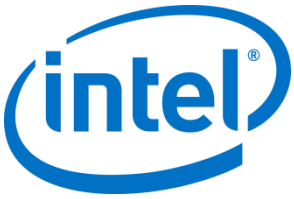


Intel® Cache Acceleration Software for Linux* v3.1

Administrator Guide

June 2016

Order Number: 328496-011US

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

No computer system can provide absolute security. Requires an enabled Intel® processor, enabled chipset, firmware and/or software optimized to use the technologies. Consult your system manufacturer and/or software vendor for more information.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at intel.com**.**

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

All products, computer systems, dates, and figures specified are preliminary based on current expectations, and are subject to change without notice.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Contents

1 Introduction

This guide offers the quickest way to install and begin using Intel® Cache Acceleration Software for Linux* v3.1. This guide assumes users have a basic knowledge of storage and application management, as well as basic Linux system administration.

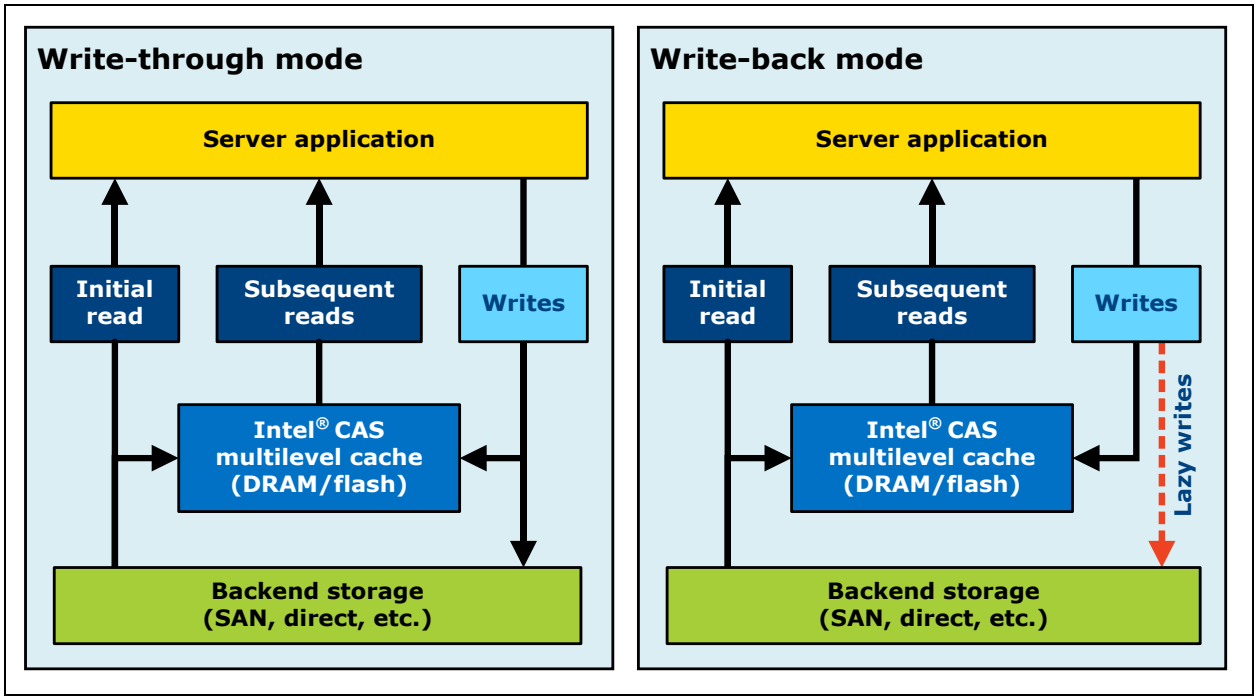
1.1 Intel CAS Overview

Intel CAS accelerates Linux applications by caching active (*hot*) data to a local flash device inside servers. Intel CAS implements caching at the server level, using local high-performance flash media as the cache drive media within the application server, thus reducing storage latency.

Intel CAS installs into the Linux operating system as a kernel module. The nature of the integration provides a cache solution that is transparent to users, applications, and your existing storage infrastructure. No storage migration or application changes are required.

As shown in Figure 1, initial read data is retrieved from backend storage and copied to the Intel CAS cache. A second read promotes data to system memory. Subsequent reads are returned at high-performance RAM or flash speed. In Write-through mode, all data is written synchronously to both the backend storage and the cache. In Write-back mode, all data is written to the cache then eventually flushed to the backend storage. When the cache is full, newly identified active data evicts stale data from the cache, utilizing the Intel CAS proprietary eviction algorithm.

Figure 1: Block Diagram (write-through and write-back modes)



Intel CAS for Linux, by default, employs a block-based caching architecture that caches all of the activity on the selected core device. A specific file or a list of multiple files (delimited by space) can be cached exclusively using the `-I` option or the *include.conf* config file (see Configuring Intel CAS starting on page 15).

1.2 New Features in this Release

This release contains the following new features:

- **NEW! In-flight Upgrade Capability** – Enables upgrade to future versions of CAS without the need to stop I/O or reboot. This maintainability feature allows your workload to continue running while you upgrade the system to the new version of CAS. With respect to Software Defined Storage solutions, such as Ceph, this feature eliminates the need to stop your storage devices when upgrading CAS (see Chapter 3.5).
- **Selectable Cache Line Size –** Enables selection of desired cache line size of 4k (default), 8k, 16k, 32k, or 64k, for reduced DRAM footprint requirement for in-memory metadata storage.
- **Automatic Partition Mapping** – CAS will automatically match the underlying core device partitioning scheme when accelerating a whole device. This feature delivers improved compatibility with Ceph deployment and activation scripts.
- **Continuous I/O during cache flushing** – CAS will continue processing I/O to the exported device (eg. intelcas1-1) during cache flush operations. (Previously, I/O was paused during flushing)
- **Improved device error handling –** CAS will continue serving I/O intelligently in the case of cache device I/O errors.
- **Bug fixes from v3.0.**

IMPORTANT: Note the following changes and update any user-created scripts accordingly:

- **--eviction-policy and --metadata-variant obsoleted from --start-cache command:** These options were previously used to reduce DRAM memory footprint needed for in-memory cache metadata. The new --cache-line-size capability now serves this purpose and replaces the obsoleted commands.

1.3 Documentation Conventions

The following conventions are used in this manual:

- `Courier font` - code examples, command line entries.
- *Italic* - user interface items, filenames, etc.
- **< > -** denotes mandatory commands
- **[ ] –** denotes optional commands

1.4 References

Refer to the resources and tools listed in Table 1 for assistance with Intel CAS testing and operations, or to learn more about caching and application I/O performance management.

Table 1: Reference Documents

Name	Location
FIO	http://freecode.com/projects/fio
Vdbench	http://vdbench.sourceforge.net/

1.5 Revision History

Revision	Description	Date
001B	Initial Beta release of document.	November 2013
001	Initial public release of document.	February 2013
002	Document updates for Linux v2.1	May 2013
003	Document updates for Linux v2.5	August 2013
004	Document updates for Linux v2.6	December 2013
005	Document updates for Linux v2.6.1	April 2014
006	Document updates for Linux v2.7	May 2014
007	Document updates for Linux v2.7 GA	July 2014
008	Document updates for Linux v2.8 GA	December 2014
009	Document updates for Linux v2.9 GA	June 2015
010	Document updates for Linux v3.0 GA	January 2016
011	Document updates for Linux v3.1 GA	May 2016

2 Product Specifications and System Requirements

2.1 Supported Operating Systems

Table 2 lists the platforms for 64-bit processors that Intel CAS supports.

Table 2: Supported Operating Systems

Operating System	Kernel
Red Hat* Enterprise Linux* (RHEL*) 6.6	x86_64, Kernel 2.6.32-504.3.3
Red Hat* Enterprise Linux* (RHEL*) 6.7	x86_64, Kernel 2.6.32-573
Red Hat* Enterprise Linux* (RHEL) 7.0	x86_64, Kernel 3.10.0-123.13.2
Red Hat* Enterprise Linux* (RHEL) 7.1	x86_64, Kernel 3.10.0-229
Red Hat* Enterprise Linux* (RHEL) 7.2	x86_64, Kernel 3.10.0-327
CentOS* 6.6	x86_64, Kernel 2.6.32-504.3.3
CentOS* 6.7	x86_64, Kernel 2.6.32-573
CentOS* 7.0	x86_64, Kernel 3.10.0-123.13.2
CentOS* 7.1	x86_64, Kernel 3.10.0-229
CentOS* 7.2	x86_64, Kernel 3.10.0-327
Oracle 6.6	x86_64, Kernel 2.6.32-504.3.3
Oracle 6.7	x86_64, Kernel 2.6.32-573
SUSE* Linux* Enterprise Server (SLES*) Version 11 SP4	x86_64, Kernel 3.0.101-63.1
Red Hat* Enterprise Linux* (RHEL*) 6.5 (custom)	x86_64, Kernel 3.10.0-123.4.4 (custom)
Ubuntu Server 14.04.3	x86_64, Kernel 3.19.0-39
Other distros – Intel CAS will install and compile from source on other distros and other kernels, but the user may be required to reproduce any issues on a validated distro & kernel to receive support.	Other kernels

Table 3 lists the hypervisor/OS combinations Intel Cache Acceleration Software for Linux v3.1 Administrator Guide supports.

Table 3: Supported Hypervisor/OS Combinations

Hypervisor	Supported Configuration	Notes
Xen*	Supported in hypervisor or guest.	Paravirtualized drivers are not supported.
KVM*	Supported in hypervisor or guest.	
VMWare*	Supported in guest.	

2.2 System Requirements

Table 4 lists system requirements for Intel CAS.

Table 4: Intel CAS System Requirements

Memory	RAM requirement is approximately 1GiB + 2% cache device capacity * 4KiB/<selected cache line size>. For example, when using 4KiB (default) cache line size, RAM requirement is approximately 1GiB + 2% cache device capacity.
CPU Overhead	Intel CAS consumes minimal CPU bandwidth (less than 10% in most cases). Ensure that Intel CAS has adequate CPU capacity to function optimally.
Flash/SSD	Any Linux flash device (SAS, SATA, PCIe*, Fibre Channel, RAID) is supported, and can be direct attached, expander attached, or attached via SAN (with a single worker). Additionally, the following Intel SSDs have been fully qualified: • Intel® SSD DC S3700 Series • Intel® SSD DC P3700 Series Cache device logical block size must be smaller than or equal to the logical block size of the core storage device. 60GB or larger capacity is recommended.
Storage	Primary storage devices: Any device that appears as a local block device. For example, local disk, RAID, SAN iSCSI (via Fibre Channel, Infiniband, Ethernet), etc. Core device (HDD) logical block size must be 512 B or larger. Operating system must be on separate partition or drive than the core storage device to be cached.
File Systems	The following file systems are supported for primary storage or core storage: • ext3 (limited to 16TiB volume size). This is an ext3 file system limitation. • ext4 • xfs **NOTE:** For best performance, core device file system block size should match the core device partition logical block size. **NOTE:** For best performance, I/O request sizes in multiples 4KiB are recommended.
Software	The following prerequisite software must be installed prior to Intel® CAS installation: • bzip2 • tar • sed • make • gcc • kernel-devel • kernel-headers The following software is optional. If desired, it must be installed prior to Intel® CAS installation: • dkms (http://linux.dell.com/dkms/)
Power Management	Power Management Suspend (S3) and Hibernate (S4) states must be disabled.

3 Installing CAS

This section describes how to perform a typical installation of Intel CAS. The installation package consists of loadable kernel modules that provide the caching layer and a management CLI that controls caching behavior.

3.1 Disabling Sleep States

Before configuring Intel CAS, you must disable Power Management Suspend (S3) and Hibernate (S4) states on your OS. Consult your OS documentation for how to accomplish this on your particular OS.

3.2 Configuring the Flash Device

Before configuring Intel CAS, you must have a flash device installed. The flash device can be any solid-state drive (SSD) or any PCIe flash drive supported by the Linux operating system (see Table 4 on page 10 for details).

Follow the instructions in the installation guide included with the flash device, and make note of the device's serial number or WWN so you can identify it once the system is up and running. All devices to be used for Intel CAS should be un-mounted and any auto-mount procedures disabled. When using a partition of a device as cache space, make sure the partition is aligned to the device's physical sector size for best performance.

