMW_HOSTNAME=smw-hostname
login# XC_INTERNAL_ADDRESS=$(grep -om 1 http.*: $XC_REPO_FILE | cut -d : -f 1-2)
login# sed -i -e "s@$XC_INTERNAL_ADDRESS@http://$SMW_HOSTNAME@" $XC_REPO_FILE
login# sed -i -e '/ http/d' $XC_REPO_FILE
```

4. Copy $XC_REPO_FILE to the /etc/zypp/repos.d/ directory on the COTS node.

   Remember to define COTS_NODE_HOSTNAME appropriately.

```
login# COTS_NODE_HOSTNAME=cots-hostname
login# scp $XC_REPO_FILE root@$COTS_NODE_HOSTNAME:/etc/zypp/repos.d/
```

Section B: Configure a COTS node to use a local directory as a software repository source

5. Log on to a COTS node, then create a directory to store hand-built RPMs that can be installed by zypper.

   IMPORTANT: Remember to set LOCAL_RPM_DIR appropriately. This must be a persistent location that survives reboots. This directory is referenced again in this procedure and should be noted for later use. Cray recommends locking down permissions on this directory as the cray-datawarp-cots-api-access RPM may contain sensitive data.

```
cots# LOCAL_RPM_DIR=local-rpm-dir
cots# mkdir -p $LOCAL_RPM_DIR
cots# chmod go-rwx $LOCAL_RPM_DIR
```

6. Add the new local repository.

```
cots# zypper addrepo -G -p 1 --check --refresh --name local_rpms $LOCAL_RPM_DIR local_rpms
Adding repository 'local_rpms' ...
...
Repository 'local_rpms' successfully added
```

Section C: Build the cray-datawarp-cots-api-access RPM

This procedure captures configuration data and dependency information located on the XC internal login node and places it in to an RPM. The RPM is later installed on to the COTS node as part of a subsequent procedure.

7. Log on to an XC internal login node, then install the source RPM.

```
login# zypper source-install cray-datawarp-cots-api-access
Building repository 'common_cle_6.0up06_sles_12sp3_x86-64_ari'
cache ..............[done]
Building repository 'common_cle_6.0up06_sles_12sp3_x86-64_ari_updates'
cache .......[done]
...
Reading installed packages...
Loading repository data...
Resolving package dependencies...

The following source package is going to be installed:
  cray-datawarp-cots-api-access

1 source package to install.
Overall download size: 7.2 KiB. Already cached: 0 B. After the operation,
additional 11.2 KiB will be used.
Continue? [y/n/...? shows all options] (y): y
Retrieving srcpackage cray-datawarp-cots-api-
access-1.0.0-20171020154045.noarch      (1/1)
Retrieving: cray-datawarp-cots-api-
access-1.0.0-20171020154045.src.rpm ..............[done]
Checking for file
conflicts: ..........................................................[done]
(1/1) Installing: cray-datawarp-cots-api-
access-1.0.0-20171020154045.src .............[done]
```

8. Change directory to the `rpmbuild` SPECS directory.

```
login# cd $(rpm --eval '%{_specdir}')
```

9. Lock down permissions on the `rpmbuild` parent directory.

```
login# chmod go-x $(rpm --eval '%{_topdir}')
```

10. Build the RPM.

WARNING: By default, the RPM built from the `cray-datawarp-cots-api-access` spec file includes a MUNGE credential file. This file is usually permissions-restricted but the built RPM is itself world-readable. In order to protect the MUNGE secret, this procedure includes steps to lock down permissions on the `rpmbuild` directories as well as the RPM itself. Treat the built RPM as any other sensitive file and be careful when copying it around.

```
login# rpmbuild -bb cray-datawarp-cots-api-access.spec
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.jOo9Ik
...
Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.ASSe2M
...
Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.vzztWf
...
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.EDeVVb
...
exit 0
```

Note that there are three switches that can be passed to `rpmbuild` in order to limit the type of configuration data that the `cray-datawarp-cots-api-access` RPM gathers. These are advanced options that do not need to be used in a typical installation. The options are:

`--without dw_configs`

Do not capture DW configuration information for use on the COTS node

`--without cray_root_cert`

Do not capture the cray root certificate for use with the DataWarp RESTful API HTTPS connection

`--without munge_credential`

Do not capture the shared secret MUNGE credential in to the RPM

11. Lock down permissions on the built RPM.

