Atlas Data Center Solution
V100R020C00

CANN Software Installation Guide
(A800-9010)

Issue 01
Date 2020-08-03
Contents

1 Installation Scenarios ............................................................................................................. 1

2 Hardware Requirements ........................................................................................................ 3

3 Installing the Development Environment ......................................................................... 5
  3.1 Before You Start .............................................................................................................. 5
  3.2 Obtaining Software Packages ......................................................................................... 6
  3.3 Before You Start .............................................................................................................. 8
    3.3.1 Ubuntu (x86) ........................................................................................................... 8
    3.3.2 CentOS (x86) ......................................................................................................... 14
    3.3.3 Debian (x86) .......................................................................................................... 21
  3.4 Pure Development ........................................................................................................... 27
    3.4.1 Installing the Development Kit (Easy Installation Tool) ........................................... 27
    3.4.2 Installing the Development Kit (CLI) ................................................................. 28
  3.5 Development + Commissioning ...................................................................................... 31
    3.5.1 Installing the Ascend Chip Driver and Firmware ..................................................... 31
    3.5.2 Installing the Development Kit (Easy Installation Tool) ........................................... 32
    3.5.3 Installing the Development Kit (CLI) ................................................................. 33
    3.5.4 Installing the Deep Learning Framework .............................................................. 36
    3.5.5 Installing protobuf Python .................................................................................... 39
    3.5.6 Changing NPU IP Addresses ................................................................................. 42
  3.6 Verifying the Installation ................................................................................................. 42

4 Installing the Operating Environment ............................................................................. 45
  4.1 Before You Start .............................................................................................................. 45
  4.2 Obtaining Software Packages ......................................................................................... 46
  4.3 Installation on a Physical Machine .................................................................................. 49
    4.3.1 Before You Start ..................................................................................................... 49
      4.3.1.1 Ubuntu (x86) ................................................................................................ 49
      4.3.1.2 CentOS (x86) .............................................................................................. 55
      4.3.1.3 Debian (x86) .............................................................................................. 55
    4.3.2 Installing the Ascend Chip Driver Firmware and Training Software (Easy Installation Tool) ................................................................. 62
    4.3.3 Installing the Ascend Chip Driver Firmware and Training Software (CLI) .................. 71
      4.3.3.1 Installing the Ascend Chip Driver and Firmware ............................................ 71
      4.3.3.2 Installing the Training Software .................................................................... 71
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.2.6 Obtains the MAC address of an RoCE NIC</td>
<td>127</td>
</tr>
<tr>
<td>7.3.2.7 Queries the LLDP information</td>
<td>128</td>
</tr>
<tr>
<td>7.3.2.8 Restores the configuration of the 100GE NIC</td>
<td>128</td>
</tr>
<tr>
<td>7.3.2.9 Shuts down a link</td>
<td>129</td>
</tr>
<tr>
<td>7.3.2.10 Obtains the status of a link</td>
<td>129</td>
</tr>
<tr>
<td>7.3.2.11 Sets the MTU of an RoCE NIC</td>
<td>130</td>
</tr>
<tr>
<td>7.3.2.12 Obtains the MTU of an RoCE NIC</td>
<td>131</td>
</tr>
<tr>
<td>7.3.2.13 Obtains the network health status</td>
<td>131</td>
</tr>
<tr>
<td>7.3.2.14 Sets the MAC address of an RoCE NIC</td>
<td>132</td>
</tr>
<tr>
<td>7.3.2.15 Sets the DSCP to TC mapping</td>
<td>132</td>
</tr>
<tr>
<td>7.3.2.16 Obtains the DSCP to TC mapping</td>
<td>133</td>
</tr>
<tr>
<td>7.3.2.17 Obtains the ARP table</td>
<td>134</td>
</tr>
<tr>
<td>7.3.2.18 Obtains the routing table</td>
<td>134</td>
</tr>
<tr>
<td>7.3.2.19 Obtains PCI device information</td>
<td>135</td>
</tr>
<tr>
<td>7.3.2.20 Obtains QPC information</td>
<td>136</td>
</tr>
<tr>
<td>7.3.2.21 Obtains AEQC information</td>
<td>136</td>
</tr>
<tr>
<td>7.3.2.22 Obtains CEQC information</td>
<td>137</td>
</tr>
<tr>
<td>7.3.2.23 Obtains CQC information</td>
<td>137</td>
</tr>
<tr>
<td>7.3.2.24 Obtains MPT information</td>
<td>138</td>
</tr>
<tr>
<td>7.3.2.25 Obtains optical module information</td>
<td>139</td>
</tr>
<tr>
<td>7.3.2.26 Obtains register information</td>
<td>139</td>
</tr>
<tr>
<td>7.3.2.27 Obtains the IP address and subnet mask</td>
<td>140</td>
</tr>
<tr>
<td>7.3.2.28 Obtains statistics</td>
<td>140</td>
</tr>
<tr>
<td>7.3.2.29 Sets the bandwidth limit of an RoCE NIC</td>
<td>141</td>
</tr>
<tr>
<td>7.3.2.30 Obtains the bandwidth limit of an RoCE NIC</td>
<td>142</td>
</tr>
<tr>
<td>7.3.2.31 Restores the configuration of all RoCE NICs</td>
<td>142</td>
</tr>
<tr>
<td>7.3.2.32 Sets TLS</td>
<td>143</td>
</tr>
<tr>
<td>7.3.2.32.1 Presets or replaces the certificate suite</td>
<td>143</td>
</tr>
<tr>
<td>7.3.2.32.2 Enables or disables TLS</td>
<td>145</td>
</tr>
<tr>
<td>7.3.2.32.3 Sets the TLS certificate expiry alarm threshold</td>
<td>146</td>
</tr>
<tr>
<td>7.3.2.32.4 Obtains TLS information</td>
<td>147</td>
</tr>
<tr>
<td>7.4 Performing Fault Diagnostics</td>
<td>148</td>
</tr>
<tr>
<td>7.5 Performing a Software and Hardware Compatibility Test</td>
<td>151</td>
</tr>
<tr>
<td>A Change History</td>
<td>155</td>
</tr>
</tbody>
</table>
1 Installation Scenarios

Installing the Development Environment

The development environment is used to develop and commission operators.

In addition to developing and commissioning operators, if you need to train network models in the development environment, deploy the development environment on an Ascend AI device (such as the training server or training card).

The development kit needs to be installed in the development environment.

In the model training scenario, the driver, firmware, and deep learning framework need to be installed.

Installing the Operating Environment

The operating environment is the actual environment for model training. The Ascend AI device (such as the training server or training card) must be configured in the installation environment.

The operating environment supports the following installation modes:

- Using a physical machine
  a. Install the Ascend chip driver and firmware.
  b. Install the training software, including the deep learning acceleration engine package, framework plug-in package, and toolbox package.
  c. Install the deep learning framework. After the training software is installed, you need to install TensorFlow before developing and verifying operators and training services.
  d. Install protobuf Python. If the training script depends on the Python version of protobuf to store data in the serialized structure (for example, the serialization interfaces of TensorFlow), you need to install protobuf Python.
  e. Configuring the IP address of the NPU card: If the AI training device is deployed in a cluster, you need to configure the NPU IP address after installing the driver, firmware, and training software. The IP address is used to transmit network model parameters between servers during model training, implementing network model parameter synchronization between servers.
Installation in a container: Use the following installation procedure.

a. Install the Ascend chip driver and firmware on the host machine.

b. Install the Toolbox on the host machine.

c. Install Ascend Docker on the host.

d. Use the Docker file to create a container image.

e. Deploy the image file: Upload the packed image file to the operating environment for deployment.

f. Configuring the IP address of the NPU card: If the AI training device is deployed in a cluster, you need to configure the NPU IP address after installing the driver, firmware, and training software. The IP address is used to transmit network model parameters between servers during model training, implementing network model parameter synchronization between servers.

Easy Installation Tool

The easy installation tool allows you to install software required by a specific service scenario by one click. For example, you can install the training operating environment and install Ascend chip drivers, firmware, deep learning engines, framework plug-ins, and utility kit at the same time by one click. Therefore, the easy installation tool is recommended for you to install the operating environment on a physical machine to improve the installation efficiency.