For best performance, it is recommended to use the *noop* IO scheduler on the cache device (SSD). Initially, the intelcas virtual block device that is created when caching is enabled will inherit the IO scheduler of the primary storage device. If desired, the user can change the IO schedulers of the virtual block device and primary storage device independently after the virtual block device is created.

3.3 DKMS Installation

Intel CAS for Linux will automatically recompile from source via DKMS upon any kernel update when DKMS is installed prior to CAS installation and opted into during CAS installation.

The latest DKMS build can be obtained here: http://linux.dell.com/dkms/

Note: DKMS installation is optional and Intel CAS automatic recompile via DKMS is opt-in at the time of CAS installation.

Note: Using DKMS to automatically recompile Intel CAS is strongly recommended. If the user chooses not to install DKMS or not to opt-in for automatic recompile, then whenever a kernel update occurs, Intel CAS will be disabled until the user manually re-installs it.

3.4 Local Installation

Before you begin the installation, log on as an administrator or verify that you have the requisite privileges to install software. You must have "root" privileges or login as "root" in order to proceed.

Note: If Intel CAS 3.1 or newer is already installed, you can perform an in-flight upgrade to install the latest version without stopping I/O to cached devices (see Upgrading Intel CAS on page 13 for details).

Note: If Intel CAS 3.0 or older is already installed, you must first Uninstall the Software prior to Installation (see Uninstalling the Software on page 13 for details).

1. Download or copy the Intel CAS installer file to a directory on the target Intel CAS server. The installation instructions use the example of ~ or ~/ (equivalent of $HOME) on the server file system. The installer file name is in the format:

   `installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run`

   where: *XX.XX.XX.XXXXXXXXX* is the version information. For example:

   `installer-Intel-CAS-03.01.00.00000000.run`

   Note: Be sure to copy/transfer the installer files in binary format. The installer will not work if it is transferred in ASCII or non-binary format.

   Note: Installer automatically detects the correct distribution of Linux.

2. Navigate to the directory with the Intel CAS installation file.

3. Make the installer file executable:

   ```
   # chmod u+x installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

4. Launch the installation:

   ```
   #./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

   Note: For more information on the installer, type:

   ```
   #./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run --help
   ```

5. Read and accept the end-user license agreement (EULA) to proceed with the installation.

   Upon accepting the license agreement the following text should appear confirming the successful installation of Intel CAS:

```
Checking for validated GNU/Linux distribution              [  OK  ]
Checking for validated kernel                              [  OK  ]
Verifying installation dependencies...
- Looking for bzip2...                                     [  OK  ]
- Looking for tar...                                       [  OK  ]
- Looking for sed...                                       [  OK  ]
- Looking for gcc...                                       [  OK  ]
- Looking for make...                                      [  OK  ]
- Looking for bash...                                      [  OK  ]
- Looking for kernel-devel...                              [  OK  ]
- Looking for Linux kernel headers in
   /lib/modules/3.10.0-123.13.2.el7.x86_64/build/...       [  OK  ]
Building userspace management utility...                   [ DONE ]
Compiling kernel module...
[================================================]100%   [ DONE ]
Verifying build products...                                [ DONE ]
- Looking for dkms...                                      [  OK  ]
Intel(R) CAS installer has detected presence of DKMS on your GNU/Linux
operating system.

The installer can configure DKMS to recompile Intel(R) CAS after each Linux
kernel version upgrade. By accepting, you acknowledge that CAS may cease to
work properly if your new kernel version is not compatible with the software as
validated against supported configurations. By declining, you acknowledge that
Intel(R) CAS won't load after any Linux kernel upgrade until you manually
reinstall the software.
```

```
Do you want to configure DKMS for Intel(R) CAS? [y/N]
y
Installing casadm utility...                                    [ DONE ]
Setting up DKMS on a machine...                                 [ DONE ]
Installing DKMS module... (this may take up to few minutes)[ DONE ]
Configuring service...                                          [ DONE ]
Module intelcas loaded successfully!
Setting module autoload
Installation successful!
Starting intelcas (via systemctl):                              [  OK  ]
```

3.5 Upgrading Intel CAS

Intel CAS features in-flight upgrade capability. This capability allows the user to upgrade to a newer version of Intel CAS without the need to stop I/O to the cached devices and without the need for a system reboot. This maintainability feature is especially useful in software defined storage environments as the user does not have to stop the cluster in order to upgrade Intel CAS.

> ***Note:*** If Intel CAS 3.0 or older is already installed, you must first Uninstall the Software prior to Installation (see Uninstalling the Software on page 13 for details).

To perform an in-flight upgrade:

1. Make the installer file executable:

   ```
   # chmod u+x ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

2. Execute the Intel CAS installer with the --reinstall option:

   # ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run –-reinstall

3.6 Uninstalling the Software

Before uninstalling the software, it is important to first stop Intel CAS, as described in Stopping Intel CAS on page 20.

Also, close all applications, unmount all file systems, and remove any existing LVM volumes that use Intel CAS before uninstalling the software. The uninstallation process will not start if Intel CAS is in use.

To uninstall the software locally, navigate to the directory containing the Intel CAS installer for the version you are uninstalling. The following steps will uninstall the software: but will leave configuration information on the system for the new installation of Intel CAS.

3. Make the installer file executable:

   ```
   # chmod u+x ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

4. Launch the uninstaller:

   a. If you wish to leave the Intel CAS configuration files on the system:

      ```
      # ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run --uninstall
      ```

   b. If you wish to remove all Intel CAS files including the configuration files:

      ```
      # ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run --purge
      ```

   The following text should appear confirming that Intel CAS has been successfully uninstalled:

```
Checking Intel(R) CAS configuration [ DONE ]
Stopping intelcas service [ DONE ]
```

```
Cleaning up... [ DONE ]
Unloading kernel module... [ DONE ]
Removing DKMS configuration... [ DONE ]
Removing files... [ DONE ]
Running depmod... [ DONE ]
Uninstallation successful
```

3.7 Advanced Installation Options

The Intel CAS installer has the ability to perform automated installation across a distributed environment. For details on this type of installation, see Remote Installer Parameters on page 49.

4 Configuring Intel CAS

This chapter describes how to configure Intel CAS. Before you begin, it is important to understand the following restrictions:

- You must be logged on as root or have root privileges to start, stop, or configure Intel CAS.
- You cannot accelerate the partition that the operating system resides on.

Intel CAS provides a user-level utility called casadm to allow for simple configuration of the caching software. The utility is installed by default in the */sbin* directory.

Note: If a superuser promotes you to root, there is no guarantee that the /sbin directory will be in your $PATH environment variable. If casadm is not accessible, check this variable first. Use the command's full path (that is, /sbin/casadm).

If you launch the configuration utility in an Intel CAS shell by typing casadm –H, a list of command line options and arguments returns. For more detailed information on the different command line options, see Configuration Tool Details starting on page 37.

For Intel CAS configuration, note that the term "cache device" refers to the SSD/NVM device or RAM disk that is used for caching data from a slower device and the term "core device" refers to the slower device to be cached.

You may need to perform an *ls –l* on the */dev/disk/by-uuid* or */dev/disk/by-id* directories to verify the correct devices are being configured.

4.1 Using the Configuration Utility

The following instructions will detail how to configure Intel CAS with the *casadm* utility for typical deployments. For more details on the available commands for the *casadm* utility, see Configuration Tool Details starting on page 37.

The following is assumed for the subsequent instructions:

- The cache device (SSD) is */dev/sdc*. The cache device is either a raw block device or ram disk accessed as a block device. Ensure that the cache device does not have a file system and is not mounted.

  Note: Back up all data on your cache device before completing these steps as all data will be overwritten.

- The core device (primary storage) to be cached is */dev/sdb*.

  The core device may contain a filesystem (with or without data) or may be a raw block device. See System Requirements on page 10 for specific file system types and limitations for Intel CAS. Ensure that the device is not mounted.

- If necessary, perform an *ls –l* or *ll* on the */dev/disk/by-uuid* or */dev/disk/by-id* directory to ensure the correct devices are being configured.

- Core device (HDD) logical block size must be 512 B or larger.

- Cache device logical block size must be smaller than or equal to the logical block size of the core storage device.

- Ensure that both devices are removed from /etc/fstab and any other mechanism that auto mounts either the cache device or core device.

Verify that the Intel CAS module has loaded properly by using the following command:

```
# service intelcas status
```

If the Intel CAS module is not loaded, follow the installation instructions (starting on page 11). If the Intel CAS install fails, contact customer support.

4.2 Configuration for Write-through Mode

In write-through mode, the caching software writes data to the flash device and simultaneously writes the same data "through" to the core device (disk drives). Write-through ensures the core device is 100% in sync with the cache and its data is always available to other servers sharing that storage. However, this type of cache will accelerate only read intensive operations.

1. Ensure that the core device (*/dev/sdb*) is not mounted and that the cache device (*/dev/sdc*) is not mounted and contains no data to be saved. Entering the following command will display all mount points:

   ```
   # mount
   ```

2. Start a new cache with an ID of "1":

   ```
   # casadm –S –i 1 –d /dev/sdc
   ```

   You may notice a brief delay after entering the *casadm -S* command. Typically, this is less than 60 seconds, but can be longer.

   If the cache device is formatted or a file system already exists, you will need to use the "–f" force flag (for example, *casadm -S -d /dev/sdc -f*).

   Note: If the –f option is used; it will delete all information on the cache device. Please ensure all data has been backed up to another device (see Configuration Tool Details on page 37 for details).

3. Pair the core device to this new cache:

   ```
   # casadm –A –i 1 –d /dev/sdb
   ```

   The *add-core* command creates a new device in the */dev* directory with the following name format:

   `intelcas<cache ID>-<core #>` for example: */dev/intelcas1-1*.

   This new device can be treated as a regular block device.

4.3 Configuration for Write-back Mode

In write-back mode, the caching software writes the data first to the cache and acknowledges to the application that the write is completed, before the data is written to the core device. Periodically, those writes are "written back" to the disk opportunistically. While write-back caching will improve both write and read intensive operations, there is a risk of data loss if the cache device fails before the data is written to the core device.

Write-back mode is enabled when starting a new cache device with the option "*–c wb*":

```
# casadm –S –i 1 –d /dev/sdc –c wb
```

Pairing of a core device is similar to step 3 in the previous Configuration for Write-through Mode section.

4.4 Configuration for Write-around Mode

In write-around mode, the caching software writes data to the flash device if and only if that block already exists in the cache and simultaneously writes the same data "through" to the core device (disk drives). Write-around is

similar to write-through in that it ensures the core device is 100% in sync with the cache and in that this type of cache will accelerate only read intensive operations. However, write-around further optimizes the cache to avoid cache pollution in cases where data is written and not often subsequently re-read.

Write-around mode is enabled when starting a new cache device with the option "*–c wa*":

```
# casadm –S –i 1 –d /dev/sdc –c wa
```

Pairing of a core device is similar to step 3 in the Configuration for Write-through Mode section.

4.5 Configuration for Pass-through Mode

In pass-through mode, the caching software will bypass the cache for all operations. This allows the user to associate all their desired core devices to be cached prior to actually enabling caching. Once the core devices are associated, the user would dynamically switch to their desired caching mode (see page 39 for details).

Pass-through mode is enabled when starting a new cache device with the option "*–c pt*":

```
# casadm –S –i 1 –d /dev/sdc –c pt
```

Pairing of a core device is similar to step 3 in the Configuration for Write-through Mode section.

4.6 Automatic Partition Mapping

If the core device to be accelerated has existing partitions, selecting the parent device as the core device to be accelerated (eg. /dev/sdc as in the examples above) will accelerate all of the underlying partitions with a single command.

Intel CAS will automatically hide the existing core device partitions (eg. /dev/sdc1, /dev/sdc2, etc.) from the Operating System, create partitions on the exported device (eg. /dev/intelcas1-1p1, /dev/intelcas1-1p2, etc.), and make the exported device a child of the core device.

Table 6: Comparison of logical device layout before and after Intel CAS acceleration

Before	After
/dev/sdc /dev/sdc1 /dev/sdc2 /dev/sdc3 /dev/nvme0n1	/dev/sdc /dev/intelcas1-1 /dev/intelcas1-1p1 /dev/intelcas1-1p2 /dev/intelcas1-1p3 /dev/nvme0n1

This scheme ensures compatibility with existing object storage device creation and activation scripts from common software defined storage systems (eg. Ceph).