```
login# chmod go-rwx $(rpm --eval \
'%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
```

TIP: The following demonstrates how to determine the RPM location:

```
login# ls $(rpm --eval '%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
/usr/src/packages/RPMS/x86_64/cray-datawarp-cots-api-
access-1.0.0-20170830084400.x86_64.rpm
```

D. Update a COTS node local RPM repository with the new `cray-datawarp-cots-api-access` RPM

12. Copy the RPM into the previously created RPM directory (`$LOCAL_RPM_DIR`) on the COTS node (`$COTS_NODE_HOSTNAME`) .

`$COTS_NODE_HOSTNAME` and `$LOCAL_RPM_DIR` must match their previously defined value.

```
login# COTS_NODE_HOSTNAME=cots-hostname
login# LOCAL_RPM_DIR=local-rpm-dir
login# DW_COTS_RPM=$(rpm --eval '%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
login# scp $DW_COTS_RPM root@$COTS_NODE_HOSTNAME:$LOCAL_RPM_DIR
```

13. Log on to the COTS node, then refresh the `zypper` repository cache.

```
cots# zypper refresh
Retrieving repository 'common_cle_6.0up05_sles_12sp3_x86-64_ari' metadata
...

All repositories have been refreshed.
```

E. Complete the initial configuration of a COTS node for DataWarp API access

14. Install `cray-datawarp-cots-api-access` and all of its dependencies.

```
cots# zypper install cray-datawarp-cots-api-access
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW package is going to be installed:
cray-datawarp-cots-api-access

The following 6 packages are going to be upgraded:
...

The following 7 packages are not supported by their vendor:
...

..............[done]
```

15. Restart the MUNGE daemon.

```
cots# systemctl restart munge
```

F. Test DataWarp API access from a COTS node

After the `cray-datawarp-cots-api-access` RPM is installed on a COTS node, `dwcli`, `dwstat`, `dwgateway`, and `dw_wlm_cli` are able to access DataWarp on the XC. As there is no modules environment, the commands are executed with their full paths. They are:

- `/opt/cray/dws/default/bin/dwstat`
- `/opt/cray/dws/default/bin/dwcli`
- `/opt/cray/dws/default/bin/dwgateway`
- `/opt/cray/dw_wlm/default/bin/dw_wlm_cli`

Example 1:

```
cots# /opt/cray/dws/default/bin/dwstat pools
    pool units quantity    free   gran
wlm_pool bytes 59.67TiB 8.88TiB 384MiB
```

Example 2: if the cURL command-line tool is installed, it can query the DataWarp RESTful API directly. This example retrieves DataWarp pool information in JSON format:

```
cots# curl -H "Authorization: MUNGE cred=$(munge -n)" \
$(/opt/cray/dws/default/bin/dwgateway)/dw/v1/pools/
{
  "pools": [
    {
      "free": 24235,
      "granularity": 402653184,
      "id": "wlm_pool",
      "quantity": 162943,
      "units": "bytes"
    }
  ]
}
```

16. See *Troubleshoot DataWarp API Access from non-Cray Nodes* on page 19 if errors occur.

2.2 Update a non-Cray Node for DataWarp API Access Following a DataWarp Patch or Configuration Change

Prerequisites

- DataWarp API access tools were previously configured on the non-Cray node (i.e., completion of *Configure a non-Cray Node for DataWarp API Access* on page 4)

About this task

This procedure updates a commercial off-the-shelf (COTS), non-Cray node for DataWarp API access following a DataWarp patch or configuration change. See *Configure a non-Cray Node for DataWarp API Access* on page 4 to

perform an initial configuration. See *Upgrade a non-Cray Node for DataWarp API Access Following a CLE Update or Upgrade* on page 13 to perform an upgrade of the COTS configuration after a CLE upgrade on the XC Series system (XC).

This procedure enables native clients of the DataWarp RESTful API to run from non-Cray environments. It comprises the following subtasks:

A. Build the `cray-datawarp-cots-api-access` RPM
B. Update the COTS node local RPM repository with the new `cray-datawarp-cots-api-access` RPM
C. Complete the update of a COTS node for DataWarp API access
D. Test DataWarp API access from the COTS node

These subtasks are highlighted within the procedure.

Procedure

A. Build the `cray-datawarp-cots-api-access` RPM

This procedure captures configuration data and dependency information located on the XC internal login node and places it in to an RPM. The RPM is later installed on to the COTS node as part of a subsequent procedure.

1. Log on to an XC internal login node, then install the source RPM.