The installation of the development environment mainly involves the installation of the development kit. Therefore, the speed of installing the development environment by using the easy installation tool is similar to that by using the CLI. You can select the installation mode as required.
# 2 Hardware Requirements

Before installing the development and operating environments, you need to prepare the hardware environment based on the scenario.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Hardware</th>
<th>Hardware Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development environment</td>
<td>General-purpose server</td>
<td>-</td>
</tr>
<tr>
<td>Development and commissioning</td>
<td>Atlas 800 AI training card (Model 9010)</td>
<td>Install a server in a cabinet, configure basic server parameters, and install an OS. For details, see the <em>Atlas 800 AI Training Server User Guide (Model 9010)</em>.</td>
</tr>
<tr>
<td>Operating environment</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

## Table 2-1 Supported OSs

<table>
<thead>
<tr>
<th>OS</th>
<th>Version</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu</td>
<td>OS version: Ubuntu 18.04.2</td>
<td>Official Ubuntu website</td>
</tr>
<tr>
<td></td>
<td>Kernel version: 4.15.0-45-generic</td>
<td>Download the recommended version <em>ubuntu-18.04.2-server-amd64.iso</em> from <a href="http://old-releases.ubuntu.com/releases/18.04.2/">http://old-releases.ubuntu.com/releases/18.04.2/</a>.</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86</td>
<td></td>
</tr>
<tr>
<td>CentOS</td>
<td>OS version: CentOS 7.6</td>
<td>Official CentOS website</td>
</tr>
<tr>
<td></td>
<td>Kernel version: 3.10.0-957.el7.x86_64</td>
<td>Download the required version from <a href="http://vault.centos.org/7.6.1810/isos/x86_64/">http://vault.centos.org/7.6.1810/isos/x86_64/</a>. The recommended version is CentOS-7-x86_64-DVD-1810.iso.</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86</td>
<td></td>
</tr>
<tr>
<td>OS</td>
<td>Version</td>
<td>How to Obtain</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------</td>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>Debian</td>
<td>OS version: Debian 9.9</td>
<td>Official Debian website</td>
</tr>
<tr>
<td></td>
<td>Kernel version: 4.9.0-9-amd64</td>
<td>Download the recommended version</td>
</tr>
<tr>
<td></td>
<td>Processor architecture: x86</td>
<td>debian-9.9.0-amd64-DVD-1.iso from</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="https://cdimage.debian.org/cdimage/archive/9.9.0/amd64/iso-dvd/">https://cdimage.debian.org/cdimage/archive/9.9.0/amd64/iso-dvd/</a>.</td>
</tr>
</tbody>
</table>
3 Installing the Development Environment

3.1 Before You Start

- If the development environment is used only to convert model files and develop and compile operators, the development environment uses a general-purpose server. You do not need to configure Ascend AI devices, and only need to install the development kit (Ascend-Toolkit-{version}-{CPU type}-{linux_gcc_version}.run). For details, see the following sections:
  a. 3.2 Obtaining Software Packages
  b. 3.3 Before You Start
  c. 3.4 Pure Development

- In addition to converting model files and developing operators, you need to train and debug the developed network model in the development environment. In this case, the development environment must be an Ascend AI device (such as a training server or training card). You need to install the driver of the Ascend AI device, firmware, and development kit (Ascend-Toolkit-{version}-{CPU type}-{linux_gcc_version}.run). For details, see the following sections:
  a. 3.2 Obtaining Software Packages
  b. 3.3 Before You Start
  c. 3.5 Development + Commissioning
3.2 Obtaining Software Packages

Before installing the software, obtain software packages based on the actual scenario. The versions of the software packages must be consistent. The procedure is as follows:

**Step 1** Log in to the Resources page and select the corresponding product series and product model.

**Step 2** Obtain the software package based on the actual scenario.
- For the pure development scenario, see Table 3-1.
- For the development+commissioning scenario, see Table 3-2.

<table>
<thead>
<tr>
<th>Name</th>
<th>Package</th>
<th>OS Architecture</th>
<th>Installation User</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development kit</td>
<td>Ascend-Toolkit-{version}-x86_64-linux_gcc7.3.0.run</td>
<td>CentOS x86</td>
<td>root</td>
<td>It is mainly used for developing custom operators.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Ubuntu x86</td>
<td>non-root</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Debian x86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Easy installation tool</td>
<td>ascend-sdk-manager-{version}-x86-{os}.tgz</td>
<td>Ubuntu x86</td>
<td>root</td>
<td>Use an easy installation tool to install the development and operating environments.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CentOS x86</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Debian x86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Package</td>
<td>OS Architecture</td>
<td>Installation User</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Ascend chip driver installation package</td>
<td>A800-9010-NPU_Driver-{version}-X86_64-Ubuntu18.04.run</td>
<td>Ubuntu x86</td>
<td>The installation user must be <strong>root</strong>, and the running user must be <strong>HwHiAiUser</strong>. <strong>HwHiAiUser</strong> must be created before the Ascend chip driver and firmware are installed.</td>
<td>NPU driver</td>
</tr>
<tr>
<td></td>
<td>A800-9010-NPU_Driver-{version}-X86_64-CentOS7.6_gcc7.3.0.run</td>
<td>CentOS x86</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A800-9010-NPU_Driver-{version}-X86_64-Debian9.9.run</td>
<td>Debian x86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ascend chip firmware installation package</td>
<td>A800-9010-NPU_Firmware-{version}.run</td>
<td></td>
<td></td>
<td>NPU firmware</td>
</tr>
<tr>
<td>Development kit</td>
<td>Ascend-Toolkit-{version}-x86_64-linux_gcc7.3.0.run</td>
<td>CentOS x86, Ubuntu x86, Debian x86</td>
<td>root or non-root. If the installation user is a non-root user, the owner group of the installation user must be the same as that of the Ascend chip driver operating user <strong>HwHiAiUser</strong>. If you use the easy installation tool to install the software, perform operation as the <strong>root</strong> user.</td>
<td>It is mainly used for developing custom operators.</td>
</tr>
</tbody>
</table>
### Table 3-3 Dependency information

<table>
<thead>
<tr>
<th>Category</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
</tr>
<tr>
<td>cmake</td>
<td>3.5.1+</td>
</tr>
<tr>
<td>gcc</td>
<td>7.3.0 or later</td>
</tr>
<tr>
<td>g++</td>
<td>7.3.0 or later</td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
</tr>
</tbody>
</table>

#### 3.3 Before You Start

**3.3.1 Ubuntu (x86)**

**Environment Requirements**

The following software or dependencies need to be installed in the development environment: For details, see Installing Dependencies.

---

{version} indicates the software version, and {os} indicates the OS.

---End
### Category

<table>
<thead>
<tr>
<th>Category</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>There is no version requirement. The version to be installed is subject to the source provided by the OS.</td>
</tr>
<tr>
<td>zlib1g</td>
<td></td>
</tr>
<tr>
<td>zlib1g-dev</td>
<td></td>
</tr>
<tr>
<td>libbz2-dev</td>
<td></td>
</tr>
<tr>
<td>libsqlite3-dev</td>
<td></td>
</tr>
<tr>
<td>libssl-dev</td>
<td></td>
</tr>
<tr>
<td>libxslt1-dev</td>
<td></td>
</tr>
<tr>
<td>libffi-dev</td>
<td></td>
</tr>
<tr>
<td>unzip</td>
<td></td>
</tr>
<tr>
<td>pciutils</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>grpcio</td>
<td></td>
</tr>
<tr>
<td>grpcio-tools</td>
<td></td>
</tr>
<tr>
<td>protobuf</td>
<td></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td></td>
</tr>
</tbody>
</table>

---

**Check the umask of the root user.**

1. Log in to the installation environment as the **root** user.
2. Check the **umask** value of the **root** user.
   ```bash
   umask
   ```
3. If the **umask** value is not **0022**, append "umask 0022" to the file and save the file:
   ```bash
   vi ~/.bashrc
   source ~/.bashrc
   ```

**Creating Installation and Running Users**

*Table 3-4* lists the users for installing and running the Ascend chip driver, firmware, and CANN software.
### Table 3-4 Installation and running user list

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Installation User</th>
<th>Running User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascend chip driver and firmware installation package</td>
<td>The user must be <strong>root</strong>.</td>
<td>The value must be <strong>HwHiAiUser</strong>, and <strong>HwHiAiUser</strong> must be created before the Ascend chip driver and firmware are installed.</td>
</tr>
<tr>
<td>CANN software package</td>
<td>The user can be <strong>root</strong> or non-root. For details, see the following content below the table.</td>
<td>The user must be a non-root user. For details, see the following content below the table.</td>
</tr>
</tbody>
</table>

You must create the **HwHiAiUser** user before installing the CANN software as the **root** user or a non-root user.