4.7 Mounting the CAS Device

After the setup for the cache-core pairs are complete, do the following:

1. Verify the setup with the casadm –list-caches command:

   ```
   # casadm –L
   ```

2. If the core device already had a file system in place, proceed to step 3. Otherwise, you must configure one in order to mount the device. Below are examples of using *mkfs* to establish a file system on an Intel CAS device:

   ```
   # mkfs -b 4096 -t ext3 /dev/intelcas1-1
   ```

   ```
   # mkfs.ext4 -b 4096 /dev/intelcas1-1
   ```

   ```
   # mkfs.xfs -f –i size=2048 –b size=4096 –s size=4096 /dev/intelcas1-1
   ```

3. Mount the new Intel CAS device:

   a. A directory is required to mount the new cache device. In this example, the mount point is */mnt/cache1*: Create the mount point directory if it does not exist.

      ```
      # mkdir -p /mnt/cache1
      ```

   b. Mount the directory:

      ```
      # mount /dev/intelcas1-1 /mnt/cache1
      ```

   Note: Intel CAS ensures that no changes are required to the application; it can use the same file system mount point (for example, */local/data*) as the core device previously used.

   Note: If your application uses the raw device directly (for example, some installations of Oracle*), then you must configure the application to use the exported device (eg. intelcas1-1) instead.

It might be necessary to change permissions for the mount point to allow applications access to the cache. The example below shows two commands that give user “bob” administrator privileges to the mount point:

```
# chown bob:adm /mnt/cache1 –R
```

or

```
# chmod  g+rw  /mnt/cache1
```

5 Running Intel CAS

This chapter describes how to perform typical administrative activities with Intel CAS.

5.1 Rebooting and Power Cycling

Intel CAS devices do not automatically become available for use after a reboot or power cycling of the system. It is necessary to recreate the cache pairing of the core storage device and the caching device. Use the following steps upon every system reboot or power cycling to continue to use Intel CAS cache devices.

1. Ensure that the Intel CAS kernel module has loaded properly with the following command:

   ```
   $ service intelcas status
   ```

   If the *intelcas* module is loaded, continue with step 2. If the module did not load properly, check that all the components of Intel CAS are installed. You can use the *--list* option on the install file, for example:

   ```
   # ./installer-Intel-CAS-XX.XX.XX.XXXXXXXX.run --list
   ```

   If any of the components are missing, you will have to uninstall and reinstall Intel CAS (see Installing CAS on page 11 for more details).

2. Ensure your devices are the same as before the reboot or power cycle of the system.

   Change directory to either */dev/disk/by-uuid* or */dev/disk/by-id*.

   Use the command *ls –l* or *ll* to view the association of physical devices and their identifiers.

   Verify that the identifiers for the device and/or partitions are associated with the correct physical devices. It is critical to ensure that the identifier for the caching SSD is known. Using the wrong device will lead to undetermined system behavior and the possible loss of data.

3. Start Intel CAS using the following command syntax:

   a. To start Intel CAS in write-through mode, enter the following commands to enable the cache pair:

      ```
      # casadm -S -i <cache_id> -d <cache_device>
      # casadm -A -i <cache_id> -d <core_device>
      ```

   b. To start Intel CAS in write-back mode, enter the following commands to enable the cache pair:

      ```
      # casadm -S -i <cache_id> -d <cache_device> -c wb
      # casadm -A -i <cache_id> -d <core_device>
      ```

   c. To start Intel CAS in write-around mode, enter the following commands to enable the cache pair:

      ```
      # casadm -S -i <cache_id> -d <cache_device> -c wa
      # casadm -A -i <cache_id> -d <core_device>
      ```

   d. To start Intel CAS in pass-through mode, enter the following commands to enable the cache pair:

      ```
      # casadm -S -i <cache_id> -d <cache_device> -c pt
      # casadm -A -i <cache_id> -d <core_device>
      ```

   Optionally, there is a way to start the cache maintaining the previous context prior to the last shutdown. This option will reduce the amount of time to warm the cache. This can be done by starting the cache device with the *--load* parameter. For example:

```
# casadm --start-cache --cache-id <cache_id> --cache-device <cache_device> /
      --load
```

or

```
# casadm -S -i <cache_id> -d <cache_device> -l
```

Caution: You must ensure that the last shutdown followed the instructions in the following section, Stopping Intel CAS. If there was any change in the core data prior to enabling the cache, data would be not synced correctly and will be corrupted.

4. Mount the Intel CAS device to continue normal use:

```
# mount /dev/intelcas1-1 /mnt/cache1
```

Your Intel CAS cache device is now ready for use.

5.2 Automatically Enabling Cache After Reboot

To start the Intel CAS software automatically on a reboot, add a script to run the necessary *casadm* commands followed by mounting of the *intelcasx-x* device as documented in Configuration for Write-through Mode on page 16. Consult the Intel CAS man pages and web wikis for details on autostarting applications.

The Intel CAS installation package includes an initialization script called *intelcas*, which creates all the cache devices defined in the config files in */etc/intelcas*. To manually execute this script, type the following:

```
# service intelcas start
```

Note: If you want to automatically start a cache and associate core devices, you will also need a config file to run with the above script. See Using Config and Setup Files (Advanced Config) on page 27 for details. This script can also be useful while testing the configuration files.

The following items should be considered:

- Intel CAS must be started *before* any applications that depend on the Intel CAS device or mount point.
- Depending on the size of the device, this may significantly increase the amount of time to boot. No progress indicators are displayed and it may appear that the system is in a hung or nonresponsive state.

5.3 Stopping Intel CAS

Prior to any shutdown or reboot of your system, it is recommended to cleanly stop Intel CAS.

To stop Intel CAS cleanly on shutdown, add a script to unmount the *intelcas* device followed by the *casadm -T* command.

Intel CAS must be stopped *after* any applications that may try to access the Intel CAS device. If the cache contains any dirty data, the data is automatically transferred to the disk during the cache shutdown procedure.

The following example uses *cache_id = 1*. To stop Intel CAS, log on as root and run:

```
# umount /mnt/cache1
# casadm -T -i 1
```

To double check that the cache was stopped successfully you can check that the cache no longer appears in the list:

```
# casadm -L
```

See Configuration Tool Details starting on page 37 for command-line options.

Note: For removing many-to-one associations, you can also use *casadm –T –i 1* to stop all cache-core pairs associated with <cache_id> = “1”.

Additionally, the Intel CAS installation package includes a script to remove all caches upon shutdown. To manually execute this script, type the following:

```
# service intelcas stop
```

5.4 Handling an Unplanned Shutdown

An unplanned shutdown is any time Intel CAS devices are not shutdown properly as described in the previous section. This can include power cycling the system, or a power loss to the system, without properly shutting down Intel CAS devices.

After an unplanned shutdown, there are two options for restarting caching. The first option is to start the cache in recovery mode, which will flush all dirty data that is currently in the cache, and then re-start caching with an empty cache. The second option is to reinitialize the cache with new metadata, which will clear the cache, resulting in the loss of all dirty data that is currently in the cache.

Caution: If the file system changes while caching is not running, data loss may occur. Do not issue any IO to the core device until caching has been restarted using one of the methods below.

5.4.1 Recovering the Cache

To start Intel CAS in with recovery enabled, enter the following commands:

```
# casadm –S -d /dev/sdc –r
# casadm –S -d /dev/sdc
# casadm –A –i 1 –d /dev/sdb
```

For more details, see ‘-S | --start-cache’ on page 37.

5.4.2 Reinitializing the Cache

To clear the cache (invalidate the entire cache), do the following:

1. Log on as root.
2. Start the cache including the `–f` parameter (and omitting the `–l` parameter), as shown below:

```
# casadm –S -d /dev/sdc –f
# casadm -A -i 1 -d /dev/sdb
```

This reinitializes the cache instead of loading the old state.

Reinitializing the cache with new metadata will destroy all write-cached data that has not yet been flushed to the disk.

Intel CAS maintains state across a reboot. If the system is rebooted without starting the caching subsystem, then the state of the cache data may become out of sync with the data in primary storage (core device). In this case, restarting the cache system without clearing the cache may result in data corruption.

5.5 Device I/O Error Handling

In the event of a read or write I/O error from either the cache device (SSD) or the core device (HDD), Intel CAS will intelligently handle the error and, if possible, will return the requested data and continue processing I/O in certain cases.

Cases where Intel CAS will ***return data*** and ***continue processing I/O*** include ***cache device I/O errors*** when attempting to:

- Read clean data from the cache
- Write data to the cache on a read operation (attempting to cache the data that was read)
- Write data to the cache in Write-Through or Write-Around modes on a write operation

Cases where Intel CAS will ***return an error*** to the calling application and ***stop processing I/O*** for any exported devices (eg. intelcas1-1) served by the cache SSD on which the I/O error occurred include ***cache device I/O*** errors when attempting to:

- Read dirty data from the cache
- Write data to the cache in Write-Back mode

Cases where Intel CAS will ***return an error*** to the calling application and ***continue processing I/O*** include ***core device I/O errors*** when attempting to:

- Read data from the core device
- Write data to the core device

6 IO Classification

6.1 IO Class Configuration

Intel CAS provides the ability to control caching of your data at a finer granularity than ever before. Intel CAS for Linux can analyze every IO on the fly to determine whether the requested block is filesystem metadata or data and, if it is data, the total size of file IO requested. Using this information the administrator can determine the best IO class configuration settings for the typical workload and can set which IO classes to cache and which not to cache, and set a priority level for each IO class. Then, when it becomes necessary to evict a cache line, the software will evict cache lines of the lowest available priority first (an improvement compared to traditional LRU eviction).

1. To enable IO classification and selective allocation, first look at the provided example IO class configuration file and edit to suit your needs:

   ```
   # vi /etc/intelcas/ioclass-config.csv
   ```

   The example IO class configuration file format is as follows:

Table 5: IO Class Configuration File Fields

Field	Description
IO class id	Unique ID for the IO class. Values in this field are fixed and should not be modified. (eg. ID for Metadata must always be 1)
IO class name	String name of the IO class. The user can use any string to represent IO class, however the meaning of the particular class will remain the same (e.g. IO class ID 1 will always represent Metadata regardless of the string assigned in this field)
Eviction priority	Priority number for the IO class. When eviction occurs, cache lines will be evicted from the lowest priority IO class available first.
Allocation	Boolean value that allows the user to decide whether data of this IO class will be cached or not. 0=do not cache, 1=cache.

```
IO class id,      IO class name,      Eviction priority,      Allocation
0,                Unclassified,       22,                     1
1,                Metadata,           0,                      1
11,               <=4KiB,             10,                     1
12,               <=16KiB,            11,                     1
13,               <=64KiB,            12,                     1
14,               <=256KiB,           13,                     1
15,               <=1MiB,             14,                     1
16,               <=4MiB,             15,                     1
17,               <=16MiB,            16,                     1
18,               <=64MiB,            17,                     1
19,               <=256MiB,           18,                     1
20,               <=1GiB,             19,                     1
21,               >1GiB,              20,                     1
22,               O_DIRECT,           21,                     1
23,               Misc,               22,                     1
```

2. After you have completed your changes to the IO class configuration and saved the file, you must load the configuration for your cache device with ID number represented by <ID> from file <FILE>:

   ```
   # casadm --io-class --load-config --cache-id <ID> -f <FILE>
   ```

3. Verify that the configuration file was successfully loaded:

   ```
   # casadm --io-class --list --cache-id <ID>
   ```

7 Advanced Options

7.1 Many-to-one Option

You can add multiple core devices to the single cache device.

1. Add more core devices to the existing caching framework (associated with `<cache_id> = "1"`) by entering the following:

   ```
   # casadm -A -i 1 -d /dev/sd<n>
   ```

   where *<n>* is any unique core drive and/or partition letter. (For example: *casadm –A -i 1 -d /dev/sdd*.)

2. Repeat the step 1 for each unique core drive to be added to the many-to-one caching framework.

   For each new core device added, a cache-core pair will be created with the following naming format:

   <u>/dev/intelcas1-1</u> will be created associated with *<cache_id> =1*, first core device.
   (For example: */dev/sdd*.)