```
login# zypper source-install cray-datawarp-cots-api-access
Building repository 'common_cle_6.0up06_sles_12sp3_x86-64_ari'
cache ..............[done]
Building repository 'common_cle_6.0up06_sles_12sp3_x86-64_ari_updates'
cache .......[done]
...
Reading installed packages...
Loading repository data...
Resolving package dependencies...

The following source package is going to be installed:
  cray-datawarp-cots-api-access

1 source package to install.
Overall download size: 7.2 KiB. Already cached: 0 B. After the operation,
additional 11.2 KiB will be used.
Continue? [y/n/...? shows all options] (y): y
Retrieving srcpackage cray-datawarp-cots-api-
access-1.0.0-20171020154045.noarch      (1/1)
Retrieving: cray-datawarp-cots-api-
access-1.0.0-20171020154045.src.rpm ..............[done]
Checking for file
conflicts: ..........................................................[done]
(1/1) Installing: cray-datawarp-cots-api-
access-1.0.0-20171020154045.src ............[done]
```

2. Change directory to the `rpmbuild` SPECS directory.

```
login# cd $(rpm --eval '%{_specdir}')
```

3. Lock down permissions on the `rpmbuild` parent directory.

```
login# chmod go-x $(rpm --eval '%{_topdir}')
```

4. Build the RPM.

WARNING: By default, the RPM built from the `cray-datawarp-cots-api-access` spec file includes a MUNGE credential file. This file is usually permissions-restricted but the built RPM is itself world-readable. In order to protect the MUNGE secret, this procedure includes steps to lock down permissions on the `rpmbuild` directories as well as the RPM itself. Treat the built RPM as any other sensitive file and be careful when copying it around.

```
login# rpmbuild -bb cray-datawarp-cots-api-access.spec
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.jOo9Ik
...
Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.ASSe2M
...
Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.vzztWf
...
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.EDeVVb
...
exit 0
```

Note that there are three switches that can be passed to `rpmbuild` in order to limit the type of configuration data that the `cray-datawarp-cots-api-access` RPM gathers. These are advanced options that do not need to be used in a typical installation. The options are:

`--without dw_configs`

Do not capture DW configuration information for use on the COTS node

`--without cray_root_cert`

Do not capture the cray root certificate for use with the DataWarp RESTful API HTTPS connection

`--without munge_credential`

Do not capture the shared secret MUNGE credential in to the RPM

5. Lock down permissions on the built RPM.

```
login# chmod go-rwx $(rpm --eval \
'%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
```

TIP: The following demonstrates how to determine the RPM location:

```
login# ls $(rpm --eval '%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
/usr/src/packages/RPMS/x86_64/cray-datawarp-cots-api-
access-1.0.0-20170830084400.x86_64.rpm
```

B. Update the COTS node local RPM repository with the new `cray-datawarp-cots-api-access` RPM

6. Copy the RPM into the previously created RPM directory (`$LOCAL_RPM_DIR`) on the COTS node (`$COTS_NODE_HOSTNAME`) .

`$COTS_NODE_HOSTNAME` and `$LOCAL_RPM_DIR` must match their previously defined value.

