# System Deployment

The DAOS deployment workflow requires to start the DAOS server instances early on to enable administrators to perform remote operations in parallel across multiple storage nodes via the dmg management utility. Security is guaranteed via the use of certificates. The first type of commands run after installation include network and storage hardware provisioning and would typically be run from a login node.

After daos\_server instances have been started on each storage node for the first time, dmg storage prepare will set PMem storage into the necessary state for use with DAOS. Then dmg storage format formats persistent storage devices (specified in the server configuration file) on the storage nodes and writes necessary metadata before starting DAOS Engine processes that will operate across the fabric.

To sum up, the typical workflow of a DAOS system deployment consists of the following steps:

* Configure and start the [DAOS server](#daos-server-setup).
* [Provision Hardware](#hardware-provisioning) on all the storage nodes via the dmg utility.
* [Format](#storage-formatting) the DAOS system
* [Set up and start the agent](#agent-startup) on the client nodes
* [Validate](#system-validation) that the DAOS system is operational

Note that starting the DAOS server instances can be performed automatically on boot if start-up scripts are registered with systemd.

The following subsections will cover each step in more detail.

## DAOS Server Setup

First of all, the DAOS server should be started to allow remote administration command to be executed via the dmg tool. This section describes the minimal DAOS server configuration and how to start it on all the storage nodes.

### Example RPM Deployment Workflow

A recommended workflow to get up and running is as follows:

* Install DAOS Server RPMs - daos\_server systemd services will start in listening mode which means DAOS Engine processes will not be started as the server config file (default location at /etc/daos/daos\_server.yml) has not yet been populated.
* Run dmg config generate -l <hostset> -a <access\_points> across the entire hostset (all the storage servers that are now running the daos\_server service after RPM install). The command will only generate a config if hardware setups on all the hosts are similar and have been given sensible NUMA mappings. Adjust the hostset until you have a set with homogeneous hardware configurations.
* Once a recommended config file can be generated, copy it to the server config file default location (/etc/daos/daos\_server.yml) on each DAOS Server host and restart all daos\_server services. An example command to restart the services is clush -w machines-[118-121,130-133] "sudo systemctl restart daos\_server". The services should prompt for format on restart and after format is triggered from dmg, the DAOS Engine processes should start.

### Server Configuration File

The daos\_server configuration file is parsed when starting the daos\_server process. The configuration file location can be specified on the command line (daos\_server -h for usage) or it will be read from the default location (/etc/daos/daos\_server.yml).

Parameter descriptions are specified in [daos\_server.yml](https://github.com/daos-stack/daos/blob/master/utils/config/daos_server.yml) and example configuration files in the [examples](https://github.com/daos-stack/daos/tree/master/utils/config/examples) directory.

Any option supplied to daos\_server as a command line option or flag will take precedence over equivalent configuration file parameter.

For convenience, active parsed configuration values are written to a temporary file for reference, and the location will be written to the log.

#### Configuration Options

The example configuration file lists the default empty configuration, listing all the options (living documentation of the config file). Live examples are available at <https://github.com/daos-stack/daos/tree/master/utils/config/examples>

The location of this configuration file is determined by first checking for the path specified through the -o option of the daos\_server command line, if unspecified then /etc/daos/daos\_server.yml is used.

Refer to the example configuration file ( [daos\_server.yml](https://github.com/daos-stack/daos/blob/master/utils/config/daos_server.yml) ) for latest information and examples.

At this point of the process, the servers: and provider: section of the yaml file can be left blank and will be populated in the subsequent sections.

#### Auto generate configuration file

DAOS can attempt to produce a server configuration file that makes optimal use of hardware on a given set of hosts through the ‘dmg config generate’ command:

$ dmg config generate --help  
ERROR: dmg: Usage:  
 dmg [OPTIONS] config generate [generate-OPTIONS]  
  
Application Options:  
...  
 -l, --host-list= comma separated list of addresses <ipv4addr/hostname>  
...  
  
[generate command options]  
 -a, --access-points= Comma separated list of access point  
 addresses <ipv4addr/hostname>  
 -e, --num-engines= Set the number of DAOS Engine sections to be  
 populated in the config file output. If unset  
 then the value will be set to the number of  
 NUMA nodes on storage hosts in the DAOS  
 system.  
 -s, --min-ssds= Minimum number of NVMe SSDs required per DAOS  
 Engine (SSDs must reside on the host that is  
 managing the engine). Set to 0 to generate a  
 config with no NVMe. (default: 1)  
 -c, --net-class=[best-available|ethernet|infiniband] Network class preferred (default:  
 best-available)

The command will output recommended config file if supplied requirements are met. Requirements will be derived based on the number of NUMA nodes present on the hosts if ‘–num-engines’ is not specified on the commandline.