   /dev/intelcas1-2 will be created associated with *<cache_id>=1*, second core device.
   (For example: */dev/sdf*)

   and so on.

3. Confirm the creation of each device with one of the following commands:

   ```
   # ls /dev/intelcas<cache_id>-<core_id>
   ```

   or

   ```
   # casadm -L
   ```

   or

   ```
   # casadm -P -i 1
   ```

4. (Optional) To cache a single file or a list of files, use the -I argument.

   ```
   # casadm -I -f /mnt/cache1/info.db
   ```

In some instances, like sequential read on a newly created cache device, the cache device may become overloaded, causing slower performance, and the queue may increase, consuming system buffers. To avoid this issue, it may be necessary to manually force an optimal queue size for the cache write queue specific to a use case. The default *max_writeback queue_size* value is 65,536. Another parameter may also be used to determine the unblock threshold for this queue: *writeback_queue_unblock_size* (default value is set at 60,000). You can use the following command to overwrite this default value and set a custom *max_writeback queue_size* or alter the *writeback_queue_unblock_size* threshold:

```
# modprobe intelcas max_writeback_queue_size=<size>
writeback_queue_unblock_size=<size>
```

or

```
# insmod intelcas max_writeback_queue_size=<size>
writeback_queue_unblock_size=<size>
```

Note: The above examples are each a single line.

Intel CAS is now enabled on the system and files to be accelerated can be written to */mnt/cache1*.

7.2 Multi-Level Caching

Intel CAS for Linux supports multi-level caching. For example, this enables the user to cache warm data from slow HDD media to faster SSD media, and then cache hot data from the SSD to an even faster media such as a RAMdisk. In this case, a fixed portion of DRAM will be allocated for buffering, and the SSD will remain as a fully inclusive cache so DRAM cache data will always be available on the SSD.

To set up a RAMdisk-based cache level, do the following:

1. Create a RAMdisk (40 MB or larger).

   In RHEL, for example, the default RAMdisk size is 8 MB so it must be configured to be equal or greater than 40 MB.

   a. For kernels prior to 3.0, modify *grub.conf* and change the *ramdisk_size* parameter to the desired capacity in KiB. Example highlighted below in **red**:

```
# grub.conf generated by anaconda
#
# Note that you do not have to rerun grub after making changes to this file
# NOTICE:  You have a /boot partition.  This means that
#          all kernel and initrd paths are relative to /boot/, eg.
#          root (hd0,0)
#          kernel /vmlinuz-version ro root=/dev/hda5
#          initrd /initrd-version.img
#boot=/dev/hda
default=0
timeout=10
splashimage=(hd0,0)/grub/splash.xpm.gz
title Red Hat Linux (2.4.20-20.9)
          root (hd0,0)
        kernel /vmlinuz-2.4.20-20.9 ro root=LABEL=/ hdc=ide-scsi
        ramdisk_size=40000
        initrd /initrd-2.4.20-20.9.img
```

   b. For kernels 3.0 or newer, issue the following command to increase the RAMdisk size to 40 MB:

```
# modprobe brd rd_size=40000
```

Note: Do not format or create a file system or mount the RAMdisk, at this time.

Note: Do not use */dev/ram0* since it is reserved by the kernel. Use any available RAMdisk */dev/ram1* or greater. We will use */dev/ram1* in this example.

2. Reboot the system and start a fresh Intel CAS install, if necessary.
3. Create the SSD caching framework, where */dev/sdc* is your SSD cache device:

```
# casadm -S -d /dev/sdc
```

4. Add a core device (*/dev/sdb*) mapping to the SSD caching framework:

   ```
   # casadm –A –i 1 –d /dev/sdb
   ```

5. Create the RAMdisk caching framework:

   ```
   # casadm –S –d /dev/ram1 –i 2
   ```

6. Add the tiered core device (*/dev/intelcas1-1*) mapping to the RAMdrive caching framework:

   ```
   # casadm –A –i 2 –d /dev/intelcas1-1
   ```

7. Create the file system from the newly created tiered cache drive:

   ```
   # mkfs -b 4096 –t ext3 /dev/intelcas2-1
   ```

   or

   ```
   # mkfs.ext4 -b 4096 /dev/intelcas2-1
   ```

   or

   ```
   # mkfs.xfs -f –i size=2048 –b size=4096 –s size=4096 /dev/intelcas2-1
   ```

8. Create a cache directory and mount the cache to it:

   ```
   # mkdir –p /mnt/cache1
   # mount /dev/intelcas2-1 /mnt/cache1
   ```

The newly created and mounted cache drive */mnt/cache1* is ready for use.

By default, any RAMdisk tiered buffering setup is volatile and will need to be set up again following a system restart. Any data should remain in the SSD following a clean restart. See Stopping Intel CAS on page 20 for details on stopping Intel CAS.

Caution: If the RAMdisk cache tier is started in write-back mode and there is a dirty shutdown, data loss may occur.

7.3 Linux LVM support

Intel CAS supports LVM in two ways.

1. LVM physical volumes can be created directly on Intel CAS devices. This allows creation of volume groups thereon, and later creation of logical volumes on those groups.

   NOTE: With older versions of LVM, you must first create a partition on the core device (eg. if the core device is /dev/sdc, you must create /dev/sdc1) prior to accelerating that device with Intel CAS device and creating the physical volume or creation of the logical volume will fail. If you see the following warning: *"WARNING: Ignoring duplicate config node: types (seeking types); Found duplicate PV"*, then you must use this workaround.

2. LVM logical volumes can be used as core devices, just like regular partitions.

LVM must be configured in the system before using it with Intel CAS, which is outside of the scope of this document.

Ensure Intel CAS devices are listed as acceptable block device types in */etc/lvm/lvm.conf* by adding the following line to the advanced settings:

```
# Advanced settings.
# List of pairs of additional acceptable block device types found
# in /proc/devices with maximum (non-zero) number of partitions.
types = [ "inteldisk", 16 ]
```

After the LVM is configured and a CAS device has been created (see Configuring Intel CAS starting on page 15 for further details) a physical volume can be created thereon:

```
# pvcreate /dev/intelcas1-1
```
(or `#pvcreate /dev/intelcas1-1p1` if creating the physical volume from a partition on the Intel CAS device)

After the physical volume exists, the user can create a volume group, for example: *vg_cas*, and then a logical volume on it, for example: *lv_cas*:

```
# vgcreate vg_cas /dev/intelcas1-1
# lvcreate -n lv_cas -l 100%FREE vg_cas
```

Create and mount LVM filesystem to "*vfs_cas*":

```
# mkfs.ext4 <-b 4096> /dev/vg_cas/lv_cas
# mkdir -p /mnt/vfs_cas
# mount /dev/vg_cas/lv_cas /mnt/vfs_cas
```

To remove the LVM device, use the following steps:

```
# umount /mnt/vfs_cas
# vgchange -an vg_cas
# casadm -R -i<cache_id> -j <core_id>
# casadm -T -i 1
```

Dynamic logical volume reduction or extension is not automatically supported (e.g. *vgreduce* or *vgextend*). The user must remove the CAS device (e.g. *intelcas1-1*) as the physical volume, perform the necessary size changes and then recreate the physical volume based on the CAS device. Consult Intel CAS LVM man pages and web wikis for details on how to use and manage LVMs.

7.4 Using Config and Setup Files (Advanced Config)

You can configure Intel CAS by using config and setup files. All the files must be stored in the */etc/intelcas* directory.

The */etc/intelcas* directory contains example files after a fresh installation; we recommend reviewing them before proceeding with the steps described in this section.

The following files are in the */etc/intelcas* directory:

- `intelcas.conf` – contains cache/core drive matching info.
- `include.conf` – contains lists of files included to be cached exclusively.

You can also create and use a script named *installation_setup_file* to automate remote installation and configuration using local config files.

7.4.1 Formats of Setup and Config Files

The *installation_setup_file* script has the following format and syntax:

```
# Comments begin with hash symbol
config: </etc/intelcas/intelcas.conf> </etc/intelcas/include.conf>
<IP Address of system A>
<IP Address of system B>
<IP Address of system N>

# Empty lines are ignored
```

Example of an *installation_setup_file*:

```
# INSTALL_SETUP_FILE
# Sample installation_setup_file for nodes 192.168.1.101 through
# 192.168.1.112

config: /etc/intelcas/intelcas.conf /etc/intelcas/include.conf
192.168.1.101
192.168.1.102
...
192.168.1.112
```

Each config file to be copied to the machines in a particular group must be listed in the `config:` line (delimited by space) for that group. All machines whose addresses are listed in a particular group will receive the config files listed in the group header. End of group is indicated with a new configuration group header or an end of file.

Format and syntax for *intelcas.conf* file:

```
# Make sure that devices listed below exist and correspond to proper roles.

# Cache configuration section
[caches]
# Cache ID   Cache device              Load     Mode             Extra flags
<cache_id>   /dev/disk/by-uuid/<uuid>  <yes,no> <WT,WB,WA,PT>    ioclass_file=<file>,cleaning_policy=<alru,nop>
#Core Device Configuration
 [cores]
# Cache ID   Core device              Extra flags
<cache_id>   /dev/disk/by-uuid/<uuid>  core_id=<core_id>
```

Here is an explanation of each of the fields:

- *<cache_id>* is a numeric value between 1 and 16,384 (valid cache instance numbers).
- Cache device and core device must point to an existing SSD device, ideally referenced by the UUID (as in Intel CAS command '`ls -l /dev/disk/by-uuid`'). Alternatively, devices can be referenced by device name (eg. /dev/sdb).
- Load takes the following values (case insensitive):
    - *"Yes"* – the metadata on cache device will be loaded upon start (i.e. will be read from the cache device - not recommended; risky in case of unplanned shutdown).
    - *"No"* – the metadata on cache device will be reinitialized upon start (i.e. will not be read from the cache device).
- Mode determines the cache mode, either write-through, write-back, write-around, or passthru.
- *Optional:* Extra flags allow extra settings for the cache and core devices and are comma separated.
    - *ioclass_file* allows the user to load a file that specifies an IO class policy for this cache
    - *cleaning_policy* allows the user to specify the cache cleaning policy to be used for this cache, either alru or nop.
    - *<core_id>* is a numeric value between 1 and 16,384 (valid core instance numbers)

Example *intelcas.conf*:

```
# Cache configuration section
[caches]
# Cache ID   Cache device Load     Mode     Extra flags
  1          /dev/sdc1    no       WT
  2          /dev/sdd     no       WB       ioclass_file=ioclass_config.csv

#Core device configuration
[cores]
# Cache ID   Core device  Extra flags
  1          /dev/sde1
  2          /dev/sdf1    core_id=7
```

Format and syntax for *include.conf* file:

```
# Intel® CAS file inclusion list

# List of files delimited by a new line
/mnt/cache/file1
/mnt/cache/file2
/mnt/storage/docs/doc1.txt
```

Example *include.conf*:

```
# Cache 1
# Files from core 1
/usr/local/phonebook.db
/usr/doc/database.db

# Files from core 2
/home/user/webpage.cgi-bin
/home/user/webpage.html
```

7.4.2 Installing with installation_setup_file

The remote installation utility (see Remote Installation on page 51) supports batch configuration on multiple target systems, by automatically copying (via *ssh* to multiple system IP addresses) previously created config files to appropriate remote target systems.

Note: The remote installation utility requires that the remote machine supports *ssh* connections and file transfer from the installation system/server.

The configuration procedure goes as follows:

Note: The same `installation_setup_file` is used for both installation and configuration (as well as for uninstallations).

Note: Installation and setup are two separate steps that can be performed independently of each other (i.e. a target system can be pre-configured even though there may be no existing installation of Intel CAS on it).

1. After you complete all the steps from Remote Installation on page 51, do the following to place configuration files on a single target system one at time:

   ```
   # ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
    --setup --one-machine <IP ADDRESS> </etc/intelcas/intelcas.conf>
   ```

   and/or

   ```
   # ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
    --setup --one-machine <IP ADDRESS> </etc/intelcas/include.conf>
   ```

2. Alternatively, you can implement the same two steps above by including them in to an *installation_setup_file* script and typing the following:

   ```
   # ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
    --setup --use-config <path>/installation_setup_file
   ```

Note: Ensure the files listed in group headers in the installation config file point to existing files in the filesystem, otherwise they will not be copied.