1. Create the **HwHiAiUser** user.
   
   Switch to the **root** user and run the following command to create the **HwHiAiUser** user:
   ```
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser
   ``

   The user group of the **HwHiAiUser** user must be **HwHiAiUser**.

2. Select an installation mode.
   
   - **For installation as the root user:**
     You can install the software as the **root** user, but you must create the **HwHiAiUser** user. After the CANN software is installed, you must run the software as a non-root user, that is, the running user can only be in the **HwHiAiUser** user group.

   - **For installation as a non-root user:**
     In this scenario, the installation and running users must be the same.
     
     - If a non-root user exists, add the user to the **HwHiAiUser** user group and switch to the user for normal installation. The default installation user and running user are the current user.
     
     - If you want to use a new non-root user, you need to create the user first.

   To create a non-root user, run the following commands as the **root** user:
   
   a. Create a non-root user.
      ```
      useradd -g HwHiAiUser -d /home/username -m username
      ``
      
      **username** is user-defined, but its owner group must be **HwHiAiUser**.
   
   b. Set the password of the non-root user.
      ```
      passwd username
      ```
● The created running user cannot belong to the root user group.

● After the HwHiAiUser user is created, do not disable the login authentication function of the user.

● The password validity period is 90 days. You can change the validity period in the /etc/login.defs file or using the chage command. For details, see 5.5 Parameters description.

Checking the Source Validity

Before installing the deep learning engine package, toolbox, and framework plug-in package, you need to download related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the root user to check whether the source is valid:

```bash
apt-get update
```

If an error is reported during command execution or dependency installation, check whether the network connection is normal, or replace the source in the /etc/apt/sources.list file with an available source or use an image source. For details about how to configure a network proxy, see 6.7 Configuring a System Network Proxy.

Configuring Permissions for the Installation User

If a non-root user is used for installation, perform the operations described in this section. If the root user is used for installation, you only need to perform step 1.

Before installing the development kit, you need to download the dependencies, which require sudo apt-get permission. Run the following commands as the root user:

1. Run the following command to install sudo:
   ```bash
   apt-get install sudo
   ```

2. Open the /etc/sudoers file:
   ```bash
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

3. Add the following content below # User privilege specification of the file:
   ```bash
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3, /bin/ln -s /usr/local/python3.7.5/bin/python3.7.5 /usr/bin/python3, /bin/ln -s /usr/local/python3.7.5/bin/pip3.7 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3, /bin/ln -s /usr/local/python3.7.5/bin/pip3.7.5 /usr/bin/pip3.7.5
   ```

   Replace username with the name of the common user who executes the installation script.

4. Run :wq! to save the file.

5. Run the following command to revoke the write permission on the /etc/sudoers file:
   ```bash
   chmod u-w /etc/sudoers
   ```
Installing Dependencies

**NOTE**

- If you install Python and its dependencies as the root user, perform steps 1 to 3. Note that sudo needs to be deleted from the commands in steps 1 to 3.
- If python and its dependency are installed as a non-root user, run the su - username command to switch to the non-root user, and then perform steps 1 to 3.

**Step 1** Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

```
gcc --version

g++ --version

make --version

cmake --version

dpkg -l zlib1g| grep zlib1g| grep ii
dpkg -l zlib1g-dev| grep libbz2-dev| grep ii

dpkg -l libbz2-dev| grep libbz2-dev| grep ii

dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii

dpkg -l openssl| grep openssl| grep ii

dpkg -l libssl-dev| grep libssl-dev| grep ii

dpkg -l libxslt1-dev| grep libxslt1-dev| grep ii

dpkg -l libffi-dev| grep libffi-dev| grep ii

dpkg -l unzip| grep unzip| grep ii

dpkg -l pciutils| grep pciutils| grep ii

dpkg -l net-tools| grep net-tools| grep ii
```

If the following information is displayed, the installation is complete. Go to the next step.

```
gcc (Ubuntu/Linaro 7.5.0-3ubuntu1-18.04) 7.5.0
GNU Make 4.7

cmake version 3.10.2

liblzma amd64 1:5.2.1-1 Ubuntu-18.04 amd64 compression library - runtime

zlib1g amd64 1:1.2.11.dfsg-0ubuntu2 amd64 compression library - runtime

zlib1g-dev amd64 1:1.2.11.dfsg-0ubuntu2 amd64 compression library - development

libbz2 amd64 1:1.0.6-8.1ubuntu0.2 amd64 high-quality block-sorting file compressor library - development

libsqlite3 amd64 3.22.0-1ubuntu0.3 amd64 SQLite 3 development files

openssl 1.1.1-1ubuntu1-18.04.6 amd64 Secure Sockets Layer toolkit - cryptographic utility

libssl amd64 1:1.1.1-1ubuntu1-18.04.6 amd64 Secure Sockets Layer toolkit - cryptography

libssl-dev amd64 1:1.1.1-1ubuntu1-18.04.6 amd64 Secure Sockets Layer toolkit - development files

libsqlite3-dev amd64 1:1.12.1-1ubuntu0.3 amd64 SQLite 3 development files

libffi amd64 1:3.2.1-8 amd64 Foreign Function Interface library (development files)

unzip 6.0-2ubuntu1 amd64 De-archiver for .zip files

pcitools 1.3.5.2-1ubuntu1 amd64 Linux PCI Utilities

net-tools 1.60+git20161116.90da8a0-1ubuntu1 amd64 NET-3 networking toolkit
```

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

```
sudo apt-get install gcc g++ make cmake zlib1g zlib1g-dev libbz2-dev libsqlite3-dev libssl-dev libssl1-dev libffi-dev unzip pcitools net-tools -y
```

**Step 2** Check whether the Python development environment is installed.

The development kit depends on the Python environment. Run the python3.7.5 --version and pip3.7.5 --version commands to check whether Python has been installed. If the following information is displayed, Python has been installed. Go to the next step.

```
Python 3.7.5
pip 7.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:
1. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the installation environment. The command is as follows:

```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
```

2. Run the following command to go to the download directory and decompress the source code package:

```
tar -zxvf Python-3.7.5.tgz
```

3. Go to the decompressed folder and run the following configuration, build, and installation commands:

```
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5
make
```

The `--prefix` parameter specifies the Python installation path. You can change it based on the site requirements. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library.

This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, compilation, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

4. Check whether `libpython3.7m.so.1.0` exists in `/usr/lib64` or `/usr/lib`. If yes, skip this step or back up the `libpython3.7m.so.1.0` file provided by the system and run the following command:

```
Copy the compiled file `libpython3.7m.so.1.0` to `/usr/lib64`:
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
```

If the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.

```
cp: overwrite 'libpython3.7m.so.1.0'?
```

If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.

```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
```

Replace the path of the `libpython3.7m.so.1.0` file as required.

5. Run the following commands to set the soft link:

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
dsud0 ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
```

If a message indicating that the link already exists is displayed during the command execution, run the following command to delete the existing link and run the command again:

```
sudo rm -rf /usr/bin/python3.7.5
dsud0 rm -rf /usr/bin/pip3.7.5
```

6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

```
python3.7.5 --version
pip3.7.5 --version
```

**Step 3** Table 3-5 lists the dependencies to be installed. Before installing a software package, ensure that its dependencies have been installed. If you run the following
command as a non-root user, add `--user` to the end of the command. For example:

`pip3.7 install numpy --user`. The installation command can be run in any path.

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Version</th>
<th>Installation Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
<td>For details, see Step 5 Check whether the Python development environment has been installed.</td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3</td>
<td><code>pip3.7 install numpy</code></td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
<td><code>pip3.7 install decorator</code></td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
<td><code>pip3.7 install sympy==1.4</code></td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
<td><code>pip3.7 install cffi==1.12.3</code></td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
<td><code>pip3.7 install pyyaml</code></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
<td><code>pip3.7 install pathlib2</code></td>
</tr>
<tr>
<td>grpcio</td>
<td></td>
<td><code>pip3.7 install grpcio</code></td>
</tr>
<tr>
<td>grpcio-tools</td>
<td></td>
<td><code>pip3.7 install grpcio-tools</code></td>
</tr>
<tr>
<td>protobuf</td>
<td></td>
<td><code>pip3.7 install protobuf</code></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
<td><code>pip3.7 install scipy</code></td>
</tr>
<tr>
<td>requests</td>
<td></td>
<td><code>pip3.7 install requests</code></td>
</tr>
</tbody>
</table>

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, rectify the fault by referring to 6.2 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When `pip3.7.5 install` Is Run?.