```
login# COTS_NODE_HOSTNAME=cots-hostname
login# LOCAL_RPM_DIR=local-rpm-dir
login# DW_COTS_RPM=$(rpm --eval '%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
login# scp $DW_COTS_RPM root@$COTS_NODE_HOSTNAME:$LOCAL_RPM_DIR
```

7. Log on to the COTS node, then refresh the `zypper` repository cache.

```
cots# zypper refresh
Retrieving repository 'common_cle_6.0up05_sles_12sp3_x86-64_ari' metadata
...

All repositories have been refreshed.
```

C. Complete the update of a COTS node for DataWarp API access

8. Update all RPMs.

   Cray recommends updating all RPMs, rather than just the `cray-datawarp-cots-api-access` RPM.

```
cots# zypper update
```

9. Restart the MUNGE daemon.

```
cots# systemctl restart munge
```

D. Test DataWarp API access from the COTS node

After the `cray-datawarp-cots-api-access` RPM is installed on a COTS node, `dwcli`, `dwstat`, `dwgateway`, and `dw_wlm_cli` are able to access DataWarp on the XC. As there is no modules environment, the commands are executed with their full paths. They are:

- `/opt/cray/dws/default/bin/dwstat`
- `/opt/cray/dws/default/bin/dwcli`
- `/opt/cray/dws/default/bin/dwgateway`
- `/opt/cray/dw_wlm/default/bin/dw_wlm_cli`

Example 1:

```
cots# /opt/cray/dws/default/bin/dwstat pools
    pool units quantity    free   gran
wlm_pool bytes 59.67TiB 8.88TiB 384MiB
```

Example 2: if the cURL command-line tool is installed, it can query the DataWarp RESTful API directly. This example retrieves DataWarp pool information in JSON format:

```
cots# curl -H "Authorization: MUNGE cred=$(munge -n)" \
$(/opt/cray/dws/default/bin/dwgateway)/dw/v1/pools/
{
  "pools": [
    {
      "free": 24235,
      "granularity": 402653184,
      "id": "wlm_pool",
      "quantity": 162943,
      "units": "bytes"
    }
  ]
}
```

10. See *Troubleshoot DataWarp API Access from non-Cray Nodes* on page 19 if errors occur.

2.3 Upgrade a non-Cray Node for DataWarp API Access Following a CLE Update or Upgrade

Prerequisites

- DataWarp API access tools were previously configured on the non-Cray node (i.e., completion of *Configure a non-Cray Node for DataWarp API Access* on page 4)
- The XC Series system (XC) with which the non-Cray node is associated has either been updated to a new CLE update level (e.g., CLE 6.0.UP05 to CLE 6.0.UP06) or upgraded to a new CLE release level (e.g., CLE 5.2 to CLE 6.0)

About this task

This procedure upgrades a commercial off-the-shelf (COTS), non-Cray node for DataWarp API access following a CLE update or upgrade on the XC. See *Configure a non-Cray Node for DataWarp API Access* on page 4 to perform an initial configuration. See *Update a non-Cray Node for DataWarp API Access Following a DataWarp Patch or Configuration Change* on page 9 to perform an update after installing a DataWarp patch or configuration change.

This procedure enables native clients of the DataWarp RESTful API to run from non-XC environments. However, the upgrade path is dependent upon the current version of SLES on the COTS node as compared to the version used by CLE on the XC as determined in step *1* on page 13.

- If the SLES versions do not match, the upgrade path is:

  A: Exit this procedure to upgrade SLES and reconfigure the COTS node

- If the SLES versions match, the upgrade path is:

  B. Remove the existing repository
  C. Configure the COTS node to use an XC SMW as a software repository source
  D. Build the `cray-datawarp-cots-api-access` RPM
  E. Update the COTS node local RPM repository with the new `cray-datawarp-cots-api-access` RPM
  F. Complete the update of a COTS node for DataWarp API access
  G. Test DataWarp API access from the COTS node

These subtasks are highlighted within the procedure.

Procedure

1. Compare the COTS node SLES release level to the SLES release level on which the new CLE release is based.

This information is contained in the `/etc/os-release` file on both the COTS node and the XC internal login node.

The version of SLES on the COTS node must match the version used by CLE on the XC. If the CLE release on the XC uses a new version of SLES, the version of SLES on the COTS node must be upgraded to match it. Proceed based on the following:

- If the SLES versions do not match:

  A. Exit this procedure to upgrade SLES and reconfigure the COTS node

  Complete the following tasks after exiting this procedure:

  1. Upgrade SLES on the COTS node as directed in the SLES documentation.
  2. Follow the procedure *Configure a non-Cray Node for DataWarp API Access* on page 4. Because the SLES installer removed the `cray-datawarp-cots-api-access` RPM and software repositories previously configured, it is necessary to reconfigure the COTS node rather than continue with this procedure.

- If the SLES versions match, continue with this upgrade procedure.

B. Remove the repository configuration file

The repository was originally placed on the COTS node as part of *Configure a non-Cray Node for DataWarp API Access* on page 4.

2. Log on to the COTS node, then remove the repository configuration file (`$XC_REPO_FILE`) located in `/etc/zypp/repos.d/`.

   As an example, if an XC (named `seymour`) was recently upgraded to CLE 6.0.UP06, the file may be called `seymour-6.0.UP05-SLES12.3-20170821165130.repo`.