* ‘–num-engines’ specifies the number of engine sections to populate in the config file output. Each section will specify a persistent memory (PMem) block devices that must be present on the host in addition to a fabric network interface and SSDs all bound to the same NUMA node. If not set explicitly on the commandline, default is the number of NUMA nodes detected on the host.
* ‘–min-ssds’ specifies the minimum number of NVMe SSDs per-engine that need to be present on each host. For each engine entry in the generated config, at least this number of SSDs must be bound to the NUMA node that matches the affinity of the PMem device and fabric network interface associated with the engine. If not set on the commandline, default is “1”. If set to “0” NVMe SSDs will not be added to the generated config and SSD validation will be disabled.
* ‘–net-class’ specifies preference for network interface class, options are ‘ethernet’, ‘infiband’ or ‘best-available’. ‘best-available’ will attempt to choose the most performant (as judged by libfabric) sets of interfaces and supported provider that match the number and NUMA affinity of PMem devices. If not set on the commandline, default is “best-available”.

The configuration file that is generated by the command and output to stdout can be copied to a file and used on the relevant hosts and used as server config to determine the starting environment for ‘daos\_server’ instances.

Config file output will not be generated in the following cases: - PMem device count, capacity or NUMA mappings differ on any of the hosts in the hostlist (the hostlist can be specified either in the ‘dmg’ config file or on the commandline). - NVMe SSD count, PCI address distribution or NUMA affinity differs on any of the hosts in the host list. - NUMA node count can’t be detected on the hosts or differs on any host in the host list. - PMem device count or NUMA affinity doesn’t meet the ‘num-engines’ requirement. - NVMe device count or NUMA affinity doesn’t meet the ‘min-ssds’ requirement. - network device count or NUMA affinity doesn’t match the configured PMem devices, taking into account any specified network device class preference (ethernet or infiniband).

Some CentOS 7.x kernels from before the 7.9 release were known to have a defect that prevented ndctl from being able to report the NUMA affinity for a namespace. This prevents generation of dual engine configs using dmg config generate when running with one of the above-mentioned affected kernels.

#### Certificate Configuration

The DAOS security framework relies on certificates to authenticate components and administrators in addition to encrypting DAOS control plane communications. A set of certificates for a given DAOS system may be generated by running the gen\_certificates.sh script provided with the DAOS software if there is not an existing TLS certificate infrastructure. The gen\_certificates.sh script uses the openssl tool to generate all of the necessary files. We highly recommend using OpenSSL Version 1.1.1h or higher as keys and certificates generated with earlier versions are vulnerable to attack.

When DAOS is installed from RPMs, this script is provided in the base daos RPM, and may be invoked in the directory to which the certificates will be written. As part of the generation process, a new local Certificate Authority is created to handle certificate signing, and three role certificates are created:

# /usr/lib64/daos/certgen/gen\_certificates.sh  
Generating Private CA Root Certificate  
Private CA Root Certificate created in ./daosCA  
...  
Generating Server Certificate  
Required Server Certificate Files:  
 ./daosCA/certs/daosCA.crt  
 ./daosCA/certs/server.key  
 ./daosCA/certs/server.crt  
...  
Generating Agent Certificate  
Required Agent Certificate Files:  
 ./daosCA/certs/daosCA.crt  
 ./daosCA/certs/agent.key  
 ./daosCA/certs/agent.crt  
...  
Generating Admin Certificate  
Required Admin Certificate Files:  
 ./daosCA/certs/daosCA.crt  
 ./daosCA/certs/admin.key  
 ./daosCA/certs/admin.crt

The files generated under ./daosCA should be protected from unauthorized access and preserved for future use.

The generated keys and certificates must then be securely distributed to all nodes participating in the DAOS system (servers, clients, and admin nodes). Permissions for these files should be set to prevent unauthorized access to the keys and certificates.

Client nodes require: - CA root cert - Agent cert - Agent key

Administrative nodes require: - CA root cert - Admin cert - Admin key

Server nodes require: - CA root cert - Server cert - Server key - All valid agent certs in the DAOS system (in the client cert directory, see config file below)

After the certificates have been securely distributed, the DAOS configuration files must be updated in order to enable authentication and secure communications. These examples assume that the configuration and certificate files have been installed under /etc/daos:

# /etc/daos/daos\_server.yml (servers)  
  
transport\_config:  
 # Location where daos\_server will look for Client certificates  
 client\_cert\_dir: /etc/daos/certs/clients  
 # Custom CA Root certificate for generated certs  
 ca\_cert: /etc/daos/certs/daosCA.crt  
 # Server certificate for use in TLS handshakes  
 cert: /etc/daos/certs/server.crt  
 # Key portion of Server Certificate  
 key: /etc/daos/certs/server.key

# /etc/daos/daos\_agent.yml (clients)  
  
transport\_config:  
 # Custom CA Root certificate for generated certs  
 ca\_cert: /etc/daos/certs/daosCA.crt  
 # Agent certificate for use in TLS handshakes  
 cert: /etc/daos/certs/agent.crt  
 # Key portion of Agent Certificate  
 key: /etc/daos/certs/agent.key

# /etc/daos/daos\_control.yml (dmg/admin)  
  
transport\_config:  
 # Custom CA Root certificate for generated certs  
 ca\_cert: /etc/daos/certs/daosCA.crt  
 # Admin certificate for use in TLS handshakes  
 cert: /etc/daos/certs/admin.crt  
 # Key portion of Admin Certificate  
 key: /etc/daos/certs/admin.key

### Server Startup

One instance of the daos\_server process is to be started per storage node. The server can be started either individually (e.g. independently on each storage node via systemd) or collectively (e.g. pdsh, mpirun or as a Kubernetes Pod).