If the error message "subprocess.CalledProcessError: Command '('/lsb_release', '-a')' return non-zero exit status 1 " is displayed, rectify the fault by referring to 6.4 The error message "subprocess.CalledProcessError: Command '('/lsb_release', '-a')' return non-zero exit status 1 " is displayed during `pip3.7.5 install` installation.

----End

### 3.3.2 CentOS (x86)

#### Environment Requirements

The following software or dependencies need to be installed in the development environment: For details, see Installing Dependencies.
Table 3-6 Dependency information

<table>
<thead>
<tr>
<th>Category</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
</tr>
<tr>
<td>cmake</td>
<td>3.5.1+</td>
</tr>
<tr>
<td>gcc</td>
<td>4.8.5</td>
</tr>
<tr>
<td>g++</td>
<td>NOTE: CentOS 7.6 uses GCC 4.8.5 by default, which needs to be upgraded to GCC 7.3.0.</td>
</tr>
<tr>
<td>numpy</td>
<td>1.17.2</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
</tr>
<tr>
<td>make</td>
<td></td>
</tr>
<tr>
<td>unzip</td>
<td></td>
</tr>
<tr>
<td>zlib-devel</td>
<td></td>
</tr>
<tr>
<td>libffi-devel</td>
<td></td>
</tr>
<tr>
<td>openssl-devel</td>
<td></td>
</tr>
<tr>
<td>pciutils</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>sqlite-devel</td>
<td></td>
</tr>
<tr>
<td>blas-devel</td>
<td></td>
</tr>
<tr>
<td>lapack-devel</td>
<td></td>
</tr>
<tr>
<td>openblas-devel</td>
<td></td>
</tr>
<tr>
<td>gcc-gfortran</td>
<td></td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>grpcio</td>
<td></td>
</tr>
<tr>
<td>grpcio-tools</td>
<td></td>
</tr>
<tr>
<td>protobuf</td>
<td></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td>There is no version requirement. The version to be installed is subject to the source provided by the OS.</td>
</tr>
</tbody>
</table>

Check the umask of the root user.

1. Log in to the installation environment as the root user.
2. Check the umask value of the root user.
   ```bash
   umask
   ```
3. If the umask value is not 0022, append "umask 0022" to the file and save the file:
Creating Installation and Running Users

Table 3-7 lists the users for installing and running the Ascend chip driver, firmware, and CANN software.

Table 3-7 Installation and running user list

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Installation User</th>
<th>Running User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascend chip driver and firmware installation</td>
<td>The user must be root.</td>
<td>The value must be HwHiAiUser, and HwHiAiUser must be created before the Ascend chip driver and firmware are installed.</td>
</tr>
<tr>
<td>CANN software package</td>
<td>The user can be root or non-root. For details, see the following content below the table.</td>
<td>The user must be a non-root user. For details, see the following content below the table.</td>
</tr>
</tbody>
</table>

You must create the HwHiAiUser user before installing the CANN software as the root user or a non-root user.

1. Create the HwHiAiUser user.
   
   Switch to the root user and run the following command to create the HwHiAiUser user:
   ```bash
   groupadd HwHiAiUser           //Create the HwHiAiUser user group.
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser           // Create an HwHiAiUser user whose owner group is HwHiAiUser.
   
   The user group of the HwHiAiUser user must be HwHiAiUser.
   
   2. Select an installation mode.
      
      - For installation as the root user:
        You can install the software as the root user, but you must create the HwHiAiUser user. After the CANN software is installed, you must run the software as a non-root user, that is, the running user can only be in the HwHiAiUser user group.
      
      - For installation as a non-root user:
        In this scenario, the installation and running users must be the same.
        
        - If a non-root user exists, add the user to the HwHiAiUser user group and switch to the user for normal installation. The default installation user and running user are the current user.
        
        - If you want to use a new non-root user, you need to create the user first.

To create a non-root user, run the following commands as the root user:
a. Create a non-root user.

```
useradd -g HwHiAiUser -d /home/username -m username
```

*username* is user-defined, but its owner group must be *HwHiAiUser*.

b. Set the password of the non-root user.

```
pwd username
```

**NOTE**

- The created running user cannot belong to the *root* user group.
- After the *HwHiAiUser* user is created, do not disable the login authentication function of the user.
- The password validity period is 90 days. You can change the validity period in the */etc/login.defs* file or using the *chage* command. For details, see 5.5 Parameters description.

### Checking the Source Validity

The installation requires the download of related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the *root* user to check whether the source is valid:

```
yum repolist
```

If an error is reported during command execution or dependency installation, check whether the network connection is normal, or replace the source in the */etc/yum.repos.d/xxxx.repo* file with an available source or use an image source. For details about how to configure a network proxy, see 6.7 Configuring a System Network Proxy.

### Configuring Permissions for the Installation User

If a non-root user is used for installation, perform the operations described in this section. If the *root* user is used for installation, you only need to perform step 1.

Before installing the development kit, you need to download the dependencies, which require *sudo yum* permission. Run the following commands as the *root* user:

1. Run the following command to install *sudo*:
   ```
yum install sudo
   ```

2. Open the */etc/sudoers* file:
   ```
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

3. Add the following content below # User privilege specification of the file:
   ```
   username ALL=(ALL:ALL)   NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```

   Replace *username* with the name of the common user who executes the installation script.

**NOTE**

Ensure that the last line of the */etc/sudoers* file is #includedir /etc/sudoers.d. Otherwise, add it manually.
4. Run `:wq!` to save the file.

5. Run the following command to revoke the write permission on the `/etc/ sudoers` file:
   ```
   chmod u-w /etc/sudoers
   ```

### Installing Dependencies

**NOTE**

- If you install Python and its dependencies as the root user, perform steps 1 to 4. Note that sudo needs to be deleted from the commands in steps 1 to 4.
- If python and its dependency are installed as a non-root user, run the `su - username` command to switch to the non-root user, and then perform steps 1 to 4.

**Step 1** Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

```

gcc --version

gcc-c++ --version

make --version

cmake --version

rpm -qa | grep unzip

rpm -qa | grep zlib-devel

rpm -qa | grep libffi-devel

rpm -qa | grep openssl-devel

rpm -qa | grep pciutils

rpm -qa | grep net-tools

rpm -qa | grep sqlite-devel

rpm -qa | grep blas-devel

rpm -qa | grep lapack-devel

rpm -qa | grep openblas-devel

rpm -qa | grep gcc-gfortran
```

If the following information is displayed, the installation is complete. Go to the next step.

```
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
GNU Make 3.82

cmake version 2.8.12.2

unzip-6.0-21.el7.x86_64

zlib-devel-1.2.7-18.el7.x86_64

libffi-devel-3.0.13-18.el7.x86_64

openssl-devel-1.0.2k-19.el7.x86_64

pciutils-3.5.1-3.el7.x86_64

net-tools-2.0.26.20131004git.el7.x86_64

sqlite-devel-3.7.17-8.el7_7.1x86_64

blas-devel-3.4.2-8.el7.x86_64

lapack-devel-3.4.2-8.el7.x86_64

openblas-devel-0.3.3-2.el7.x86_64

gcc-gfortran-4.8.5-39.el7.x86_64
```

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

```
sudo yum install -y gcc gcc-c++ make cmake unzip zlib-devel libffi-devel openssl-devel pcre-devel net-tools
```

In the preceding steps, the message “No package libopenblas available” is displayed during the installation of openblas-devel. Run the following command to install the Linux extension package of the enterprise edition:

```
sudo yum install epel-release
```

If the CMake version installed using the preceding method is earlier than 3.5.1, see **6.5 Installing CMake 3.5.2.**
Step 2 Check whether the Python development environment is installed.

The deep learning engine package, toolbox, and framework plug-in package depend on the Python environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether they have been installed. If the following information is displayed, they have been installed. Go to the next step.

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:

1. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the installation environment. The command is as follows:
   ```bash
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```
2. Run the following command to go to the download directory and decompress the source code package:
   ```bash
   tar -zxvf Python-3.7.5.tgz
   ```
3. Go to the decompressed folder and run the following configuration, build, and installation commands:
   ```bash
   cd Python-3.7.5
   ./configure --prefix=/usr/local/python3.7.5 --enable-shared
   make
   sudo make install
   ```
   The `--prefix` parameter specifies the Python installation path. You can change it based on the site requirements. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library.

