SDK Installation

Release 1.8.0

Software and Controls Group

Aug 06, 2020

Contents

CHAPTER

ONE

DEVELOPMENT PLATFORM

The development platform is no longer distributed as a standalone ISO file. Instead, the following guide is provided to assist with hardware, operating system and third-party software configuration required for running the OCS SDK. Some key areas, such as disk partitioning, user authentication and NFS mounting of home directories, depending on individual factors at each partner institution and should, therefore, be considered examples only.

Benefits of this approach, versus distributing the complete ISO, include the ability to support different network, hardware and software platforms used for software development by different partner institutions, as well as allowing for separate upgrade schedules for the Development Platform and Software Development Kit without affecting ongoing subsystem software development.

The Observatory Control System (OCS) is designed to be a distributed system with device control components running on real-time computers, connected to common services and user interface components via the control network.

For device control systems, the following operating systems are supported:

- CentOS 8

For user interfaces, the following operating systems are supported:

- MacOS
- CentOS 8

CHAPTER

TWO

SERVER CONFIGURATION

Servers are used for developing, running and testing device control software and core services. When real-time communication with hardware is required, the real-time kernel should be installed and configured. The following guidelines for creating a server should be tailored according to its intended purpose.

2.1 Required Hardware

Minimum hardware requirements for development machines:

- Intel Pentium 4 or higher, 2 GHz CPU
- 1 GB Memory
- 20 GB Hard drive

Typical GMT OCS development machine specs:

- Intel Xeon E3 or higher, 3.2 GHz CPU
- 4 - 8 GB Memory
- 120 - 250 GB Hard drive

2.2 Operating System

1. Install the Centos 8 Operating System. A *minimal* server installation is sufficient for the use of the GMT SDK. Change the filesystem of the partitions to ext4 if using the real-time kernel.

Warning: If you plan to develop real-time components, the Linux kernel requires the root partition to be an **ext4** file system. Please ensure that this is configured correctly in the disk partitioning settings.

Warning: Due to a bug, openssh-server should temporarily not be upgraded to any version greater than 8.0p1-3.el8. To fix the openssh-server version, run this command right after the Operating System installation:

2. Disable firewall

3. Disable SELinux

Warning: Make sure an external firewall protects your server

2.3 Repository Configuration

Some required RPMs are built by GMTO and need to be downloaded from the GMTO Yum Repository, currently hosted on Amazon Web Services.

To add the GMT repositories:

1. Add the file `/etc/yum.repos.d/gmt.repo` with the following content:

```
[gmt]
name=GMT $releasever - $basearch
baseurl=http://52.52.46.32/srv/gmt/yum/stable/$releasever/
gpgcheck=0
enabled=1
```

2. Add the file `/etc/yum.repos.d/gmt-updates.repo` with the following content:

```
[gmt-updates]
name=GMT $releasever - $basearch - Updates
baseurl=http://52.52.46.32/srv/gmt/yum/updates/$releasever/
gpgcheck=0
enabled=1
```

2.4 Package List

An Administrative user should install the following RPM packages for use in the development environment:

1. Install Common OS Utilities

```
$ sudo dnf install -y xorg-x11-xauth urw-fonts wget net-tools pciutils
$ sudo dnf install -y strace bash-completion sed
```

2. Install Development Tools

```
$ sudo dnf install -y autoconf automake cmake elfutils gcc gdb libtool make
$ sudo dnf install -y cpp cscope ctags gc gcc-c++ gcc-gdb-plugin glibc-devel
$ sudo dnf install -y glibc-headers kernel-headers libstdc++-devel
$ sudo dnf install -y flex git libcurl-devel
$ sudo dnf install -y python3
```

3. Install OCS Dependencies

```
$ sudo dnf install -y rdma librdmacm-devel boost-devel
```

2.5 Node Installation

1. Install Python 2 and set is a default interpreter. It is necessary for the node package manager.

```
$ sudo dnf install -y python2
$ sudo alternatives --set python /usr/bin/python2
```

2. Install **Node version 12**:

```
$ sudo dnf module install -y nodejs:12
```

3. Install necessary node packages:

```
$ sudo npm install -g coffeescript webpack webpack-cli raw-loader
```

Note: If you encounter problems installing nodejs 12, you probably have the node module v10 activated. To change it to v12, run: *sudo dnf remove -y nodejs && sudo dnf module reset -y nodejs && sudo dnf module enable -y nodejs:12*

2.6 MongoDB Configuration

1. Add the file `/etc/yum.repos.d/mongodb-org-4.repo` with the following content:

```
[mongodb-org-4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/7/mongodb-org/4.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.2.asc
```

2. Install the necessary packages:

```
$ sudo dnf -y install mongodb-org
```

3. Enable the MongoDB service

```
$ sudo systemctl enable mongod
$ sudo systemctl start mongod
```

4. Check that the MongoDB service is up

```
$ sudo systemctl status -l mongod
```

2.7 EtherCAT Configuration

EtherCAT is a high-speed Fieldbus communication system used for real-time control. The following configuration steps should be used as a guide when configuring EtherCAT communications.

1. Install the real-time kernel and relevant packages

```
$ sudo dnf install -y --nogpgcheck kernel-3.14.73-rt78.x86_64 ethercat-devel
```

Warning: Before installing the RT kernel, check restrictions on Operating System warnings.