#### Parallel Launcher

Practically any parallel launcher can be used to start the DAOS server collectively on a set of storage nodes. pdsh, clush and orterun are most commonly used.

$ clush -w <server\_list> -o "-t -t" daos\_server start -o <config\_file>`

will launch daos\_server on the specified hosts connecting to the port parameter value specified in the server config file. If the number of storage node exceed the default fanout value, then “-f” followed by the number of storage nodes should be used.

Similarly, pdsh can be used:

$ pdsh -w <server\_list> daos\_server start -o <config\_file>`

As for orterun, the list of storage nodes can be specified on the command line via the -H option. To start the DAOS server, run:

$ orterun --map-by node --mca btl tcp,self --mca oob tcp -np <num\_servers>  
-H <server\_list> --enable-recovery daos\_server start -o <config\_file>

The –enable-recovery is required for fault tolerance to guarantee that the fault of one server does not cause the others to be stopped.

The –allow-run-as-root option can be added to the command line to allow the daos\_server to run with root privileges on each storage nodes (for example when needing to perform privileged tasks relating to storage format). See the orterun(1) man page for additional options.

#### Systemd Integration

The DAOS Server can be started as a systemd service. The DAOS Server unit file is installed in the correct location when installing from RPMs. The DAOS Server will be run as daos-server user which will be created during RPM install.

If you wish to use systemd with a development build, you must copy the service file from utils/systemd to /usr/lib/systemd/system. Once the file is copied, modify the ExecStart line to point to your daos\_server binary.

After modifying ExecStart, run the following command:

$ sudo systemctl daemon-reload

Once the service file is installed you can start daos\_server with the following commands:

$ systemctl enable daos\_server.service  
$ systemctl start daos\_server.service

To check the component status use:

$ systemctl status daos\_server.service

If DAOS Server failed to start, check the logs with:

$ journalctl --unit daos\_server.service

After RPM install, daos\_server service starts automatically running as user “daos”. The server config is read from /etc/daos/daos\_server.yml and certificates are read from /etc/daos/certs. With no other admin intervention other than the loading of certificates, daos\_server will enter a listening state enabling discovery of storage and network hardware through the dmg tool without any I/O Engines specified in the configuration file. After device discovery and provisioning, an updated configuration file with a populated per-engine section can be stored in /etc/daos/daos\_server.yml, and after reestarting the daos\_server service it is then ready for the storage to be formatted.

#### Kubernetes Pod

DAOS service integration with Kubernetes is planned and will be supported in a future DAOS version.

## DAOS Server Remote Access

Remote tasking of the DAOS system and individual DAOS Server processes can be performed via the dmg utility.

To set the addresses of which DAOS Servers to task, provide either: - -l <hostlist> on the commandline when invoking, or - hostlist: <hostlist> in the control configuration file [daos\_control.yml](https://github.com/daos-stack/daos/blob/master/utils/config/daos_control.yml)

Where <hostlist> represents a slurm-style hostlist string e.g. foo-1[28-63],bar[256-511]. The first entry in the hostlist (after alphabetic then numeric sorting) will be assumed to be the access point as set in the server configuration file.

Local configuration files stored in the user directory will be used in preference to the default location e.g. ~/.daos\_control.yml.

## Hardware Provisioning

Once the DAOS server started, the storage and network can be configured on the storage nodes via the dmg utility.

### SCM Preparation

This section addresses how to verify that PMem (Intel(R) Optane(TM) persistent memory) modules are correctly installed on the storage nodes and how to configure in interleaved mode to be used by DAOS. Instructions for other types of SCM may be covered in the future.

Provisioning the SCM occurs by configuring PMem modules in interleaved memory regions (interleaved mode) in groups of modules local to a specific socket (NUMA), and resultant nvdimm namespaces are defined by a device identifier (e.g., /dev/pmem0).

PMem preparation is required once per DAOS installation and requires the DAOS Control Servers to be running as root.

This step requires a reboot to enable PMem resource allocation changes to be read by BIOS.

PMem preparation can be performed from the management tool dmg storage prepare --scm-only or using the Control Server directly sudo daos\_server storage prepare --scm-only.

The first time the command is run, the SCM interleaved regions will be created as resource allocations on any available PMem modules (one region per NUMA node/socket). The regions are activated after BIOS reads the new resource allocations. Upon completion, the storage prepare command will prompt the admin to reboot the storage node(s) in order for the BIOS to activate the new storage allocations. The storage prepare command does not initiate the reboot itself.