   This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, compilation, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

4. Check whether `libpython3.7m.so.1.0` exists in `/usr/lib64` or `/usr/lib`. If yes, skip this step or back up the `libpython3.7m.so.1.0` file provided by the system and run the following command:
   ```bash
   Copy the compiled file libpython3.7m.so.1.0 to /usr/lib64:
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   If the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.
   ```bash
   cp: overwrite 'libpython3.7m.so.1.0'? 
   ```
   If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.
   ```bash
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```
   Replace the path of the `libpython3.7m.so.1.0` file as required.

5. Run the following commands to set the soft link:
   ```bash
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```
   If a message indicating that the link already exists is displayed during the command execution, run the following command to delete the existing link and run the command again:
   ```bash
   sudo rm -rf /usr/bin/python3.7
   sudo rm -rf /usr/bin/pip3.7
   ```
sudo rm -rf /usr/bin/python3.7
sudo rm -rf /usr/bin/pip3.7

6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

python3.7.5 --version
pip3.7.5 --version

Step 3 Table 3-8 lists the dependencies to be installed. Before installing a software package, ensure that its dependencies have been installed. If you run the following command as a non-root user, add --user to the end of the command. For example:

`pip3.7 install numpy --user`. The installation command can be run in any path.

Table 3-8 Dependency list

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Version</th>
<th>Installation Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
<td>For details, see Step 5 Check whether the Python development environment has been installed.</td>
</tr>
</tbody>
</table>
| numpy      | 1.17.2  | export CFLAGS=-std=c99
            |         | pip3.7 install numpy==1.17.2 |
| decorator  | >=4.4.0 | pip3.7 install decorator |
| sympy      | 1.4     | pip3.7 install sympy==1.4 |
| cffi       | 1.12.3  | pip3.7 install cffi==1.12.3 |
| pyyaml     | -       | pip3.7 install pyyaml |
| pathlib2   | -       | pip3.7 install pathlib2 |
| grpcio     | -       | pip3.7 install grpcio |
| grpcio-tools | -  | pip3.7 install grpcio-tools |
| protobuf   | -       | pip3.7 install protobuf |
| scipy      | -       | pip3.7 install scipy |
| requests   | -       | pip3.7 install requests |

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, rectify the fault by referring to 6.2 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 install Is Run?.

If the error message "subprocess.CalledProcessError: Command "('lsb_release', '-a')" return non-zero exit status 1 " is displayed, rectify the fault by referring to 6.4 The error message "subprocess.CalledProcessError: Command "('lsb_release', '-a')" return non-zero exit status 1 " is displayed during pip3.7.5 installation.
Step 4  CentOS 7.6 uses GCC 4.8.5 by default, which needs to be upgraded to GCC 7.3.0. See 6.6 Installing GCC 7.3.0.

-----End

3.3.3 Debian (x86)

Environment Requirements

The following software or dependencies need to be installed in the development environment: For details, see Installing Dependencies.

Table 3-9 Dependency information

<table>
<thead>
<tr>
<th>Category</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
</tr>
<tr>
<td>cmake</td>
<td>3.5.1+</td>
</tr>
<tr>
<td>gcc</td>
<td>6.3.0</td>
</tr>
<tr>
<td>g++</td>
<td><strong>NOTE</strong> Debian x86 uses GCC 6.3.0 by default, which needs to be upgraded to GCC 7.3.0.</td>
</tr>
<tr>
<td>numpy</td>
<td>≥1.13.3</td>
</tr>
<tr>
<td>decorator</td>
<td>≥4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
</tr>
<tr>
<td>make</td>
<td></td>
</tr>
<tr>
<td>zlib1g-dev</td>
<td></td>
</tr>
<tr>
<td>libbz2-dev</td>
<td></td>
</tr>
<tr>
<td>libsqlite3-dev</td>
<td></td>
</tr>
<tr>
<td>libssl-dev</td>
<td></td>
</tr>
<tr>
<td>libxml2-dev</td>
<td></td>
</tr>
<tr>
<td>libffi-dev</td>
<td></td>
</tr>
<tr>
<td>unzip</td>
<td></td>
</tr>
<tr>
<td>pcutils</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>grpcio</td>
<td></td>
</tr>
<tr>
<td>grpcio-tools</td>
<td></td>
</tr>
<tr>
<td>protobuf</td>
<td></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td></td>
</tr>
</tbody>
</table>

There is no version requirement. The version to be installed is subject to the source provided by the OS.
Check the umask of the root user.

1. Log in to the installation environment as the root user.
2. Check the umask value of the root user.

   umask

3. If the umask value is not 0022, append "umask 0022" to the file and save the file:

   vi ~/.bashrc
   source ~/.bashrc

Creating Installation and Running Users

Table 3-10 lists the users for installing and running the Ascend chip driver, firmware, and CANN software.

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Installation User</th>
<th>Running User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascend chip driver and firmware installation package</td>
<td>The user must be root.</td>
<td>The value must be HwHiAiUser, and HwHiAiUser must be created before the Ascend chip driver and firmware are installed.</td>
</tr>
<tr>
<td>CANN software package</td>
<td>The user can be root or non-root. For details, see the following content below the table.</td>
<td>The user must be a non-root user. For details, see the following content below the table.</td>
</tr>
</tbody>
</table>

You must create the HwHiAiUser user before installing the CANN software as the root user or a non-root user.

1. Create the HwHiAiUser user.

   Switch to the root user and run the following command to create the HwHiAiUser user:

   ```bash
   groupadd HwHiAiUser  # Create the HwHiAiUser user group.
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser  # Create an HwHiAiUser user whose owner group is HwHiAiUser.
   ```

   The user group of the HwHiAiUser user must be HwHiAiUser.

2. Select an installation mode.

   - For installation as the root user:

     You can install the software as the root user, but you must create the HwHiAiUser user. After the CANN software is installed, you must run the software as a non-root user, that is, the running user can only be in the HwHiAiUser user group.
For installation as a non-root user:
In this scenario, the installation and running users must be the same.
- If a non-root user exists, add the user to the HwHiAiUser user group and switch to the user for normal installation. The default installation user and running user are the current user.
- If you want to use a new non-root user, you need to create the user first.

To create a non-root user, run the following commands as the root user:

a. Create a non-root user.
   `useradd -g HwHiAiUser -d /home/username -m username`
   `username` is user-defined, but its owner group must be HwHiAiUser.

b. Set the password of the non-root user.
   `passwd username`

**NOTE**
- The created running user cannot belong to the root user group.
- After the HwHiAiUser user is created, do not disable the login authentication function of the user.
- The password validity period is 90 days. You can change the validity period in the `/etc/login.defs` file or using the `chage` command. For details, see 5.5 Parameters description.

**Checking the Source Validity**

Before installing the deep learning engine package, toolbox, and framework plug-in package, you need to download related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the root user to check whether the source is valid:

`apt-get update`

If an error is reported during command execution or dependency installation, check whether the network connection is normal, or replace the source in the `/etc/apt/sources.list` file with an available source or use an image source. For details about how to configure a network proxy, see 6.7 Configuring a System Network Proxy.

**Configuring Permissions for the Installation User**

If a non-root user is used for installation, perform the operations described in this section. If the root user is used for installation, you only need to perform step 1.

Before installing the development kit, you need to download the dependencies, which require `sudo apt-get` permission. Run the following commands as the root user:

1. Run the following command to install `sudo`:
   `apt-get install sudo`

2. Open the `/etc/sudoers` file:
   `chmod u+w /etc/sudoers`
   `vi /etc/sudoers`
3. Add the following content below # User privilege specification of the file:

```
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/in -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5/bin/pip3 /usr/bin/python3 /usr/bin/python3.7.5, /bin/in -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/in -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/in -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

Replace `username` with the name of the common user who executes the installation script.

**NOTE**

Ensure that the last line of the `/etc/sudoers` file is `#includedir /etc/sudoers.d`. Otherwise, add it manually.

4. Run `:wq!` to save the file.

5. Run the following command to revoke the write permission on the `/etc/sudoers` file:

```
chmod u-w /etc/sudoers
```

Installing Dependencies

**NOTE**

- If you install Python and its dependencies as the root user, perform steps 1 to 4. Note that sudo needs to be deleted from the commands in steps 1 to 4.
- If python and its dependency are installed as a non-root user, run the `su - username` command to switch to the non-root user, and then perform steps 1 to 4.