Note: After Intel CAS has been properly installed and configured on a system, ensure that the config files contain valid information before restarting that system.

Note: For your reference, your installation package includes example files named *<filename>.example* under the */etc/intelcas* directory (placed in that directory as part of the normal install process – no additional user steps are required) for each of the setup and config files referred to in this chapter. You may want to consult these files for further examples of correct syntax and formatting.

8 Monitoring Intel CAS

There are a number of performance counters available for viewing. These counters are accessible via the `casadm -P -i <cache_id>` command. This section contains a brief description of data included in the *stats* command line option.

```
# casadm --stats --cache-id <ID>
```

or

```
# casadm -P -i <ID>
```

See the '–P | --stats' section on page 41 for details.

The output of this command contains three tables that describe the activities in the caching system:

- **Usage statistics**
- **Request statistics**
- **Block statistics**

Entries are in the form: stat, actual number, percent of overall, units.

You can use these statistics to learn about your data usage. For example, looking at the sequential versus random read statistics reveals how your data is used.

The following tables list the statistics (counters) that Intel CAS records:

Table 7: Usage Statistics

Statistic	Description
Occupancy	Number of cached blocks
Free	Number of empty blocks in the cache
Clean	Number of clean blocks (cache data matches core data)
Dirty	Number of dirty blocks (block written to cache but not yet synced to core)

Table 8: Request Statistics

Statistic	Description
Read hits	Number of reads that were cache hits
Read partial misses	Number of reads that spanned both data that was in the cache and data that was not in the cache
Read full misses	Number of reads that were cache misses
Read total	Total number of reads
Write hits	Number of writes that were cache hits

Write partial misses	Number of writes that spanned both data that was in the cache and data that was not in the cache
Write full misses	Number of writes that were cache misses
Write total	Total number of writes
Pass-through reads	Number of read requests sent directly to the core device (not cached by CAS). This statistic is incremented when the request is handled while the cache is in pass-through mode (see –Q | --set-cache-mode for details).
Pass-through writes	Number of write requests sent directly to the core device (not cached by CAS). This statistic is incremented when the request is handled while the cache is in pass-through mode (see –Q | --set-cache-mode for details).
Serviced requests	Total number of IO requests serviced (that did not bypass the cache)
Total requests	Total number of IO requests (both serviced and bypassed requests)

Percentages are calculated as percentage of total requests. (e.g. Read Hits % = 100*(# read hits / total requests)).

Table 9: Block Statistics

Statistic	Description
Reads from core(s)	Number of blocks read from core device(s)
Writes to core(s)	Number of blocks written to core device(s)
Total to/from core(s)	Total number of blocks read from or written to core device(s)
Reads from cache	Number of blocks read from cache device
Writes to cache	Number of blocks written to cache device
Total to/from cache	Total number of blocks read from or written to cache device
Reads from exported object(s)	Number of blocks read from exported object(s) (eg. intelcas1-1)
Writes to exported object(s)	Number of blocks written to exported object(s)
Total to/from exported object(s)	Total number of blocks read from or written to exported object(s)

Table 10: Error Statistics

Statistic	Description
Cache read errors	Number of read errors to the cache device
Cache write errors	Number of write errors to the cache device
Cache total errors	Total number of read or write errors to the cache device

Core read errors	Number of read errors to the core device(s)
Core write errors	Number of write errors to the core device(s)
Core total errors	Total number of read or write errors to the core device(s)
Total errors	Total number of read or write errors to the cache or core devices

8.1 Viewing Cache Statistics

Usage example for cache-level statistics:

```
# casadm –P -i 1
```

Returned output:

```
Cache Id                 1
Cache Size               12055031 [4KiB Blocks] / 45.99 [GiB]
Cache Device             /dev/nvme0n1p1
Core Devices             3
Write Policy             wt
Eviction Policy          lru
Cleaning Policy          alru
Cache line size          4 [KiB]
Metadata Memory Footprint 639.1 [MiB]
Dirty for                0 [s] / Cache clean
Status                   Running
```

Usage statistics	Count	%	Units
Occupancy	12055001	100.0	4KiB blocks
Free	30	0.0	4KiB blocks
Clean	12055001	100.0	4KiB blocks
Dirty	0	0.0	4KiB blocks

Request statistics	Count	%	Units
Read hits	50532296	32.8	Requests
Read partial misses	202850	0.1	Requests
Read full misses	48537517	31.5	Requests
Read total	99272663	64.4	Requests
Write hits	31762847	20.6	Requests
Write partial misses	436	0.0	Requests
Write full misses	16197432	10.5	Requests
Write total	47960715	31.1	Requests
Pass-Through reads	2779876	1.8	Requests
Pass-Through writes	4027209	2.6	Requests
Serviced requests	147233378	95.6	Requests
Total requests	154040463	100.0	Requests

Block statistics	Count	%	Units
Reads from core(s)	471298381	57.4	4KiB blocks
Writes to core(s)	349624858	42.6	4KiB blocks

Total to/from core(s)	820923239	100.0	4KiB blocks
Reads from cache	140451806	16.5	4KiB blocks
Writes to cache	713000302	83.5	4KiB blocks
Total to/from cache	853452108	100.0	4KiB blocks
Reads from exported object(s)	611750187	63.6	4KiB blocks
Writes to exported object(s)	349624858	36.4	4KiB blocks
Total to/from exported object(s)	961375045	100.0	4KiB blocks

Error statistics	Count	%	Units
Cache read errors	0	0.0	Requests
Cache write errors	0	0.0	Requests
Cache total errors	0	0.0	Requests
Core read errors	0	0.0	Requests
Core write errors	0	0.0	Requests
Core total errors	0	0.0	Requests
Total errors	0	0.0	Requests

Usage example for core-level statistics:

```
# casadm –P -i 1 –j 1
```

Returned output:

```
Core Id          1
Core Device      /dev/sdd1
Exported Object /dev/intelcas1-1
Core Size        36620800 [4KiB Blocks] / 139.70 [GiB]
Dirty for        0 [s] / Cache clean
```

Usage statistics	Count	%	Units
Occupancy	4363950	36.2	4KiB blocks
Free	1	0.0	4KiB blocks
Clean	4363950	100.0	4KiB blocks
Dirty	0	0.0	4KiB blocks

Request statistics	Count	%	Units
Read hits	21075939	29.7	Requests
Read partial misses	77109	0.1	Requests
Read full misses	18448305	26.0	Requests
Read total	39601353	55.9	Requests
Write hits	17240919	24.3	Requests
Write partial misses	105	0.0	Requests
Write full misses	9972336	14.1	Requests
Write total	27213360	38.4	Requests
Pass-Through reads	983305	1.4	Requests
Pass-Through writes	3053282	4.3	Requests
Serviced requests	66814713	94.3	Requests
Total requests	70851300	100.0	Requests

Block statistics	Count	%	Units

Reads from core	156393665	57.0	4KiB blocks
Writes to core	118061392	43.0	4KiB blocks
Total to/from core	274455057	100.0	4KiB blocks
Reads from cache	49888641	17.7	4KiB blocks
Writes to cache	231251909	82.3	4KiB blocks
Total to/from cache	281140550	100.0	4KiB blocks
Reads from exported object	206282306	63.6	4KiB blocks
Writes to exported object	118061392	36.4	4KiB blocks
Total to/from exported object	324343698	100.0	4KiB blocks

Error statistics	Count	%	Units
Cache read errors	0	0.0	Requests
Cache write errors	0	0.0	Requests
Cache total errors	0	0.0	Requests
Core read errors	0	0.0	Requests
Core write errors	0	0.0	Requests
Core total errors	0	0.0	Requests
Total errors	0	0.0	Requests

Usage example for IO class-level statistics:

```
# casadm -P -i 1 -j 1 -d
```

Returned output:

```
IO class ID          1
IO class name        Metadata
Eviction priority    0
Selective allocation Yes
```

Usage statistics	Count	%	Units
Occupancy	864660	7.2	4KiB blocks
Free	0	0.0	4KiB blocks
Clean	864660	100.0	4KiB blocks
Dirty	0	0.0	4KiB blocks

Request statistics	Count	%	Units
Read hits	8097835	11.4	Requests
Read partial misses	0	0.0	Requests
Read full misses	17	0.0	Requests
Read total	8097852	11.4	Requests
Write hits	414587	0.6	Requests
Write partial misses	104	0.0	Requests
Write full misses	273056	0.4	Requests
Write total	687747	1.0	Requests
Pass-Through reads	2	0.0	Requests
Pass-Through writes	5517	0.0	Requests
Serviced requests	8785599	12.4	Requests
Total requests	8791118	12.4	Requests

Block statistics	Count	%	Units
Blocks reads	8097854	3.9	4KiB blocks
Blocks writes	9007260	7.6	4KiB blocks

Note: This command will output the above format for each defined IO class.

Usage example for IO class-level statistics output in csv format and saved to file

```
# casadm -P -i 1 -j 1 -d -o csv > stats.txt
```

Returned output:

```
IO class ID,IO class name,Eviction priority,Selective allocation,Occupancy [4KiB
blocks],Occupancy [%],Free [4KiB blocks],Free [%],Clean [4KiB blocks],Clean [%],Dirty [4KiB
blocks],Dirty [%],Read hits [Requests],Read hits [%],Read partial misses [Requests],Read
partial misses [%],Read full misses [Requests],Read full misses [%],Read total
[Requests],Read total [%],Write hits [Requests],Write hits [%],Write partial misses
[Requests],Write partial misses [%],Write full misses [Requests],Write full misses [%],Write
total [Requests],Write total [%],Pass-Through reads [Requests],Pass-Through reads [%],Pass-
Through writes [Requests],Pass-Through writes [%],Serviced requests [Requests],Serviced
requests [%],Total requests [Requests],Total requests [%],Blocks reads [4KiB blocks],Blocks
reads [%],Blocks writes [4KiB blocks],Blocks writes [%]
0,Unclassified,22,Yes,0,0.0,97,100.0,0,0.0,0,0.0,0,0.0,0,0.0,82,0.0,82,0.0,0,0.0,0,0.0,0,0.0,
0,0.0,1,0.0,0,0.0,82,0.0,83,0.0,83,0.0,0,0.0
1,Metadata,0,Yes,864660,7.2,0,0.0,864660,100.0,0,0.0,8068949,11.4,0,0.0,17,0.0,8068966,11.4,4
13037,0.6,104,0.0,273056,0.4,686197,1.0,2,0.0,5517,0.0,8755163,12.4,8760682,12.4,8068968,3.9,
8979669,7.6
```

Note: This command will output the above format for each defined IO class.

Note: Any of the previously referenced examples can be output to csv format as well.

8.2 Resetting the Performance Counters

The performance counters are automatically reset every time the cache is started. To manually clear the performance counters, use the `casadm -Z -i <cache_id> -j <core_id>` command line option.

9 Configuration Tool Details

The Intel CAS product includes a user-level configuration tool that provides complete control of the caching software. The commands and parameters available with this tool are detailed in this chapter.

To access help from the CLI, type the *–H* or *--help* parameter for details. You can also view the man page for this product by entering the following command:

```
# man casadm
```

9.1 -S | --start-cache

Usage: `casadm --start-cache --cache-device <DEVICE> [option...]`

Example:

```
# casadm --start-cache --cache-device /dev/sdc
```

or

```
# casadm -S -d /dev/sdc
```

Description: Prepares a block device to be used as device for caching other block devices. Typically the cache devices are SSDs or other NVM block devices or RAM disks. The process starts a framework for device mappings pertaining to a specific cache ID. The cache can be loaded with an old state when using the *–l* or *--load* parameter (previous cache metadata will not be marked as invalid) or with a new state as the default (previous cache metadata will be marked as invalid).

Required Parameters:

[-d, --cache-device <DEVICE>] : Caching device to be used. This is an SSD or any NVM block device or RAM disk shown in the */dev* directory. <device> needs to be the complete path describing the caching device to be used, for example `/dev/sdc`.

Optional Parameters:

***[-i, --cache-id <ID>]*:** Cache ID to create; <1 to 16384>. The ID may be specified or by default the command will use the lowest available number first.