After running the command a reboot will be required, then the Control Servers will then need to be started again and the command run for a second time to expose the namespace device to be used by DAOS.

Example usage:

* dmg -l wolf-[118-121,130-133] -i storage prepare --scm-only after running, the user should be prompted for a reboot.
* clush -w wolf-[118-121,130-133] reboot
* clush -w wolf-[118-121,130-133] daos\_server start -o utils/config/examples/daos\_server\_sockets.yml
* dmg -l wolf-[118-121,130-133] -i storage prepare --scm-only after running, /dev/pmemX devices should be available on each of the hosts.

sudo daos\_server storage prepare --scm-only should be run for a second time after system reboot to create the pmem kernel devices (/dev/pmemX namespaces created on the new SCM regions).

On the second run, one namespace per region is created, and each namespace may take up to a few minutes to create. Details of the pmem devices will be displayed in JSON format on command completion.

Example output from the initial call (with the SCM modules set to default MemoryMode):

Memory allocation goals for SCM will be changed and namespaces modified, this  
will be a destructive operation. ensure namespaces are unmounted and SCM is  
otherwise unused.  
A reboot is required to process new memory allocation goals.

Example output from the subsequent call (SCM modules configured to interleaved mode, and host rebooted):

Memory allocation goals for SCM will be changed and namespaces modified. This  
will be a destructive operation. Ensure namespaces are unmounted and the SCM  
is otherwise unused.  
creating SCM namespace, may take a few minutes...  
creating SCM namespace, may take a few minutes...  
Persistent memory kernel devices:  
[{UUID:5d2f2517-9217-4d7d-9c32-70731c9ac11e Blockdev:pmem1 Dev:namespace1.0 NumaNode:1} {UUID:2bfe6c40-f79a-4b8e-bddf-ba81d4427b9b Blockdev:pmem0 Dev:namespace0.0 NumaNode:0}]

Upon successful creation of the pmem devices, the Intel(R) Optane(TM) persistent memory is configured and one can move on to the next step.

If required, the pmem devices can be destroyed via the –reset option:

sudo daos\_server [<app\_opts>] storage prepare [--scm-only|-s] --reset [<cmd\_opts>]

All namespaces are disabled and destroyed. The SCM regions are removed by resetting modules into “MemoryMode” through resource allocations.

Note that undefined behavior may result if the namespaces/pmem kernel devices are mounted before running reset (as per the printed warning).

A subsequent reboot is required for BIOS to read the new resource allocations.

Example output when resetting the SCM modules:

Memory allocation goals for SCM will be changed and namespaces modified, this  
will be a destructive operation. ensure namespaces are unmounted and SCM is  
otherwise unused.  
removing SCM namespace, may take a few minutes...  
removing SCM namespace, may take a few minutes...  
resetting SCM memory allocations  
A reboot is required to process new memory allocation goals.

### Storage Selection

While the DAOS server auto-detects all the usable storage, the administrator will still be provided with the ability through the configuration file (see next section) to whitelist or blacklist the storage devices to be (or not) used. This section covers how to manually detect the storage devices potentially usable by DAOS to populate the configuration file when the administrator wants to have finer control over the storage selection.

dmg storage scan can be run to query remote running daos\_server processes over the management network.

sudo daos\_server storage scan can be used to query daos\_server directly (scans locally-attached SSDs and Intel Persistent Memory Modules usable by DAOS). Output will be equivalent running dmg storage scan --verbose remotely.

bash-4.2$ dmg storage scan  
Hosts SCM Total NVMe Total  
----- --------- ----------  
wolf-[71-72] 6.4 TB (2 namespaces) 3.1 TB (3 controllers)  
  
bash-4.2$ dmg storage scan --verbose  
------------  
wolf-[71-72]  
------------  
SCM Namespace Socket ID Capacity  
------------- --------- --------  
pmem0 0 3.2 TB  
pmem1 1 3.2 TB  
  
NVMe PCI Model FW Revision Socket ID Capacity  
-------- ----- ----------- --------- --------  
0000:81:00.0 INTEL SSDPED1K750GA E2010325 1 750 GB  
0000:87:00.0 INTEL SSDPEDMD016T4 8DV10171 1 1.6 TB  
0000:da:00.0 INTEL SSDPED1K750GA E2010325 1 750 GB

The NVMe PCI field above is what should be used in the server configuration file to identified NVMe SSDs.

Devices with the same NUMA node/socket should be used in the same per-engine section of the server configuration file for best performance.

For further info on command usage run dmg storage --help.