**Step 1**

Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

```
gcc --version
g++ --version
make --version
cmake --version
dpkg -l zlib1g-dev| grep zlib1g-dev| grep ii
dpkg -l libbz2-dev| grep libbz2-dev| grep ii
dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii
dpkg -l libssl-dev| grep libssl-dev| grep ii
dpkg -l libxslt1-dev| grep libxslt1-dev| grep ii
dpkg -l libffi-dev| grep libffi-dev| grep ii
dpkg -l unzip| grep unzip| grep ii
dpkg -l net-tools| grep net-tools| grep ii
```

If the following information is displayed, the installation is complete. Go to the next step.

```
 gcc (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
 GNU Make 4.1
 cmake version 3.7.2
 zlib1g-dev:amd64 1:1.2.8.dfsg-5 amd64  compression library - development
 libbz2-dev:amd64 1.0.6-8.1 amd64  high-quality block-sorting file compressor library - development
 libsqlite3-dev:amd64 3.16.2-5+deb9u1 amd64  SQLite 3 development files
 libssl-dev:amd64 1.1.1f-1~deb9u1 amd64  Secure Sockets Layer toolkit - development files
 libxslt1-dev:amd64 1.1.29-2.1+deb9u2 amd64  XSLT 1.0 processing library - development kit
 libffi-dev:amd64 3.2.1-6 amd64  Foreign Function Interface library (development files)
 unzip 6.0-21+deb9u2 amd64  De-archiver for .zip files
 pcktutils 1.3.5.2-1 amd64  Linux PCI Utilities
 net-tools 1.60+git20161116.90da8a0-1 amd64  NET-3 networking toolkit
```

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.
sudo apt-get install -y gcc g++ cmake make zlib1g-dev libbz2-dev libsqlite3-dev libssl-dev libxml2-dev libffi-dev unzip pciutils net-tools

Step 2 Check whether the Python development environment is installed.

The development kit depends on the Python environment. Run the **python3.7.5 --version** and **pip3.7.5 --version** commands to check whether Python has been installed. If the following information is displayed, Python has been installed. Go to the next step.

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:

1. Run the **wget** command to download the source code package of Python 3.7.5 to any directory of the installation environment. The command is as follows:
   ```
wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```

2. Run the following command to go to the download directory and decompress the source code package:
   ```
tar -zxvf Python-3.7.5.tgz
   ```

3. Prepare for the installation.
   Check the system time. If the system time is different from the actual time, adjust the system time to be the same as the actual time. The command is as follows:
   ```
date          // Check the system time and date.
date -s "20200620 11:21:30"    // Change the system time to the current time.
hwclock -w       // Write the current time and date to the BIOS to prevent them from becoming invalid after the restart.
```

   In the preceding command, **20200620 11:21:30** is an example of the actual time. Change it based on the actual situation.

4. Go to the decompressed folder and run the following configuration, build, and installation commands:
   ```
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-shared
make
sudo make install
   ```
   The **--prefix** parameter specifies the Python installation path. You can change it based on the site requirements. The **--enable-shared** parameter is used to compile the **libpython3.7m.so.1.0** dynamic library.

   This document uses **--prefix=/usr/local/python3.7.5** as an example. After the configuration, compilation, and installation commands are executed, the installation package is output to the **/usr/local/python3.7.5** directory, and the **libpython3.7m.so.1.0** dynamic library is output to the **/usr/local/python3.7.5/lib/libpython3.7m.so.1.0** directory.

5. Check whether **libpython3.7m.so.1.0** exists in **/usr/lib64** or **/usr/lib**. If yes, skip this step or back up the original file and run the following command:
   ```
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   If the **/usr/lib64** directory does not exist in the environment, copy the file to the **/usr/lib** directory.
   ```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```

   Replace the path of the **libpython3.7m.so.1.0** file as required.
6. Run the following commands to set the soft link:

```bash
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

If a message indicating that the link already exists is displayed during the command execution, run the following command to delete the existing link and run the command again:

```bash
sudo rm -rf /usr/bin/python3.7.5
sudo rm -rf /usr/bin/pip3.7.5
sudo rm -rf /usr/bin/python3.7
sudo rm -rf /usr/bin/pip3.7
```

7. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

```bash
python3.7.5 --version
pip3.7.5 --version
```

### Step 3 Table 3-11

Table 3-11 lists the dependencies to be installed. Before installing a software package, ensure that its dependencies have been installed. If you run the following command as a non-root user, add `--user` to the end of the command. For example:

```bash
pip3.7 install numpy --user
```

The installation command can be run in any path.

**Table 3-11 Dependency list**

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Version</th>
<th>Installation Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
<td>For details, see Step 3 Check whether the Python development environment has been installed.</td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3</td>
<td>pip3.7 install numpy</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
<td>pip3.7 install decorator</td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
<td>pip3.7 install sympy==1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
<td>pip3.7 install cffi==1.12.3</td>
</tr>
<tr>
<td>pyyaml</td>
<td>-</td>
<td>pip3.7 install pyyaml</td>
</tr>
<tr>
<td>pathlib2</td>
<td>-</td>
<td>pip3.7 install pathlib2</td>
</tr>
<tr>
<td>grpcio</td>
<td>-</td>
<td>pip3.7 install grpcio</td>
</tr>
<tr>
<td>grpcio-tools</td>
<td>-</td>
<td>pip3.7 install grpcio-tools</td>
</tr>
<tr>
<td>protobuf</td>
<td>-</td>
<td>pip3.7 install protobuf</td>
</tr>
<tr>
<td>scipy</td>
<td>-</td>
<td>pip3.7 install scipy</td>
</tr>
<tr>
<td>requests</td>
<td>-</td>
<td>pip3.7 install requests</td>
</tr>
</tbody>
</table>

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, rectify
the fault by referring to **6.2 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 install Is Run?**.

If the error message "subprocess.CalledProcessError: Command '('/lsb_release', '-a')' return non-zero exit status 1 " is displayed, rectify the fault by referring to **6.4 The error message "subprocess.CalledProcessError: Command '('/lsb_release', '-a')' return non-zero exit status 1 " is displayed during pip3.7.5 installation.**

**Step 4** Debian x86 uses GCC 6.3.0 by default, which needs to be upgraded to GCC 7.3.0. See **6.6 Installing GCC 7.3.0**.

----End

### 3.4 Pure Development

#### 3.4.1 Installing the Development Kit (Easy Installation Tool)

Use an easy installation tool to install the development environment.

**Prerequisites**

- Prepare for the installation by referring to **3.3 Before You Start**.
- Obtain the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` by referring to **3.2 Obtaining Software Packages**.

**Obtaining Easy Installation Tools**

Download the easy installation tool `ascend-sdk-manager-{version}-arm-{os}.tgz` (for ARM OSs) or `ascend-sdk-manager-{version}-x86-{os}.tgz` (for x86 OSs) and upload the tool to any directory on the server, for example, `/home/HwHiAiUser`.

Run the `tar -xzvf ascend-sdk-manager-{version}-{arch}-{os}.tgz` command to decompress the package. The executable binary file `ascend-sdk-manager` is generated.

**Installation Description**

To use easy installation tools, you must be the root user.

If you use the easy installation to install the software, the software can be installed only in the default directory `/usr/local/Ascend`.

**Procedure**

**Step 1** Log in to the installation environment as the root user.

**Step 2** Upload the obtained development kit to any directory (for example, `/home/install`) in the installation environment.

**Step 3** Run the following command to install the development kit:

```
/home/HwHiAiUser/ascend-sdk-manager app install /home/install dev --mode=auto --silent=y
```
The command format is `{sdk_path}/ascend-sdk-manager app install dir_path scene --mode=auto --silent=y`. The following table describes the parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{sdk_path}</td>
<td>Directory of the binary file <code>ascend-sdk-manager</code>, for example, <code>/home/HwHiAiUser</code>.</td>
</tr>
<tr>
<td>dir_path</td>
<td>Full path of the installation package, for example, <code>/home/install</code>.</td>
</tr>
</tbody>
</table>
| scene       | Installation scenario. The current version supports the following scenarios: `dev` (development environment), `infer` (inference operating environment), and `train` (training operating environment). The following lists the packages that can be installed in different scenarios.  
  ● dev: Ascend-Toolkit  
  ● infer: Ascend-NNRT, Ascend-Toolbox, NPU_Driver, NPU_Firmware  
  ● train: Ascend-NNAE, Ascend-Toolbox, Ascend-TFPlugin, NPU_Driver, NPU_Firmware |
| mode        | The value is `auto`. |
| silent      | Silent installation. Enter `y` or `n` to replace `y` or `n` that the system requires during the uninstallation. |

When the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` is installed using the easy installation tool in the x86 system, the silent installation option can only be `--silent=n`. After the installation is complete, restart the OS for the installation to take effect.