[-l, --load]: Load existing cache metadata from caching device. If the cache device has been used previously and then disabled (like in a reboot) and it is determined that the data in the core device has not changed since the cache device was used, this option will allow continuing the use of the data in the cache device without the need to re-warm the cache with data.

> ***Caution:*** You must ensure that the last shutdown followed the instructions in Stopping Intel CAS on page 20 of this document. If there was any change in the core data prior to enabling the cache, data would be not synced correctly and will be corrupted.

***[-f, --force]*:** Forces creation of a cache even if a file system exists on the cache device. This is typically used for devices that have been previously utilized as a cache device.

> ***Caution:*** This will delete the file system and any existing data on the cache device.

[-c, --cache-mode <NAME>]: Sets the cache mode for a cache instance the first time it is started or created. The mode can be one of the following:

wt: (default mode) Turns write-through mode on. When using this parameter, the write-through feature is enabled which allows the acceleration of only read intensive operations.

wb: Turns write-back mode on. When using this parameter, the write-back feature is enabled which allows the acceleration of both read and write intensive operations.

Caution: A failure of the cache device may lead to the loss of data that has not yet been flushed to the core device.

wa: Turns write-around mode on. When using this parameter, the write-around feature is enabled which allows the acceleration of reads only. All write locations that do not already exist in the cache (i.e. the locations have not be read yet or have been evicted), are written directly to the core drive bypassing the cache. If the location being written already exists in cache, then both the cache and the core drive will be updated.

pt: Starts cache in pass-through mode. Caching is effectively disabled in this mode. This allows the user to associate all their desired core devices to be cached prior to actually enabling caching. Once the core devices are associated, the user would dynamically switch to their desired caching mode (see '–Q | --set-cache-mode' on page 39 for details).

[-r, --recovery-mode]: Flush dirty pages when starting after unclean shutdown (UNSAFE). In the event of a power loss while using a write-back cache, this option allows the flushing of any dirty data from the cache device to the core device.

Note: This command will only flush dirty data from the cache. It will not start the cache. After issuing the recovery command, you must then start the cache as normal.

Caution: It may be possible for data to be not flushed properly if the metadata has been corrupted due to the interruption. This may lead to data corruption or loss.

[-x, --cache-line-size <SIZE>]: Set cache line size {4 (default), 8, 16, 32, 64}. The cache line size can only be set when starting the cache and cannot be changed after cache is started.

9.2 -T | --stop-cache

Usage: `casadm --stop-cache --cache-id <ID> [option...]`

Example:

```
# casadm --stop-cache --cache-id 1
```
or
```
# casadm -T -i 1
```

Description: Stops all cache-core pairs associated with the cache device.

Required parameters:

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>.

Optional parameters:

[-n, --no-data-flush]: Do not flush dirty data on exit (UNSAFE). This parameter will not allow the flushing of dirty data from the cache device to the core device upon the stopping of the cache. This will significantly reduce the time needed to stop the cache to allow for activities such as a fast reboot. The core device should not be used until the cache is started again with the `--load` parameter. Then the Intel CAS device can be used as normal.

Caution: Data on the core device will not be complete or in sync with the cache device upon stopping the device. If the core device is used without starting Intel CAS cache it will lead to data corruption or data loss.

Note: The user may interrupt the blocking --stop-cache operation by pressing CTRL-C. When dirty data exists, interrupting the operation prior to the cache being fully stopped will result in the cache continuing to run. If the desire is to stop the cache without flushing the dirty data, use the --no-data-flush command.

9.3 –Q | --set-cache-mode

Usage: `casadm --set-cache-mode --cache-mode <NAME> --cache-id <ID> --flush-cache <yes/no>`

Example:

```
# casadm --set-cache-mode --cache-mode wb --cache-id 1 --flush-cache yes
```
or
```
# casadm -Q -c wb -i 1 -f yes
```

Description: Allows users to dynamically change cache modes while the cache is running.

Required Parameters:

[-c, --cache-mode <NAME>]:

- **wt –** switch from the current cache mode to write-through mode.
- **wb –** switch from the current cache mode to write-back mode.
- **wa –** switch from the current cache mode to write-around mode
- **pt –** switch from the current cache mode to pass-through mode.

Note: Dynamically switching to pass-through mode is useful in preventing cache pollution when the system is undergoing maintenance operations, for example.

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>

***[-f, --flush-cache]*: (required only when switching from write-back mode)**

Caution: You should carefully consider the following choices.

- **yes** - Flush cache contents immediately to core drive before switching to new cache mode.

  When choosing *yes* to flush the cache immediately, the operation may take a long time to complete depending on number of dirty blocks. I/O to the device will continue at reduced performance until flush completes.

- **no** – Begin transition to new cache mode immediately, but flush cache contents opportunistically.

  When choosing *no*, I/O to the device will continue at normal performance, but you must be aware that the cache will be in a transition state, and not yet in the newly chosen state until the cache is fully flushed. The transition to the new state will take longer than choosing the *yes* option. Current cache state and flush % can be checked using the casadm –L command.

9.4 –A | --add-core

Usage: `casadm --add-core --cache-id <ID> --core-device <DEVICE> [option...]`

Example:

```
# casadm --add-core --cache-id 1 --core-device /dev/sdb
```
or
```
# casadm -A -i 1 -d /dev/sdb
```

Description: Adds/maps a core device (either the full device or a partition) to the framework associated with a specified cache ID. This command can be repeated using the same *cache-id* number to map multiple cores to the same cache device.

Required Parameters:

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>.

***[-d, --core-device <DEVICE>]*:** Location of the HDD storage/core device.
You must use the complete device path in the /dev directory, for example `/dev/sdb`.

Optional Parameters:

[-j, --core-id <ID>]: Unique identifier for core <1 to 16384>.

9.5 –R | --remove-core

Usage: `casadm --remove-core --cache-id <ID> --core-id <ID>`

Example:

```
# casadm --remove-core --cache-id 1 --core-id 1
```
or
```
# casadm -R -i 1 -j 1
```

Description: Deletes the cache/core device mapping, which is one way to disable caching of a device.

Required Parameters:

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>

***[-j, --core-id <ID>]*:** Unique identifier for core <1 to 4096>.
You can identify the assigned value for a particular core device using the *casadm -L* command.

Caution: Before using *casadm -R*, stop all I/O to the mapped core device, ensure it is not in use, and unmount it.

Note: You can interrupt the blocking --remove-core operation by pressing CTRL-C. When dirty data exists, interrupting the operation prior to the core being fully removed will result in the core continuing to be cached.

9.6 –L | --list-caches

Usage: `casadm --list-caches`

Example:

```
# casadm --list-caches
```
or
```
# casadm -L
```

Description: Provides a list of every cache instance currently running on the system. It also lists:

- Flash/SSD device used in the instance

- Storage device used in the instance
- WritePolicy of the instance (write-through by default)
- Status of the instance

Example output:

```
type      id   disk              status            write policy   device
cache     1    /dev/sda1         Running           wb             -
+core     1    /dev/sdc1         -                 -              /dev/intelcas1-1
```

9.7 –P | --stats

Usage: `casadm --stats --cache-id <ID> [option...]`

Example:

```
# casadm --stats --cache-id 1
```
or
```
# casadm -P -i 1
```

Description: Prints performance and status counters for a specific cache instance. **Viewing Cache Statistics** shows the detailed output.

Required Parameters:

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>.

Optional Parameters:

[-j, --core-id <ID>]: Unique identifier for core <1 to 16384>. Display statistics for a specific core device.

[-d, --io-class-id <ID>]: Unique identifier for io class <0 to 23>. Display statistics for a specific IO class.

Note: <ID> is optional. When the --io-class-id parameter is specified without specifying an <ID>, statistics will be displayed for each individual IO class.

[-f, --filter <filter-spec>]: Comma separated list of filters (eg. --filter=conf, req). Filter statistics output to only the requested statistics.

- ***all:*** (default mode) Displays all available cache statistics.
- ***conf:*** Displays cache and core configuration information and dirty timestamp.
- ***usage:*** Displays statistics on occupancy, free, clean, and dirty.
- ***req:*** Displays IO request level statistics.
- ***blk:*** Displays block level statistics.
- ***err:*** Displays I/O error statistics.

[-o, --output-format <format>]: Sets desired output format for statistics.

- ***table:*** (default mode) Displays a table of the statistics information.
- ***csv:*** Outputs a comma separated list of statistics information. This output can be piped to a file and easily parsed or opened in a spreadsheet editor.

9.8 –Z | --reset-counters

Usage: `casadm --reset-counters --cache-id <ID> --core-id <ID>`

Example:

```
# casadm --reset-counters --cache-id 1 --core-id 1
```
or
```
# casadm -Z -i 1 -j 1
```

Description: Resets performance and status counters for a specific cache/core pair.

Required Parameters:

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>.

***[-j, --core-id <ID>]*:** Unique identifier for core <1 to 4096>. You can identify the assigned value for a particular core device using the *casadm -L* command.

9.9 –I | --include-files

Usage: `casadm --include-files --files <FILE ...>`

Example:

```
# casadm --include-files --files /root/samplefile /usr/bob/anotherfile ...
```
or
```
# casadm -I -f /root/samplefile /usr/bob/anotherfile ...
```

Description: Adds a *single file* or a *list of files* into the inclusion table of the cache instance. By default, Intel CAS software caches everything unless `--include-files` option lists specific files to be included.

After `--include-files` option is used, ONLY the listed files will be included; all other files will not be cached. Multiple file names are delimited by a space. Any file included is translated by the software into a list of static address ranges (Software max limit is 65536 total address ranges). If the original file grows or shrinks beyond that range, or if the file moves to a different location, the kernel module is NOT aware of this and will not dynamically adjust. Thus, the additional data will NOT be cached. Therefore, if using this option you should refresh the mappings periodically.

Filenames and paths may be specified as absolute paths, relative paths, or symbolic links. It should be noted that only wildcard characters parsable by the user's specific shell are supported. Consult your Linux shell documentation/man page for specific wildcard characters supported.

- Once entered, the included list of files cannot be viewed or retrieved by the system.
- Once an include file or multiple files are added, it is not possible to 'exclude' them except to overwrite with a new include file list command or to execute the `--reset_files` command (see below).
- Each subsequent execution of `casadm` with the `-I` argument resets the existing list and any older instance is overwritten with the new list.
- Duplicate, invalid, missing, or nonexistent file names are not allowed.

Required Parameters:

[-f, --files <FILE ...>]: Path(s) to a file or multiple files to be included.

Optional parameters:

[-v, --verbose]: Sets verbose output. Intel CAS will output a list of LBA ranges that will be cached based on the file list in the selected include file.

9.10 –X | --reset-files

Usage: `casadm --reset-files --cache-id <ID> --core-id <ID>`

Example:

```
# casadm --reset-files --cache-id 1 --core-id 1
```
or
```
# casadm -X -i 1 -j 1
```

Description: Resets the include list associated with a specific cache-core pair (identified by *<cache_ID>* and *<core_ID>*). Is also used in resetting included files in specific cache-core pairs under many-to-one mapping.

Required Parameters:

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>

***[-j, --core-id <ID>]*:** Unique identifier for core <1 to 4096>

9.11 –F | --flush-cache

Usage: `casadm --flush-cache --cache-id <ID>`

Example:

```
# casadm --flush-cache --cache-id 1
```
or
```
# casadm -F -i 1
```

Description: Flushes all dirty data from the cache device to all the associated core devices.

Required Parameters:

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>.

Note: You can interrupt the blocking --flush-cache operation by pressing CTRL-C. When dirty data exists, interrupting the operation prior to the cache being fully flushed will result in some dirty data remaining in the cache. The dirty data will be flushed opportunistically as normal. I/O to the device will continue with reduced performance during cache flushing.

9.12 –E | --flush-core

Usage: `casadm --flush-core --cache-id <ID> --core-id <ID> [option...]`

Example:

```
# casadm --flush-core --cache-id 1 --core-id 2
```
or
```
# casadm -E -i 1 -j 2
```

Description: Flushes all dirty data from the specified cache device to the specified associated core device.