SSD health state can be verified via dmg storage scan --nvme-health:

bash-4.2$ dmg storage scan --nvme-health  
-------  
wolf-71  
-------  
PCI:0000:81:00.0 Model:INTEL SSDPED1K750GA FW:E2010325 Socket:1 Capacity:750 GB  
 Health Stats:  
 Temperature:318K(44.85C)  
 Controller Busy Time:0s  
 Power Cycles:15  
 Power On Duration:10402h0m0s  
 Unsafe Shutdowns:13  
 Error Count:0  
 Media Errors:0  
 Read Errors:0  
 Write Errors:0  
 Unmap Errors:0  
 Checksum Errors:0  
 Error Log Entries:0  
 Critical Warnings:  
 Temperature: OK  
 Available Spare: OK  
 Device Reliability: OK  
 Read Only: OK  
 Volatile Memory Backup: OK  
  
PCI:0000:da:00.0 Model:INTEL SSDPED1K750GA FW:E2010325 Socket:1 Capacity:750 GB  
 Health Stats:  
 Temperature:320K(46.85C)  
 Controller Busy Time:0s  
 Power Cycles:15  
 Power On Duration:10402h0m0s  
 Unsafe Shutdowns:13  
 Error Count:0  
 Media Errors:0  
 Read Errors:0  
 Write Errors:0  
 Unmap Errors:0  
 Checksum Errors:0  
 Error Log Entries:0  
 Critical Warnings:  
 Temperature: OK  
 Available Spare: OK  
 Device Reliability: OK  
 Read Only: OK  
 Volatile Memory Backup: OK  
  
-------  
wolf-72  
-------  
PCI:0000:81:00.0 Model:INTEL SSDPED1K750GA FW:E2010435 Socket:1 Capacity:750 GB  
 Health Stats:  
 Temperature:316K(42.85C)  
 Controller Busy Time:8m0s  
 Power Cycles:23  
 Power On Duration:10399h0m0s  
 Unsafe Shutdowns:18  
 Error Count:0  
 Media Errors:0  
 Read Errors:0  
 Write Errors:0  
 Unmap Errors:0  
 Checksum Errors:0  
 Error Log Entries:0  
 Critical Warnings:  
 Temperature: OK  
 Available Spare: OK  
 Device Reliability: OK  
 Read Only: OK  
 Volatile Memory Backup: OK  
  
PCI:0000:da:00.0 Model:INTEL SSDPED1K750GA FW:E2010435 Socket:1 Capacity:750 GB  
 Health Stats:  
 Temperature:320K(46.85C)  
 Controller Busy Time:1m0s  
 Power Cycles:23  
 Power On Duration:10399h0m0s  
 Unsafe Shutdowns:19  
 Error Count:0  
 Media Errors:0  
 Read Errors:0  
 Write Errors:0  
 Unmap Errors:0  
 Checksum Errors:0  
 Error Log Entries:0  
 Critical Warnings:  
 Temperature: OK  
 Available Spare: OK  
 Device Reliability: OK  
 Read Only: OK  
 Volatile Memory Backup: OK

The next step consists of adjusting in the server configuration the storage devices that should be used by DAOS. The servers section of the yaml is a list specifying details for each DAOS I/O instance to be started on the host (currently a maximum of 2 per host is imposed).

Devices with the same NUMA rating/node/socket should be colocated on a single DAOS I/O instance where possible. [more details](#server-configuration)

* bdev\_list should be populated with NVMe PCI addresses
* scm\_list should be populated with PMem interleaved set namespaces (e.g. /dev/pmem1)
* DAOS Control Servers will need to be restarted on all hosts after updates to the server configuration file.
* Pick one host in the system and set access\_points to list of that host’s hostname or IP address (don’t need to specify port). This will be the host which bootstraps the DAOS management service (MS).

To illustrate, assume a cluster with homogeneous hardware configurations that returns the following from scan for each host:

[daos@wolf-72 daos\_m]$ dmg -l wolf-7[1-2] -i storage scan --verbose  
-------  
wolf-7[1-2]  
-------  
SCM Namespace Socket ID Capacity  
------------- --------- --------  
pmem0 0 2.90TB  
pmem1 1 2.90TB  
  
NVMe PCI Model FW Revision Socket ID Capacity  
-------- ----- ----------- --------- --------  
0000:81:00.0 INTEL SSDPED1K750GA E2010325 0 750.00GB  
0000:87:00.0 INTEL SSDPEDMD016T4 8DV10171 0 1.56TB  
0000:da:00.0 INTEL SSDPED1K750GA E2010325 1 750.00GB

In this situation, the configuration file servers section could be populated as follows:

<snip>  
port: 10001  
access\_points: ["wolf-71"] # <----- updated  
<snip>  
servers:  
-  
 targets: 8 # number of I/O service threads per-engine  
 first\_core: 0 # offset of the first core for service threads  
 nr\_xs\_helpers: 2 # count of I/O offload threads per target  
 fabric\_iface: eth0 # map to OFI\_INTERFACE=eth0  
 fabric\_iface\_port: 31416 # map to OFI\_PORT=31416  
 log\_mask: ERR # map to D\_LOG\_MASK=ERR  
 log\_file: /tmp/server.log # map to D\_LOG\_FILE=/tmp/server.log  
 env\_vars: # influence DAOS I/O Engine behavior by setting env variables  
 - DAOS\_MD\_CAP=1024  
 - CRT\_CTX\_SHARE\_ADDR=0  
 - CRT\_TIMEOUT=30  
 - FI\_SOCKETS\_MAX\_CONN\_RETRY=1  
 - FI\_SOCKETS\_CONN\_TIMEOUT=2000  
 scm\_mount: /mnt/daos # map to -s /mnt/daos  
 scm\_class: dcpm  
 scm\_list: [/dev/pmem0] # <----- updated  
 bdev\_class: nvme  
 bdev\_list: ["0000:87:00.0", "0000:81:00.0"] # <----- updated  
-  
 targets: 8 # number of I/O service threads per-engine  
 first\_core: 0 # offset of the first core for service threads  
 nr\_xs\_helpers: 2 # count of I/O offload threads per target  
 fabric\_iface: eth0 # map to OFI\_INTERFACE=eth0  
 fabric\_iface\_port: 31416 # map to OFI\_PORT=31416  
 log\_mask: ERR # map to D\_LOG\_MASK=ERR  
 log\_file: /tmp/server.log # map to D\_LOG\_FILE=/tmp/server.log  
 env\_vars: # influence DAOS I/O Engine behavior by setting env variables  
 - DAOS\_MD\_CAP=1024  
 - CRT\_CTX\_SHARE\_ADDR=0  
 - CRT\_TIMEOUT=30  
 - FI\_SOCKETS\_MAX\_CONN\_RETRY=1  
 - FI\_SOCKETS\_CONN\_TIMEOUT=2000  
 scm\_mount: /mnt/daos # map to -s /mnt/daos  
 scm\_class: dcpm  
 scm\_list: [/dev/pmem1] # <----- updated  
 bdev\_class: nvme  
 bdev\_list: ["0000:da:00.0"] # <----- updated  
<end>

### Network Scan and Configuration

The daos\_server supports the network scan function to display the network interfaces, related OFI fabric providers and associated NUMA node for each device. This information is used to configure the global fabric provider and the unique local network interface for each I/O Engine instance on this node. This section will help you determine what to provide for the provider, fabric\_iface and pinned\_numa\_node entries in the daos\_server.yml file.

The following commands are typical examples:

$ daos\_server network scan  
$ daos\_server network scan -p all  
$ daos\_server network scan -p ofi+sockets  
$ daos\_server network scan --provider 'ofi+verbs;ofi\_rxm'

In the early stages when a daos\_server has not yet been fully configured and lacks a declaration of the system’s fabric provider, it may be helpful to view an unfiltered list of scan results.

Use either of these daos\_server commands in the early stages to accomplish this goal:

$ daos\_server network scan  
$ daos\_server network scan -p all

Typical network scan results look as follows:

$ daos\_server network scan -p all  
---------  
localhost  
---------  
  
 -------------  
 NUMA Socket 0  
 -------------  
  
 Provider Interfaces  
 -------- ----------  
 ofi+verbs;ofi\_rxm ib0  
 ofi+tcp;ofi\_rxm ib0, eth0  
 ofi+verbs ib0  
 ofi+tcp ib0, eth0  
 ofi+sockets ib0, eth0  
 ofi+psm2 ib0  
  
 -------------  
 NUMA Socket 1  
 -------------  
  
 Provider Interfaces  
 -------- ----------  
 ofi+verbs;ofi\_rxm ib1  
 ofi+tcp;ofi\_rxm ib1  
 ofi+verbs ib1  
 ofi+tcp ib1  
 ofi+sockets ib1  
 ofi+psm2 ib1

Use one of these providers to configure the provider in the daos\_server.yml. Only one provider may be specified for the entire DAOS installation. Client nodes must be capable of communicating to the daos\_server nodes via the same provider. Therefore, it is helpful to choose network settings for the daos\_server that are compatible with the expected client node configuration.

After the daos\_server.yml file has been edited and contains a provider, subsequent daos\_server network scan commands will filter the results based on that provider. If it is desired to view an unfiltered list again, issue daos\_server network scan -p all.

Regardless of the provider in the daos\_server.yml file, the results may be filtered to the specified provider with the command daos\_server network scan -p ofi\_provider where ofi\_provider is one of the available providers from the list.

The results of the network scan may be used to help configure the I/O Engine instances for this DAOS Server.

Each I/O Engine instance is configured with a unique fabric\_iface and optional pinned\_numa\_node. The interfaces and NUMA Sockets listed in the scan results map to the daos\_server.yml fabric\_iface and pinned\_numa\_node respectively. The use of pinned\_numa\_node is optional, but recommended for best performance. When specified with the value that matches the network interface, the I/O Engine will bind itself to that NUMA node and to cores purely within that NUMA node. This configuration yields the fastest access to that network device.

### Changing Network Providers

Information about the network configuration is stored as metadata on the DAOS storage.

If, after initial deployment, the provider must be changed, it is necessary to reformat the storage devices using dmg storage format after the configuration file has been updated with the new provider.