---

### 3.4.2 Installing the Development Kit (CLI)

#### Prerequisites

- Prepare for the installation by referring to 3.3 Before You Start.
- Obtain the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` by referring to 3.2 Obtaining Software Packages.

#### Procedure

**Step 1** Log in to the installation environment as the installation user of the software package.
Ensure that the installation user of the software package is the same as the installation dependency user in 3.3 Before You Start.

**Step 2** Upload the obtained development kit to any directory (for example, `/home`) in the installation environment.

**Step 3** Go to the directory where the software packages are stored.

**Step 4** Grant the execute permission on the software package.

```bash
chmod +x *.run
```

In the preceding command, `*` indicates the development kit name. Replace it with the actual package name.

**Step 5** Run the following command to check the consistency and integrity of the software package installation file:

```
./*.run --check
```

**Step 6** Create a software installation path.

- If you need to specify the installation path, you need to create it first. For example, if the installation path is `/home/work`, run the `mkdir -p /home/work` command to create an installation path and then select the path to install the software.
- If you do not specify an installation path, the software is installed in the default path. For details about the default path, see Table Software package installation path.

### Table 3-12 Software package installation path

<table>
<thead>
<tr>
<th>Item</th>
<th>Path</th>
</tr>
</thead>
</table>
| Default installation path of a software package | - root user: `/usr/local/Ascend/${package_name}/ {version}/(arch-linux-gccversion)`  
|                                   | - Non-root user: `${HOME}/Ascend/${package_name}/ {version}/(arch-linux-gccversion)` |
| Installation log path             | `${install_path}/${package_name}/ {version}/(arch-linux-gccversion)/ascend_install.log`  |
| Path for recording information such as the software package version, CPU architecture, GCC version, and installation path after the installation | `${install_path}/${package_name}/ {version}/(arch-linux-gccversion)/ascend-${package_name}_install.info` |
**Table Variable description** describes the variables in the **Table Software package installation path**.

### Table 3-13 Variable description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>${package_name}$</td>
<td>Software package directory, which is named after the software package.</td>
</tr>
<tr>
<td>${version}$</td>
<td>Version directory, which is named after the version number of the software package.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>Multiple versions can be installed, which are differentiated by version numbers.</td>
</tr>
<tr>
<td>${arch-linux_gcc_version}$</td>
<td>Architecture directory, which is named after the combination of the CPU architecture, Linux branch, and GCC version of the software package.</td>
</tr>
<tr>
<td></td>
<td><strong>NOTE</strong></td>
</tr>
<tr>
<td></td>
<td>Multiple versions can be installed.</td>
</tr>
<tr>
<td>${HOME}$</td>
<td>Current user name.</td>
</tr>
<tr>
<td>${install_path}$</td>
<td>Installation path of the software package.</td>
</tr>
</tbody>
</table>

**Step 7** Install the software.

- If the installation path is specified, run the `./*.run --install --install-path=<path>` command to perform the installation.
  
  In the preceding command, `<path>` indicates the specified installation path, for example, `/home/work`.

- If the installation path is not specified, run the `./*.run --install` command to perform the installation.
The development kits `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` of multiple versions can be installed by different users in the same development environment. The users must be in the same group as the Driver running user. If the owner groups are different, add the users to the group of the Driver running user.

The AICPU operator package in the development kit must be installed as the root user. If you perform the installation as a non-root user, the AICPU operator package (`Ascend910-aicpu_kernels-{version}.run`) will be automatically decompressed to the `${install_path}/ascend-toolkit/{version}/xxx-linux_gccx.x.x` directory during the installation. You need to add `sudo` before the following command or switch to the root user to install the operator package separately. Go to the `${install_path}/ascend-toolkit/{version}/xxx-linux_gccx.x.x` directory and run the following command to perform the installation:

```
./Ascend910-aicpu_kernels-{version}.run --full
```

The AICPU does not support the specified installation path and shares the installation path of the driver.

The `--quiet` option is not supported when the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` is installed in an x86 system.

If the following information is displayed during the installation of the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` in the x86 system, asking you whether to perform a hot reset, enter `n`. After the installation is complete, restart the OS for the setting to take effect. In the current version, only `n` is supported.

The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n]

For more installation modes, see 7.1 Parameters.

If the following information is displayed, the software is successfully installed:

```
[INFO] xxx install success
[INFO] process end
```

`xxx` indicates the name of the software package to be installed.

---End

### 3.5 Development + Commissioning

#### 3.5.1 Installing the Ascend Chip Driver and Firmware

**Step 1** Check the OS kernel version. For details, see "Checking the OS and Environment" in the *Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9010)*.

**Step 2** Install the Ascend chip driver and firmware. For details, see "Installing the Driver and Firmware" in the *Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9010)*.

---End
3.5.2 Installing the Development Kit (Easy Installation Tool)

Use an easy installation tool to install the development environment.

Prerequisites

- Prepare for the installation by referring to 3.3 Before You Start.
- Obtain the development kit Ascend-Toolkit-{version}-xxx-{gcc_version}.run by referring to 3.2 Obtaining Software Packages.

Obtaining Easy Installation Tools

Download the easy installation tool ascend-sdk-manager-{version}-arm-{os}.tgz (for ARM OSs) or ascend-sdk-manager-{version}-x86-{os}.tgz (for x86 OSs) and upload the tool to any directory on the server, for example, /home/HwHiAiUser.

Run the `tar -xzvf ascend-sdk-manager-{version}-{arch}-{os}.tgz` command to decompress the package. The executable binary file ascend-sdk-manager is generated.

Installation Description

To use easy installation tools, you must be the root user.

If you use the easy installation to install the software, the software can be installed only in the default directory /usr/local/Ascend.

Procedure

**Step 1** Log in to the installation environment as the root user.

**Step 2** Upload the obtained development kit to any directory (for example, /home/install) in the installation environment.

**Step 3** Run the following command to install the development kit:

```
/home/HwHiAiUser/ascend-sdk-manager app install /home/install dev --mode=auto --silent=y
```

**NOTE**

The command format is `{sdk_path}/ascend-sdk-manager app install dir_path scene --mode=auto --silent=y`. The following table describes the parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{sdk_path}</td>
<td>Directory of the binary file <code>ascend-sdk-manager</code>, for example, <code>/home/HwHiAiUser</code>.</td>
</tr>
<tr>
<td>dir_path</td>
<td>Full path of the installation package, for example, <code>/home/install</code>.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>scene</td>
<td>Installation scenario. The current version supports the following scenarios: dev (development environment), infer (inference operating environment), and train (training operating environment). The following lists the packages that can be installed in different scenarios.</td>
</tr>
<tr>
<td></td>
<td>● dev: Ascend-Toolkit</td>
</tr>
<tr>
<td></td>
<td>● infer: Ascend-NNRT, Ascend-Toolbox, NPU_Driver, NPU_Firmware</td>
</tr>
<tr>
<td></td>
<td>● train: Ascend-NNAE, Ascend-Toolbox, Ascend-TFPlugin, NPU_Driver, NPU_Firmware</td>
</tr>
<tr>
<td>mode</td>
<td>The value is auto.</td>
</tr>
<tr>
<td>silent</td>
<td>Silent installation. Enter y or n to replace y or n that the system requires during the uninstallation.</td>
</tr>
</tbody>
</table>

**CAUTION**

When the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` is installed using the easy installation tool in the x86 system, the silent installation option can only be `--silent=n`. After the installation is complete, restart the OS for the installation to take effect.

----End

### 3.5.3 Installing the Development Kit (CLI)

**Prerequisites**

- Prepare for the installation by referring to **3.3 Before You Start**.
- Obtain the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` by referring to **3.2 Obtaining Software Packages**.

**Procedure**

**Step 1** Log in to the installation environment as the installation user of the software package.

Ensure that the installation user of the software package is the same as the installation dependency user in **3.3 Before You Start**.

**Step 2** Upload the obtained development kit to any directory (for example, `/home`) in the installation environment.

**Step 3** Go to the directory where the software packages are stored.

**Step 4** Grant the execute permission on the software package.

`chmod +x *.run`
In the preceding command, * indicates the development kit name. Replace it with the actual package name.