Required parameters:

***[-i, --cache-id <ID>]*:** Unique identifier for cache <1 to 16384>.

***[-j, --core-id <ID>]*:** Unique identifier for core <1 to 4096>.

Note: You can interrupt the blocking --flush-core operation by pressing CTRL-C. When dirty data exists, interrupting the operation prior to the cache being fully flushed will result in some dirty data

remaining in the cache. The dirty data will be flushed opportunistically as normal. I/O to the device will continue with reduced performance during cache flushing.

9.13 –D | --flush-parameters

Usage: `casadm --flush-parameters --cache-id <ID> --cleaning-policy-type <NAME> [option...]`

Example:

```
# casadm --flush-parameters --cache-id 1 --cleaning-policy-type alru --wake-up 20 --staleness-time 120 --flush-max-buffers 100 --activity-threshold 10000
```
or
```
# casadm -D -i 1 -c alru -w 20 -s 120 -b 100 -t 10000
```

Description: This command allows the customization of various behaviors for flushing of dirty data.

Caution: A deep understanding of caching concepts must be understood before using this command. A change from the default settings could have a negative effect on caching performance.

Required Parameters:

[-i, --cache-id <ID>]: Unique identifier for cache <1 to 16384>.

[-c, --cleaning-policy-type <NAME>]: This parameter specifies the flushing policy to be utilized for the cache specified. The default is `alru` which is a modified least recently used method that will flush dirty data periodically. The only other option is `nop` which simply disables cache flushing except as needed for cache line replacement.

Optional Parameters:

[-w, --wake-up <NUM>]: This is the wake up period for the flushing thread, in seconds. The default is 20 seconds.

[-s, --staleness-time <NUM>]: The amount of time, in seconds, that must pass after the last write operation before a block in the cache can be scheduled to be flushed. The default is 120 seconds.

[-b, --flush-max-buffers <NUM>]: The maximum number of dirty cache blocks that will be flushed in one flush cycle. The default is 100.

[-t, --activity-threshold <NUM>]: The amount of time, in milliseconds, that has to pass from the last I/O operation before the cleaning thread can start. Default is 10000 ms.

9.14 –H | --help

Usage: `casadm --help` or `casadm --<command> --help`

Examples:

```
# casadm -help
```
or
```
# casadm -H

# casadm --start-cache --help
```
or
```
# casadm -S -H
```

Description: Displays a list of *casadm* commands along with a brief description. Use this command to also get more information on specific commands.

9.15 –V | --version

Usage: `casadm --version`

Example:

```
# casadm --version
```
or
```
# casadm -V
```

Description: Reports the Intel CAS kernel module and command line utility version numbers.

Optional parameters:

[-o, --output-format <format>]: Sets desired output format of the IO class configuration.

table: (default mode) Displays a table of the IO class configuration.

csv: Outputs a comma separated list of the IO class configuration. This output can be piped to a file and easily parsed or opened in a spreadsheet editor.

9.16 –C | --io-class

9.16.1 –C | --load-config

Usage: `casadm --io-class --load-config --cache-id <ID> --file <file_path>`

Example:

```
# casadm --io-class --load-config --cache-id 1 --file /etc/intelcas/ioclass-config.csv
```
or
```
# casadm -C -C -i 1 -f /etc/intelcas/ioclass-config.csv
```

Description: Loads IO class configuration settings for the selected cache.

Required parameters:

[-i, --cache-id <ID>]: Unique identifier for cache <1 to 16384>.

[-f,--file]: Specifies the IO class configuration csv file to load.

9.16.2 –L | --list

Usage: `casadm --io-class --list --cache-id <ID>`

Example:

```
# casadm --io-class --list --cache-id 1 --file /etc/intelcas/my-config.csv
```
or
```
# casadm -C -L -i 1 -f /etc/intelcas/ioclass-config.csv
```

Description: Displays the current IO class configuration settings for the specified cache ID. Optionally exports those settings to a file.

Required parameters:

[-i, --cache-id <ID>]: Unique identifier for cache <1 to 16384>.

Optional parameters:

[-o, --output-format <format>]: Sets desired output format of the IO class configuration.

table: (default mode) Displays a table of the IO class configuration.

csv: Outputs a comma separated list of the IO class configuration. This output can be piped to a file and easily parsed or opened in a spreadsheet editor.

10 Installer Parameters

The Intel CAS installer and remote installer have several options that can make installation more convenient. The different options are detailed below.

10.1 -al | --accept-license

It is required to accept the user license prior to installing Intel CAS. This option is for the automatic acceptance of the license which requires no further interaction to complete installation. This option can be useful for scripting the installation of Intel CAS.

By using this flag you state that you have read and accept the license!

10.2 -am | --accept-unsupported-module

Auto-accepts setting the allow_unsupported_modules flag.

This parameter updates the "unsupported module flag" in `etc/modprobe.d/unsupported-modules` on SLES 11 and newer environments. Once the flag is set, it will remain so and will apply to any other modules. Having this flag set to 1 is critical for starting Intel CAS in the SLES 11 environment.

This option has no affect if installing on non-SLES 11 systems.

By using this flag you state that you accept setting this flag by the installation script.

10.3 -ad | --accept-dkms

This flag indicates that the administrator has installed DKMS and would like Intel CAS software installed with DKMS support. This enables the automatic rebuild of the Intel CAS kernel module upon the each OS kernel version update.

10.4 -rd | --reject-dkms

This flag indicates that the administrator wants Intel CAS software installed without DKMS support.

NOTE: If this flag is specified, the administrator will be required to manually re-install Intel CAS after any OS kernel version update.

10.5 -as | --auto-start

This option will force automatic Intel CAS service startup once installation has completed if CAS configuration files are found.

Caution: By running with this option CAS might overwrite hard drives which are mentioned as caching devices in the config file.

10.6 -d | --display-license

This option displays the end-user license agreement without launching the installer.

10.7 -p | --purge

Uninstalls Intel CAS and removes all the configuration files stored in */etc/intelcas*. This would be used for the complete removal of Intel CAS.

10.8 –f | --force

Setting this flag will ignore all system checks for Intel CAS installation. This will include the –am flag setting as well.

10.9 -h | --help

This option displays a listing of the installer options along with a short description.

10.10 -l | --list

This option displays list of files installed by the package. It is useful in ensuring that all the files in the package are installed properly.

10.11 –t | --try-run

Setting this flag will only run and report the system checks performed during a normal installation.

10.12 -r | --reinstall

Perform an in-flight upgrade to a new version of Intel CAS (or in-flight reinstall of the existing version) without stopping I/O to cached devices.

10.13 -u | --uninstall

Uninstalls files for Intel CAS, but retains configuration for future use. Typically used when uninstalling an old version of Intel CAS to install a newer version.

11 Remote Installer Parameters

The Intel CAS remote installer has several options that can make installation more convenient. The different options are detailed below.

11.1 -al | --accept-license

It is required to accept the user license prior to installing Intel CAS. This option is for the automatic acceptance of the license which requires no further interaction to complete installation. This option can be useful for scripting the installation of Intel CAS.

By using this flag you state that you have read and accept the license.

11.2 -am | --accept-unsupported-module

Auto-accepts setting the allow_unsupported_modules flag.

This parameter updates the "unsupported module flag" in `etc/modprobe.d/unsupported-modules` on SLES 11 and newer environments. Once the flag is set, it will remain so and will apply to any other modules. Having this flag set to 1 is critical for starting Intel CAS in the SLES 11 environment.

This option has no affect if installing on non-SLES 11 systems.

By using this flag you state that you accept that the installer will set this flag on all remote systems to which you are installing Intel CAS.

11.3 -ad | --accept-dkms

This flag indicates that the administrator has installed DKMS and would like Intel CAS software installed with DKMS support. This enables the automatic rebuild of the Intel CAS kernel module upon the each OS kernel version update.

11.4 -rd | --reject-dkms

This flag indicates that the administrator wants Intel CAS software installed without DKMS support.

Note: If this flag is specified, the administrator will be required to manually re-install Intel CAS after any OS kernel version update.

11.5 -as | --auto-start

This option will force automatic Intel CAS service startup once installation has completed if CAS configuration files are found.

Caution: By running with this option Intel CAS will automatically overwrite any data on devices which are selected as cache devices in the config file.

11.6 -d | --display-license

This option displays the end-user license agreement without launching the installer.

11.7 -f | --force

Setting this flag will ignore all system checks for Intel CAS installation. This will include the –am flag setting as well.

11.8 -h | --help

This option displays a listing of the installer options along with a short description of each.

11.9 -r | --reinstall

Perform an in-flight upgrade to a new version of Intel CAS (or in-flight reinstall of the existing version) without stopping I/O to cached devices.

11.10 -u | --uninstall

Uninstalls files for Intel CAS, but retains configuration for future use. Typically used when uninstalling an old version of Intel CAS to install a newer version.

11.11 -1 | --one-machine

Performs remote installation on a single machine.

11.12 -c | --use-config

Specifies the path to the remote installer config file.

11.13 -s | --setup

Specifies the path to a single CAS config file to be uploaded to remote machine

11.14 -S | --use-sudo

Allows non-root user to run installer and performs remote tasks using sudo.

Note: This method is not recommended. The preferred method is to run the remote installer as root user.

Note: When installing on RHEL 7.x or CentOS 7.x, you must remove the following directive from the sudoers file in order for --use-sudo to work: `Defaults requiretty`.

12 Advanced Installation Methods

12.1 Remote Installation

This step is an alternative to the installation method described in Local Installation on page 11; however, the same assumptions apply as the ones described in notes at the beginning of that section.

To execute a remote installation, you must define two separate entities:

- *Installation server* (or system) – machine (or virtual machine) on which the installation utilities are launched.
- *Installation clients* (or target systems) – machines on which Intel CAS is to be installed and/or configured.

The remote installation scripts can be executed in one of three ways:

1. With passwordless *ssh* logins on all of the installation clients on which you are installing Intel CAS.

   In this case, the user will not be prompted for the root password during installation. To mitigate this security risk, using *ssh-keygen* to make a secured key pairing (with public and private keys) between both the installation server and each installation client is highly recommended. Details of how to implement *ssh-keygen* is beyond the scope of this document (this information is freely available on the internet), however a short generic example of how to generate and distribute *ssh* keys generated by this method is provided in Appendix C. *This is the recommended remote installation method.*

   Note: If using ssh-keygen method of installation, before the remote installation starts, ensure that you can *ssh* from the installation server/system to all the installation clients as root without being prompted for a password.

2. Without passwordless *ssh* logins, but with the *expect* utility program installed on the installation server.

   In this case the Intel CAS remote installer will prompt the user for the root password once per installation client.

3. Without passwordless *ssh* logins and without the *expect* utility program installed.

   In this case, the Intel CAS remote installer will prompt the user for the root password each time an activity is performed on an installation client (approximately 4 times per installation client machine).

To begin remote installation:

1. Download or copy the Intel CAS remote installer file to your home directory on the target Linux server or VM guest.

   The installation instructions use the example of `~` (equivalent of `$HOME`) on the server file system. The installer file name is in the format: `installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run` where *XX.XX.XX.XXXXXXXX* is the version information.

2. Make the installer file executable:

   ```
   # chmod u+x ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
   ```

There are two options available for remote installation:

- Install to a single remote machine.
- Install to multiple remote machines.

Note: For more information about the remote installer, you can access help by typing:

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run --help
```

12.1.1 Installation to a Single Remote Machine

To install Intel CAS on one remote host, do the following:

1. Type the following:

   ```
   # ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
    --one-machine <IP ADDRESS>
   ```

2. Read and accept the end-user license agreement (EULA) to proceed with the installation.
3. To confirm that remote installation was successful, *ssh* to the remote machine and at the Intel CAS shell, type the following:

   ```
   # casadm -H
   ```

   If the help command is successful and you are able see the help listing, your remote installation of Intel CAS has been successful. Additionally, you will find an installation log file in the same directory as the install after the installation is complete.

12.1.2 Batch Installation to Multiple Remote Machines

Installing to multiple remote machines requires an `installation_setup_file` (see Using Config and Setup Files (Advanced Config) for detailed information on setup and config file formats and syntax).

1. After your `installation_setup_file` is ready, type the following command to start the batch remote installation process:

   ```
   # ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
    --use-config installation_setup_file
   ```

   Note: If a current or earlier version of Intel CAS already exists on a remote host, the installer will automatically detect that and skip to the next system.

   Note: Paths to the config files cannot be relative; you must specify an absolute path.

2. Read and accept the end-user license agreement (EULA) to proceed with the installation.
3. Confirm that remote batch installation was successful.

   ssh to each of the remote machines and at the Linux shell, type the following:

   ```
   # casadm -H
   ```

   If the help command is successful and you are able see the help listing, your remote installation of Intel CAS has been successful. Additionally, you will find an installation log file in the same directory as the install after the installation is complete.

12.1.3 Remote Uninstallation

To uninstall the software remotely, do the prerequisites steps described in Uninstalling the Software on page 13, then do the following:

To uninstall from a *single* remote host, type the following:

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
--uninstall --one-machine <IP ADDRESS>
```

To batch uninstall from *multiple* remote hosts, type the following:

```
# ./installer-remote-Intel-CAS-XX.XX.XX.XXXXXXXX.run
--uninstall --use-config installation_setup_file
```

Note: The *installation_setup_file* mentioned in the batch uninstall is the same file used to install the software.