## Network Scanning All DAOS Server Nodes

While the daos\_server network scan is useful for scanning the localhost, it does not provide results for any other daos\_server instance on the network. The DAOS Management tool, dmg, is used for that purpose. The network scan operates the same way as the daos\_server network scan, however, to use the dmg tool, at least one known daos\_server instance must be running.

The command dmg network scan performs a query over all daos\_servers in the daos\_control.yml hostlist. By default, the scan will return results that are filtered by the provider that is specified in the daos\_server.yml. Like the daos\_server network scan, the dmg network scan supports the optional -p/--provider where a different provider may be specified, or all for an unfiltered list that is unrelated to what was already configured on the daos\_server installation.

dmg network scan  
-------  
wolf-29  
-------  
  
 -------------  
 NUMA Socket 1  
 -------------  
  
 Provider Interfaces  
 -------- ----------  
 ofi+sockets ib1  
  
---------  
localhost  
---------  
  
 -------------  
 NUMA Socket 0  
 -------------  
  
 Provider Interfaces  
 -------- ----------  
 ofi+sockets ib0, eth0  
  
 -------------  
 NUMA Socket 1  
 -------------  
  
 Provider Interfaces  
 -------- ----------  
 ofi+sockets ib1

## Provider Configuration and Debug

To aid in provider configuration and debug, it may be helpful to run the fi\_pingpong test (delivered as part of OFI/libfabric). To run that test, determine the name of the provider to test usually by removing the “ofi+” prefix from the network scan provider data. Although the “ofi+” prefix is required in daos\_server.yml, it cannot be used when specifying the provider for fi\_pingpong.

Then, the fi\_pingpong test can be used to verify that the targeted OFI provider works fine:

node1$ fi\_pingpong -p psm2  
  
node2$ fi\_pingpong -p psm2 ${IP\_ADDRESS\_NODE1}  
  
bytes #sent #ack total time MB/sec usec/xfer Mxfers/sec  
64 10 =10 1.2k 0.00s 21.69 2.95 0.34  
256 10 =10 5k 0.00s 116.36 2.20 0.45  
1k 10 =10 20k 0.00s 379.26 2.70 0.37  
4k 10 =10 80k 0.00s 1077.89 3.80 0.26  
64k 10 =10 1.2m 0.00s 2145.20 30.55 0.03  
1m 10 =10 20m 0.00s 8867.45 118.25 0.01

## Storage Formatting

Once the daos\_server has been restarted with the correct storage devices and network interface to use, one can move to the format phase. When daos\_server is started for the first time, it enters “maintenance mode” and waits for a dmg storage format call to be issued from the management tool. This remote call will trigger the formatting of the locally attached storage on the host for use with DAOS using the parameters defined in the server config file.

dmg -i -l <host>[,...] storage format will normally be run on a login node specifying a hostlist (-l <host>[,...]) of storage nodes with SCM/PMem modules and NVMe SSDs installed and prepared.

Upon successful format, DAOS Control Servers will start DAOS IO instances that have been specified in the server config file.

Successful start-up is indicated by the following on stdout: DAOS I/O Engine (v0.8.0) process 433456 started on rank 1 with 8 target, 2 helper XS per target, firstcore 0, host wolf-72.wolf.hpdd.intel.com.

### SCM Format

When the command is run, the pmem kernel devices created on SCM/PMem regions are formatted and mounted based on the parameters provided in the server config file.

* scm\_mount specifies the location of the mountpoint to create.
* scm\_class can be set to ram to use a tmpfs in the situation that no SCM/PMem is available (scm\_size dictates the size of tmpfs in GB), when set to dcpm the device specified under scm\_list will be mounted at scm\_mount path.

### NVMe Format

When the command is run, NVMe SSDs are formatted and set up to be used by DAOS based on the parameters provided in the server config file.

bdev\_class can be set to nvme to use actual NVMe devices with SPDK for DAOS storage. Other bdev\_class values can be used for emulation of NVMe storage as specified in the server config file. bdev\_list identifies devices to use with a list of PCI addresses (this can be populated after viewing results from storage scan command).

After the format command is run, the path specified by the server configuration file scm\_mount parameter should be mounted and should contain a file named daos\_nvme.conf. The file should describe the devices with PCI addresses as listed in the bdev\_list parameter of the server config file. The presence and contents of the file indicate that the specified NVMe SSDs have been configured correctly for use with DAOS.

The contents of the NVMe SSDs listed in the server configuration file bdev\_list parameter will be reset on format.

### Server Format

Before the format command is run, no DAOS metadata should exist under the path specified by scm\_mount parameter in the server configuration file.

After the storage format command is run, the path specified by the server configuration file scm\_mount parameter should be mounted and should contain the necessary DAOS metadata indicating that the server has been formatted.

When starting, daos\_server will skip maintenance mode and attempt to start I/O Engines if valid DAOS metadata is found in scm\_mount.

## Agent Setup

This section addresses how to configure the DAOS agents on the storage nodes before starting it.