2. Select the Ethernet interface to be used for EtherCAT communication (e.g. enp4s0) and edit the corresponding configuration file (e.g. `/etc/sysconfig/network-scripts/ifcfg-enp4s0`) to set the following options:

```
BOOTPROTO=none
ONBOOT=no
```

3. Check the Hardware Address (MAC) of the selected EtherCAT network interface

```
$ ifconfig
```

4. Edit `/etc/ethercat.conf` and set the following configuration options:

```
MASTER0_DEVICE="<mac_address_1>"
MASTER0_BACKUP="<mac_address_2>"  # optional line
```

   Where `<mac_address_1>` and `<mac_address_2>` are the two hardware addresses associated with the Ethercat network interface communicating with the Ethercat ring (redundant topology). If you prefer using a linear topology (non-redundant), comment or remove the second line (`MASTER0_BACKUP="<mac_address_2>"`).

5. Edit `/usr/lib/systemd/system/ethercat.service` and uncomment the following line:

```
Before=network.service
```

6. Reboot into the RT Kernel, if you're not in it already.

7. Enable the Ethercat service

```
$ sudo systemctl enable ethercat
$ sudo systemctl start ethercat
```

8. Edit `/etc/security/limits.d/99-realtime.conf` and add the following options:

```
@realtime - rtprio 99
@realtime - memlock unlimited
```

9. Add a new group and add the "gmto" user to it.

```
$ sudo groupadd -f -g 2001 realtime
$ sudo usermod --groups realtime gmto
```

8. Test the Ethercat configuration

```
$ ethercat master
$ ethercat slaves
```

If the `ethercat master` command does not produce the correct output, ensure that you're currently running the real-time kernel. If the `ethercat slaves` command produces no output, check that the ethernet cable is connected to the correct port as configured above.

2.8 Network Time Protocol Configuration

For general network timekeeping, use NTP, unless Precision Time Protocol is required.

1. Install the necessary packages:

```
$ sudo dnf install -y chrony
```

2. Enable the NTP Service

```
$ sudo systemctl enable chronyd
```

3. Check date/time servers

```
$ sudo chronyc sources
```

CHAPTER

THREE

SOFTWARE DEVELOPMENT KIT (SDK)

The Software Development Kit is distributed as a TAR file and can be downloaded from the GMTO release server.

The SDK should be installed in a **Global GMT Software Location**, defined by the GMT_GLOBAL environment variable (default value: /opt/gmt). A **Local Working Directory**, defined by the GMT_LOCAL variable, is used as a unique workspace for individual developers. The local working directory typically resides underneath the /home/<username> directory.

1. Download the latest SDK distribution:

```
$ wget http://52.52.46.32/srv/gmt/releases/sdk/linux/gmt-sdk.tar.gz
```

2. Extract the TAR file in the /opt directory, into a new folder for the latest release:

```
$ sudo mkdir /opt/gmt_release_1.8.0
$ sudo tar -xzvf gmt-sdk.tar.gz -C /opt/gmt_release_1.8.0
```

   where gmt-sdk.tar.gz is the file downloaded in step 1.

3. Create a symbolic link from the **Global GMT Software Location** to the latest release:

```
$ sudo ln -sfn /opt/gmt_release_1.8.0 /opt/gmt
```

4. Create a **Local Working Directory**

```
$ mkdir <local_working_dir>
```

   where `<local_working_dir>` is in the current users' home directory, for example, ~/work. The GMT software modules developed by the user are created in this folder.

5. Add the following lines to your .bash_profile (or .kshrc or .bashrc depending on your preferred shell)

```
$ export GMT_GLOBAL=/opt/gmt
$ export GMT_LOCAL=<local_working_dir>
$ source $GMT_GLOBAL/bin/gmt_env.sh
```

   This will ensure that the environment variables are correctly configured when opening a new terminal. Please log out and back in for the changes to take effect. To configure the environment for the current shell, run the commands manually.

6. Check the values of the environment variables:

```
$ gmt_env
```

7. Install Node Modules

```
$ cd $GMT_GLOBAL
$ sudo npm install

$ cd $GMT_LOCAL
$ cp $GMT_GLOBAL/package.json ./
$ npm install
```

8. Initialize the Development Environment:

```
$ cd $GMT_LOCAL
$ gds init
```

   The correct folders will be created in the $GMT_LOCAL directory for use when compiling and running modules.

9. Create a **modules** directory in $GMT_LOCAL

```
$ cd $GMT_LOCAL
$ mkdir modules
```

10. Create the **bundles.coffee** and **ocs_local_bundle.coffee** files, defining the local modules under development

    These files may be copied from $GMT_GLOBAL and then edited to reflect the developer's configuration.

```
$ mkdir $GMT_LOCAL/etc/bundles
$ cp $GMT_GLOBAL/etc/bundles/bundles.coffee $GMT_LOCAL/etc/bundles/
$ cp $GMT_GLOBAL/etc/bundles/ocs_local_bundle.coffee $GMT_LOCAL/etc/bundles/
```

    Edit **bundles.coffee** to point to the ocs_local_bundle.coffee file

```
module.exports =
    ocs_local_bundle:   {scope: "local",  desc: "GMT iSample and HDK bundle"}
```

    Edit **ocs_local_bundle.coffee** to include the ISample and HDK modules, or other modules that you are working on

```
module.exports =
name:       "local"
desc:       "List of local development modules"
elements:
    isample_dcs: { active: true, test: false, developer: 'gmto', domain:
↪'idcs' }
    hdk_dcs:     { active: true, test: false, developer: 'gmto', domain:
↪'idcs' }
```

11. Build all model files from modules in your ocs_local_bundles definition using webpack. For example:

```
$ cd $GMT_LOCAL/modules/ocs_hdk_dcs/model
$ webpack
$ cd $GMT_LOCAL/modules/ocs_isample_dcs/model
$ webpack
```