```
cots# ls /etc/zypp/repos.d
SLES12-SP3-12.3-0.repo local_rpms.repo seymour-6.0.UP05-
SLES12.3-20170821165130.repo
cots# XC_REPO_FILE=seymour-6.0.UP05-SLES12.3-20170821165130.repo
cots# rm /etc/zypp/repos.d/$XC_REPO_FILE
```

C. Configure the COTS node to use an XC Series SMW as a software repository source

This part of the procedure enables a COTS node to install RPMs hosted and mirrored by an SMW. The SMW hosting list includes SLES ISO RPM and SLES RPM update repositories, therefore, it is not necessary to maintain a separate duplicate source for these RPMs. The first step, which is optional but recommended, suggests ways to find and disable or remove these duplicate repositories.

> **TIP:** Although this procedure enables a COTS node to install any XC Series software to it via `zypper`, not all software is supported on COTS nodes. Only RPMs that come from SLES and RPMs associated with DataWarp are supported.

3. (Optional) Log on to a COTS node, then disable or remove software repositories that correspond to SLES ISO RPM material or update repositories. The proceeding steps re-establish these repositories using your SMW as a host.

   a. Review repositories.

```
cots# zypper repos
Repository priorities are without effect. All enabled repositories share the
same priority.

# | Alias              | Name               | Enabled | GPG Check | Refresh
--+--------------------+--------------------+---------+-----------+--------
1 | SLES12-SP3-12.3-0 | SLES12-SP3-12.3-0 | Yes     | (r ) Yes  | No
```

```
...
```

b. Disable repositories.

```
cots# zypper modifyrepo --disable repo
Repository 'repo' has been successfully disabled.
```

c. Permanently remove repositories.

```
non-XC# zypper removerepo repo
```

4. Log on to an XC internal login node, then capture the current LiveUpdates repository configuration.

Remember to define SYSTEM_NAME appropriately.

```
login# SYSTEM_NAME=system-name
login# CLE_RELEASE=$(grep RELEASE /etc/opt/cray/release/cle-release | cut -d = -f 2)
login# SLES_RELEASE=SLES$(lsb-release -rs)
login# SNAPSHOT=$(date +%Y%m%d%H%M%S | tr -cd '[0-9]')
login# XC_REPO_FILE=$SYSTEM_NAME-$CLE_RELEASE-$SLES_RELEASE-$SNAPSHOT.repo
login# zypper repos --export $XC_REPO_FILE
Repositories have been successfully exported to ./system-name-6.0.UP05-
SLES12.3-20170911094727.repo.
```

5. Modify the *baseurl* attribute in $XC_REPO_FILE to point to the single SMW hostname rather than multiple internal XC hostnames.

Remember to define SMW_HOSTNAME appropriately.