### Agent Certificate Generation

The DAOS security framework relies on certificates to authenticate administrators. The security infrastructure is currently under development and will be delivered in DAOS v1.0. Initial support for certificates has been added to DAOS and can be disabled either via the command line or in the DAOS Agent configuration file. Currently, the easiest way to disable certificate support is to pass the -i flag to daos\_agent.

### Agent Configuration File

The daos\_agent configuration file is parsed when starting the daos\_agent process. The configuration file location can be specified on the command line (daos\_agent -h for usage) or default location (install/etc/daos\_agent.yml). If installed from rpms the default location is (/etc/daos/daos\_agent.yml).

Parameter descriptions are specified in [daos\_agent.yml](https://github.com/daos-stack/daos/blob/master/utils/config/daos_agent.yml).

Any option supplied to daos\_agent as a command line option or flag will take precedence over equivalent configuration file parameter.

For convenience, active parsed config values are written to a temporary file for reference, and the location will be written to the log.

The following section lists the format, options, defaults, and descriptions available in the configuration file.

The example configuration file lists the default empty configuration listing all the options (living documentation of the config file). Live examples are available [here](https://github.com/daos-stack/daos/tree/master/utils/config).

The location of this configuration file is determined by first checking for the path specified through the -o option of the daos\_agent command line, if not set then /etc/daos/daos\_agent.yml is used.

Refer to the example configuration file ( [daos\_agent.yml](https://github.com/daos-stack/daos/blob/master/utils/config/daos_agent.yml) ) for latest information and examples.

### Agent Startup

DAOS Agent is a standalone application to be run on each compute node. It can be configured to use secure communications (default) or can be allowed to communicate with the control plane over unencrypted channels. The following example shows daos\_agent being configured to operate in insecure mode due to incomplete integration of certificate support as of the 0.6 release and configured to use a non-default agent configuration file.

To start the DAOS Agent from the command line, run:

$ daos\_agent -i -o <'path to agent configuration file/daos\_agent.yml'> &

Alternatively, the DAOS Agent can be started as a systemd service. The DAOS Agent unit file is installed in the correct location when installing from RPMs. If you want to run the DAOS Agent without certificates (not recommended in production deployments), you need to add the -i option to the systemd ExecStart invocation (see below).

If you wish to use systemd with a development build, you must copy the service file from utils/systemd to /usr/lib/systemd/system. Once the file is copied modify the ExecStart line to point to your in tree daos\_agent binary.

ExecStart=/usr/bin/daos\_agent -i -o <'path to agent configuration file/daos\_agent.yml'>

Once the service file is installed, you can start daos\_agent with the following commands:

$ sudo systemctl daemon-reload  
$ sudo systemctl enable daos\_agent.service  
$ sudo systemctl start daos\_agent.service

To check the component status use:

$ sudo systemctl status daos\_agent.service

If DAOS Agent failed to start check the logs with:

$ sudo journalctl --unit daos\_agent.service

#### Disable Agent Cache (Optional)

In certain circumstances (e.g. for DAOS development or system evaluation), it may be desirable to disable the DAOS Agent’s caching mechanism in order to avoid stale system information being retained across reformats of a system. The DAOS Agent normally caches a map of rank->fabric URI lookups as well as client network configuration data in order to reduce the number of management RPCs required to start an application. When this information becomes stale, the Agent must be restarted in order to repopulate the cache with new information. Alternatively, the caching mechanism may be disabled, with the tradeoff that each application launch will invoke management RPCs in order to obtain system connection information.

To disable the DAOS Agent caching mechanism, set the following environment variable before starting the daos\_agent process:

DAOS\_AGENT\_DISABLE\_CACHE=true

If running from systemd, add the following to the daos\_agent service file in the [Service] section before reloading systemd and restarting the daos\_agent service:

Environment=DAOS\_AGENT\_DISABLE\_CACHE=true

## System Validation

To validate that the DAOS system is properly installed, the daos\_test suite can be executed. Ensure the DAOS Agent is configured before running daos\_test. If the agent is using a non-default path for the socket, then configure DAOS\_AGENT\_DRPC\_DIR in the client environment to point to this new location.

DAOS automatically configures a client with a compatible fabric provider, network interface, network domain, CaRT timeout, and CaRT context share address, that will allow it to connect to the DAOS system.

The client may not override the fabric provider or the CaRT context share address.

A client application may override the three remaining settings by configuring environment variables in the client’s shell prior to launch.

To manually configure the CaRT timeout, set CRT\_TIMEOUT such as:

export CRT\_TIMEOUT=5

To manually configure the network interface, set OFI\_INTERFACE such as:

export OFI\_INTERFACE=lo

When manually configuring an Infiniband device with a verbs provider, the network device domain is required. To manually configure the domain, set OFI\_DOMAIN such as:

export OFI\_DOMAIN=hfi1\_0

### Launch the client application

mpirun -np <num\_clients> --hostfile <hostfile> ./daos\_test

daos\_test requires at least 8GB of SCM (or DRAM with tmpfs) storage on each storage node.