13 Terminology

Term	Definition
cache	The transparent storage of data so that future requests for that data can be served faster.
cache hit	When requested data is contained in (and returned from) the cache.
cache miss	When requested data is not in the cache, and therefore must be retrieved from its primary storage location.
core device	The device to be cached.
dirty data	This refers to data that is modified within the cache but not modified in main memory.
guest	An operating system running on a Virtual Machine (VM) environment.
host	The operating system running (hosting) the Virtual Machine (VM) environment, on which guest Operating Systems can be installed.
hypervisor	A hardware virtualization technique that allows multiple operating systems, termed guests, to run concurrently on a host computer.
I/O	Abbreviation for input/output as it relates to the flow of data.
lazy write	The process of mirroring to primary storage. See also *write-back*.
NAS	Network-attached storage. File-level data storage (such as fixed disk and magnetic tape drives) that are directly linked to a storage area network or other network.
pass-through	A caching mode in which the cache will be bypassed for all operations.
primary storage	As it relates to caching, the storage system or location (DAS, SAN, NAS, etc.) where the data is stored.
SAN	Storage Area Network. Framework used to attach remote computer storage devices to servers. Storage devices appear as if they were attached locally to the operating system.
SSD (solid-state disk)	A device used for data storage that uses memory chips instead of a revolving disk.
tiered storage	A data storage technique that moves data between two or more kinds of storage, which are differentiated by four primary attributes: price, performance, capacity, and function.
write-around	A caching mode in which some write operations are not cached. Writes to blocks that do not exist in cache are written directly to the core device, bypassing the cache. If a write operation is issued to a block that is already in cache (because of a previous read operation), then writes are sent to both the core device the cache device. Write-Around cache improves performance of workloads where write operations are done rarely and no further read accesses to that data are performed, so there is no benefit in caching it.
write-back	A caching mode in which data is written first to the cache and then mirrored to primary storage when I/O bandwidth is available. The process of mirroring to primary storage is known as a *lazy write*.
write-through	A caching mode in which every write to the cache causes a synchronous write to primary storage.

A. Frequently Asked Questions

This appendix provides contact information and answers to frequently asked questions.

How do I contact technical support?

Contact technical support by phone at 800-404-2284 or at the following URL: http://www.intel.com/support/go/cas.

Why does Intel CAS for Intel CAS use some DRAM space?

Intel CAS for Linux uses a small amount of system memory for metadata, which tells us which data is in the SSD, which is in the HDD. The amount of memory needed is proportional to the size of the cache space. This is true for any caching software solution.

Does Intel CAS for Intel CAS work with non-Intel® SSDs?

Yes. Intel CAS for Linux will work with any SSD but we validate only on Intel SSDs. Additionally, Intel CAS is favorably priced when purchased with Intel® SSDs.

How do I test performance?

In addition to the statistics provided (see Monitoring Intel CAS for details), third-party tools are available that can help you test I/O performance on your applications and system, including:

FIO (*http://freecode.com/projects/fio*)

dt (*http://www.scsifaq.org/RMiller_Tools/dt.html*) for disk access simulations

Is it possible to experience slower than HDD performance when using caching?

Yes, it is possible. For example, if the cache is in write-back mode and the entire cache is full of dirty data and a read occurs which requires new blocks to be loaded into the cache, performance will be degraded even if the read is sequential. The cache must first evict dirty blocks, which requires random writes to the HDD, then read the new data from the HDD and finally, write it to the cache. Whereas, without caching it would have simply resulted in a single read from the HDD. To avoid situations such as these, Intel CAS opportunistically flushes dirty data from the cache during idle IO times.

Where are the cached files located?

Intel CAS for Linux does not store files on disk; it uses a pattern of blocks on the SSD as its cache. As such, there is no way to look at the files it has cached.

How do I delete all the Intel CAS for Intel CAS installation files?

Stop the Intel CAS software as described in Chapter 5.3, then uninstall the software as described in Uninstalling the Software on page 13.

Does Intel CAS for Intel CAS support write-back caching?

Yes. Intel CAS for Linux v2.6 and newer supports write-back caching. See Configuration for Write-back Mode on page 16 for details.

Must I stop caching before adding a new pair of cache/core devices?

No, you can create new cache instances while other instances are running.

Can I assign more than one core device to a single cache?

Yes. With Intel CAS for Linux v2.5 and newer, many core devices (up to 32 have been validated) may be associated with a single cache drive or instance. You can add them using the `casadm -A` command.

Can I add more than one cache to a single core device?

No, if you want to map multiple cache devices to a single core device, the cache devices must appear as a single block device through the use of a system such as RAID-0.

Why do tools occasionally report data corruption with Intel CAS?

Some applications, especially micro benchmarks like *dt* and *FIO*, may use a device to perform direct or raw accesses. Some of these applications may also allow you to configure values like a device's alignment and block size restrictions explicitly, for instance via user parameters (rather than simply requesting these values from the device). In order for these programs to work, the block size and alignment for the cache device must match the block size and alignment selected in the tool.

Do I need to partition the cache device?

No. If you do not specify a partition, Intel CAS uses the entire device as the cache device.

Can I use a partition on a SSD as a cache device?

Yes, however, using the entire SSD device as the cache is highly recommended for best performance.

Do I need to format the partition or the device configured as the cache device?

No, the cache device has no format requirement. If any formatting is used, it is transparent to the caching software.

What is the logical and physical block size for Intel CAS cache volumes (for exported objects)?

The logical block size for Intel CAS cache volumes is inherited from the core device, while the physical block size will be represented as the larger of the physical block sizes of the cache or core devices. Note that it is not possible to add a core device to a cache instance when the logical block size of the cache device is greater than that of the core device (eg. when the SSD has 4KiB logical block size and the HDD has 512B logical block size.

What happens if my SSD or HDD becomes unresponsive or disconnected?

In the event of a cache or core device becoming unresponsive, Intel CAS will fail all IO to all exported devices for the related cache (eg. /dev/intelcas1-1, /dev/intelcas1-2, etc). To resume IO to the exported devices for the given cache, the user must restart the affected cache (in this example, cache ID 1).

When my device becomes disconnected, will I receive any notification from Intel® Cache Acceleration Software?

No. The OS does not send notification to Intel CAS of the disconnection of the device, so the software will not know of this event until IO to the device is attempted. The device will still be listed in the `--list-caches` and `--stats` output, and no warning will be logged, until IO to the device is attempted. Check `/var/log/messages` and `dmesg` for standard Linux device IO errors.

Where is the log file located?

All events are logged in the standard Intel CAS system logs. Use the *dmesg* command or inspect the */var/log/messages* file.

To log all messages during testing or kernel debugging, use the command `echo 8 > /proc/sys/kernel/printk`.

Typical log sample of successful cache initialization:

```
[Intel(R) CAS] Cache line size: 64KiB
[Intel(R) CAS] Metadata capacity: 25 MiB
[Intel(R) CAS] Parameters (policies) accepted:: 0 1 1 4
[Intel(R) CAS] Pre-existing metadata, Clean shutdown
[Intel(R) CAS] Done saving cache state!
```

```
[Intel(R) CAS] Cache 1 successfully added
[Intel(R) CAS] IO Scheduler of intelcas1-1 is cfq
[Intel(R) CAS] Core "/dev/sdc" successfully added
```

Typical log sample of successful cache removal:

```
[Intel(R) CAS] Removing Cache 1
[Intel(R) CAS] Trying to remove 1 cached device(s)
[Intel(R) CAS] Removed device intelcas1-1.
[Intel(R) CAS] Done saving cache state!
[Intel(R) CAS] Cache 1 successfully removed
```

Why does flushing the cache drive in Write Back mode take so long?

Flushing time has many factors, including but not limited to, cache device capacity, storage performance, and overall system utilization. Flushing time can vary greatly depending on these factors. You can check the status of the cache device being flushed by using the –L option in casadm.

```
# casadm –L
```

The following command can be used as well to verify activity and loading of the IO devices.

```
# iostat –xmt 1
```

B. Setting up an Intel SSD DC P3608 Series Device

This appendix provides configuration details for Intel® SSD DC P3608 Series Solid-State Drive.

If you are using an Intel SSD DC P3608 Series device as a cache drive, we highly recommended that you either configure the entire SSD cache drive as raw (using RAID-0); or you use just one partition (if partitioning is required), using the *parted* command (described below). For optimal caching performance, you should not break the Intel SSD DC P3608 Series device into multiple partitions.

Raw partition using RAID-0

If you want to RAID the entire SSD cache drive together, use the following command to do so optimally.

```
# mdadm --create --verbose /dev/md0 --level=0 --chunk=64
  --raid-devices=2 /dev/nvme0n1 /dev/nvme1n1
```

where *md0* is the resultant output RAID-0 logical drive, and *nvme0n1* and *nvme1n1* are the corresponding two RAID input drives (applicable to the Intel SSD DC P3608 Series device). RAID level is set to "0"; chunk size is set to "64K".

Note: The *mdadm* command does not save the configuration across system reboots or power-downs. You will need to add the above command in the system files such as: */etc/madadm.conf* and */etc/fstab* if your file system is mounted on a RAID.

Single partition using the *parted* command

Use the following commands to create a single partition cache drive using one of the two exposed Intel SSD DC P3608 devices (half of the total device capacity):

```
# parted -s -- /dev/nvme0n1 mkpart primary 2048s 100%
```

where */dev/nvme0n1p1* is the resultant single partition created.

Note: Intel recommends using the *parted* utility instead of *fdisk*, because older revisions of *fdisk* may cause misaligned partitions (*util-linux-ng package 2.17.2* and earlier revisions). The *parted* tool will align partitions to 1 MB boundaries by default.

C. Generic ssh-keygen Code Snippet

This appendix provides an *ssh-keygen* code snippet that allows you to pair two machines, with “no-password login” enabled.

Generate keys with the *ssh-keygen* command:

```
# ssh-keygen
```

Note: We recommend providing an optional passphrase when prompted. Do not leave the passphrase empty.

This will generate a private/public key pair in the *~/.ssh* directory. In order to copy the public key to a client machine, type the following in the Intel CAS shell:

```
# ssh-copy-id –i ~/.ssh/id_rsa.pub <admin>@<host_ip>
```

To confirm that this secure pairing has worked properly, use the following command to *ssh* to the remote system:

```
# ssh <admin>@<host_ip>
```

You must do this not only to make sure *ssh* setup has worked, but also to accept the RSA key fingerprint. (Otherwise the automatic script will fail.)

Note: Additional command information on this topic is beyond the scope of this guide, but you can consult the *man* pages or any of the freely available *ssh* and *ssh-keygen* examples on various Intel CAS how-to websites for further details.

D. Installation File Tree

This appendix provides a list of paths and file names for the files that get installed by the Intel® Cache Acceleration Software for Linux* v3.1 installer.

Installed to all OSes:

/sbin/casadm
/etc/init.d/intelcas
/etc/intelcas/*
/usr/share/man/man8/casadm.8.gz
/lib/modules/`uname -r`/extra/intelcas.ko (Note: this path may differ if DKMS is enabled)
/lib/modules/`uname -r`/extra/inteldisk.ko (Note: this path may differ if DKMS is enabled)

Additional files installed if DKMS is enabled:

/usr/src/intelcas-<CAS_version>/*

Additional file installed to SystemD based OSes (eg. RHEL 7.0):

/usr/lib/systemd/system/intelcas.service