```
login# SMW_HOSTNAME=smw-hostname
login# XC_INTERNAL_ADDRESS=$(grep -om 1 http.*: $XC_REPO_FILE | cut -d : -f 1-2)
login# sed -i -e "s@$XC_INTERNAL_ADDRESS@http://$SMW_HOSTNAME@" $XC_REPO_FILE
login# sed -i -e '/ http/d' $XC_REPO_FILE
```

6. Copy $XC_REPO_FILE to the /etc/zypp/repos.d/ directory on the COTS node.

Remember to define COTS_NODE_HOSTNAME appropriately.

```
login# COTS_NODE_HOSTNAME=cots-hostname
login# scp $XC_REPO_FILE root@$COTS_NODE_HOSTNAME:/etc/zypp/repos.d/
```

D. Build the cray-datawarp-cots-api-access RPM

This procedure captures configuration data and dependency information located on the XC internal login node and places it in to an RPM. The RPM is later installed on to the COTS node as part of a subsequent procedure.

7. Log on to an XC internal login node, then install the source RPM.

```
login# zypper source-install cray-datawarp-cots-api-access
Building repository 'common_cle_6.0up06_sles_12sp3_x86-64_ari'
cache ..............[done]
Building repository 'common_cle_6.0up06_sles_12sp3_x86-64_ari_updates'
cache .......[done]
...
Reading installed packages...
Loading repository data...
Resolving package dependencies...
```

```
The following source package is going to be installed:
  cray-datawarp-cots-api-access

1 source package to install.
Overall download size: 7.2 KiB. Already cached: 0 B. After the operation,
additional 11.2 KiB will be used.
Continue? [y/n/...? shows all options] (y): y
Retrieving srcpackage cray-datawarp-cots-api-
access-1.0.0-20171020154045.noarch      (1/1)
Retrieving: cray-datawarp-cots-api-
access-1.0.0-20171020154045.src.rpm ..............[done]
Checking for file
conflicts: ..........................................................[done]
(1/1) Installing: cray-datawarp-cots-api-
access-1.0.0-20171020154045.src .............[done]
```

8. Change directory to the `rpmbuild` SPECS directory.

```
login# cd $(rpm --eval '%{_specdir}')
```

9. Lock down permissions on the `rpmbuild` parent directory.

```
login# chmod go-x $(rpm --eval '%{_topdir}')
```

10. Build the RPM.

WARNING: By default, the RPM built from the `cray-datawarp-cots-api-access` spec file includes a MUNGE credential file. This file is usually permissions-restricted but the built RPM is itself world-readable. In order to protect the MUNGE secret, this procedure includes steps to lock down permissions on the `rpmbuild` directories as well as the RPM itself. Treat the built RPM as any other sensitive file and be careful when copying it around.

```
login# rpmbuild -bb cray-datawarp-cots-api-access.spec
Executing(%prep): /bin/sh -e /var/tmp/rpm-tmp.jOo9Ik
...
Executing(%build): /bin/sh -e /var/tmp/rpm-tmp.ASSe2M
...
Executing(%install): /bin/sh -e /var/tmp/rpm-tmp.vzztWf
...
Executing(%clean): /bin/sh -e /var/tmp/rpm-tmp.EDeVVb
...
exit 0
```

Note that there are three switches that can be passed to `rpmbuild` in order to limit the type of configuration data that the `cray-datawarp-cots-api-access` RPM gathers. These are advanced options that do not need to be used in a typical installation. The options are:

`--without dw_configs`

Do not capture DW configuration information for use on the COTS node

`--without cray_root_cert`

Do not capture the cray root certificate for use with the DataWarp RESTful API HTTPS connection

`--without munge_credential`

Do not capture the shared secret MUNGE credential in to the RPM

11. Lock down permissions on the built RPM.

```
login# chmod go-rwx $(rpm --eval \
'%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
```

TIP: The following demonstrates how to determine the RPM location:

```
login# ls $(rpm --eval '%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
/usr/src/packages/RPMS/x86_64/cray-datawarp-cots-api-
access-1.0.0-20170830084400.x86_64.rpm
```

E. Update the COTS node local RPM repository with the new `cray-datawarp-cots-api-access` RPM

12. Copy the RPM into the previously created RPM directory (`$LOCAL_RPM_DIR`) on the COTS node (`$COTS_NODE_HOSTNAME`) .

`$COTS_NODE_HOSTNAME` and `$LOCAL_RPM_DIR` must match their previously defined value.

```
login# COTS_NODE_HOSTNAME=cots-hostname
login# LOCAL_RPM_DIR=local-rpm-dir
login# DW_COTS_RPM=$(rpm --eval '%{_rpmdir}')/noarch/cray-datawarp-cots-api-access*.rpm
login# scp $DW_COTS_RPM root@$COTS_NODE_HOSTNAME:$LOCAL_RPM_DIR
```

13. Log on to the COTS node, then refresh the `zypper` repository cache.

```
cots# zypper refresh
Retrieving repository 'common_cle_6.0up05_sles_12sp3_x86-64_ari' metadata
...

All repositories have been refreshed.
```

F. Complete the update of a COTS node for DataWarp API access

14. Update all RPMs.

Cray recommends updating all RPMs, rather than just the `cray-datawarp-cots-api-access` RPM.

```
cots# zypper update
```

15. Restart the MUNGE daemon.

```
cots# systemctl restart munge
```

F. Test DataWarp API access from the COTS node

After the `cray-datawarp-cots-api-access` RPM is installed on a COTS node, `dwcli`, `dwstat`, `dwgateway`, and `dw_wlm_cli` are able to access DataWarp on the XC. As there is no modules environment, the commands are executed with their full paths. They are:

- `/opt/cray/dws/default/bin/dwstat`
- `/opt/cray/dws/default/bin/dwcli`
- `/opt/cray/dws/default/bin/dwgateway`
- `/opt/cray/dw_wlm/default/bin/dw_wlm_cli`

Example 1:

```
cots# /opt/cray/dws/default/bin/dwstat pools
    pool units quantity    free   gran
wlm_pool bytes 59.67TiB 8.88TiB 384MiB
```

Example 2: if the cURL command-line tool is installed, it can query the DataWarp RESTful API directly. This example retrieves DataWarp pool information in JSON format:

```
cots# curl -H "Authorization: MUNGE cred=$(munge -n)" \
$(/opt/cray/dws/default/bin/dwgateway)/dw/v1/pools/
{
  "pools": [
    {
      "free": 24235,
      "granularity": 402653184,
      "id": "wlm_pool",
      "quantity": 162943,
      "units": "bytes"
    }
  ]
}
```

16. See *Troubleshoot DataWarp API Access from non-Cray Nodes* on page 19 if errors occur.

3 Troubleshoot DataWarp API Access from non-Cray Nodes

The following list contains issues and possible solutions for certain problems that may occur when trying to access the DataWarp RESTful API from a commercial off-the-shelf (COTS), non-Cray node.

- `dwgateway` states that Gateway retrieval failed
  - Verify that the XC Series system (XC) is booted.
  - Verify that a network route exists between the COTS node and the XC internal login nodes.
  - Extract hostnames from the URLs in the `/etc/opt/cray/dws/dwrest_gw.conf` file on the COTS node. Verify that hostname resolves and fix as needed.
- `dwcli` or `dwstat` give MUNGE-related errors
  - Verify that time is synchronized between the COTS node and the XC internal login nodes.
  - Verify that MUNGE is running on the COTS node.
  - Verify that `/etc/munge/munge.key` is identical on the COTS node and XC internal login nodes.
- The `/etc/opt/cray/dws/dwrest_gw.conf` file is empty
  - Verify that external API gateway nodes were configured during DataWarp installation; if not, see *XC™ Series DataWarp™ Installation and Administration Guide* (S-2564) for the initial DataWarp installation procedure.

Contact Cray service personnel if issues persist.