**Step 5** Run the following command to check the consistency and integrity of the software package installation file:

```
./*.run --check
```

**Step 6** Create a software installation path.

- If you need to specify the installation path, you need to create it first. For example, if the installation path is `/home/work`, run the `mkdir -p /home/work` command to create an installation path and then select the path to install the software.
- If you do not specify an installation path, the software is installed in the default path. For details about the default path, see Table Software package installation path.

### Table 3-14 Software package installation path

<table>
<thead>
<tr>
<th>Item</th>
<th>Path</th>
</tr>
</thead>
</table>
| Default installation path of a software package | - root user: `/usr/local/Ascend/${package_name}/
{version}/${arch-linux-gccversion}`  
- Non-root user: `${HOME}/Ascend/${package_name}/
{version}/${arch-linux-gccversion}` |
| Installation log path                          | `${install_path}/${package_name}/{version}/${arch-linux-gccversion}/ascend_install.log` |
| Path for recording information such as the software package version, CPU architecture, GCC version, and installation path after the installation | `${install_path}/${package_name}/{version}/${arch-linux-gccversion}/ascend-${package_name}_install.info` |

**Table Variable description** describes the variables in the Table Software package installation path.
### Table 3-15 Variable description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>${\text{package_name}}$</td>
<td>Software package directory, which is named after the software package.</td>
</tr>
</tbody>
</table>
| $\{\text{version}\}$ | Version directory, which is named after the version number of the software package.  
**NOTE**  
Multiple versions can be installed, which are differentiated by version numbers. |
| $\{\text{arch-linux\_gcc\_version}\}$ | Architecture directory, which is named after the combination of the CPU architecture, Linux branch, and GCC version of the software package.  
**NOTE**  
Multiple versions can be installed. |
| ${\text{HOME}}$ | Current user name. |
| $\{\text{install\_path}\}$ | Installation path of the software package. |

### Step 7 Install the software.

- If the installation path is specified, run the `./*.run --install --install-path=<path>` command to perform the installation.
  
  In the preceding command, `<path>` indicates the specified installation path, for example, `/home/work`.

- If the installation path is not specified, run the `./*.run --install` command to perform the installation.
- The development kits `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` of multiple versions can be installed by different users in the same development environment. The users must be in the same group as the Driver running user. If the owner groups are different, add the users to the group of the Driver running user.

- The AICPU operator package in the development kit must be installed as the root user. If you perform the installation as a non-root user, the AICPU operator package (`Ascend910-aicpu_kernels-{version}.run`) will be automatically decompressed to the `$install_path/ascend-toolkit/{version}/xxx-linux_gccx.x.x` directory during the installation. You need to add `sudo` before the following command or switch to the root user to install the operator package separately. Go to the `$install_path/ascend-toolkit/{version}/xxx-linux_gccx.x.x` directory and run the following command to perform the installation:
  ```
  ./Ascend910-aicpu_kernels-{version}.run --full
  ```
  The AICPU does not support the specified installation path and shares the installation path of the driver.

- The `--quiet` option is not supported when the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` is installed in an x86 system.

- If the following information is displayed during the installation of the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` in the x86 system, asking you whether to perform a hot reset, enter `n`. After the installation is complete, restart the OS for the setting to take effect. In the current version, only `n` is supported.
  
  The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n]

  For more installation modes, see 7.1 Parameters.

If the following information is displayed, the software is successfully installed:

```
[INFO] xxx install success
[INFO] process end
```

`xxx` indicates the name of the software package to be installed.

----End

### 3.5.4 Installing the Deep Learning Framework

This document uses TensorFlow as an example to describe how to install the deep learning framework. If you want to use the MindSpore framework, visit [https://www.mindspore.cn/install/en](https://www.mindspore.cn/install/en) to obtain the method of installing the MindSpore framework.

#### Installation Preparations

- For the x86 architecture, skip this step.
- In the AArch64 architecture, TensorFlow depends on H5py, and H5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install H5py. Perform the following operations as the root user:
Step 1 Compile and install the HDF5.

1. Run the `wget` command to download the source code package of HDF5 to any directory of the installation environment. The command is as follows:
   
   `wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz --no-check-certificate`

2. Run the following command to go to the download directory and decompress the source code package:
   
   `tar -zxvf hdf5-1.10.5.tar.gz`

   Go to the decompressed folder and run the following configuration, build, and installation commands:
   
   ```sh
cd hdf5-1.10.5/
./configure --prefix=/usr/include/hdf5
make install
   ```

Step 2 Configure environment variables and create a soft link to the dynamic link library (DLL).

1. Configure environment variables.
   
   ```sh
   export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/"
   ```

2. Run the following command as the root user to create a soft link to the dynamic link library (DLL). Add sudo before the following commands as a non-root user:
   
   ```sh
   ln -s /usr/include/hdf5/lib/libhdf5.so /usr/lib/libhdf5.so
   ln -s /usr/include/hdf5/lib/libhdf5_hl.so /usr/lib/libhdf5_hl.so
   ```

Step 3 Install h5py.

Run the following command as root user to install the h5py dependency:

```sh
pip3.7 install Cython
```

The h5py installation command is as follows:

```sh
pip3.7 install h5py==2.8.0
```

---End

Installing TensorFlow 1.15.0

TensorFlow 1.15 is necessary to develop and verify operators and develop training services.

- For the x86 architecture, download the software package from the pip source. For details, see https://www.tensorflow.org/install/pip?lang=python3.
  
  Note that the instructions provided by the TensorFlow website are incorrect. To download the CPU version from the pip source, you need to explicitly specify `tensorflow-cpu`. Otherwise, the GPU version is downloaded by default. That is, change `tensorflow==1.15 --Release for CPU-only` to `tensorflow-gpu==1.15 --Release for CPU-only`. In addition, the installation command `pip3 install --user --upgrade tensorflow` described on the official website needs to be changed to `pip3.7 install Tensorflow-cpu==1.15` as the root user and to `pip3.7 install Tensorflow-cpu==1.15 --user` as a non-root user.

- For the AArch64 architecture, the pip source does not provide the corresponding version. Therefore, you need to use GCC 7.3.0 to compile TensorFlow 1.15.0. For details about the compilation procedure, see https://www.tensorflow.org/install/source. Pay attention to the following points:

After downloading the `tensorflow tag v1.15.0` source code, perform the following steps:
**Step 1** Download the `nsync-1.22.0.tar.gz` source code package.

1. Go to the `tensorflow` tag v1.15.0 source code directory, open the `tensorflow/workspace.bzl` file, and find the definition of `tf_http_archive` whose `name` is `nsync`.

   ```python
   tf_http_archive(
       name = "nsync",
       sha256 = "caf32e6b3d478b78cff6c2ba009c3400f8251f6468044cb65465666a9cea93c4",
       strip_prefix = "nsync-1.22.0",
       system_build_file = clean_dep("//third_party/systemlibs:nsync.BUILD"),
       urls = ["https://storage.googleapis.com/mirror.tensorflow.org/github.com/google/nsync/archive/1.22.0.tar.gz",
               "https://github.com/google/nsync/archive/1.22.0.tar.gz"],
   )
   ```

2. Download the `nsync-1.22.0.tar.gz` source code package from any path in `urls` and save it to any path.

**Step 2** Modify the `nsync-1.22.0.tar.gz` source code package:

1. Go to the directory where `nsync-1.22.0.tar.gz` is stored and decompress the source code package. Find the decompressed `nsync-1.22.0` folder and the `pax_global_header` file.

2. Edit the `nsync-1.22.0/platform/c++11/atomic.h` file. Append the following information in red to `NSYNC_CPP_START_`.

   ```c
   #include "nsync_cpp.h"
   #include "nsync_atomic.h"
   
   NSYNC_CPP_START_
   
   #define ATM_CB_() __sync_synchronize()
   static INLINE int atm_cas_nomb_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
       int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_relaxed, std::memory_order_relaxed));
       ATM_CB_();
       return result;
   }
   static INLINE int atm_cas_acq_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
       int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_acquire, std::memory_order_relaxed));
       ATM_CB_();
       return result;
   }
   static INLINE int atm_cas_rel_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
       int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_release, std::memory_order_relaxed));
       ATM_CB_();
       return result;
   }
   static INLINE int atm_cas_relacq_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
       int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_acquire, std::memory_order_relaxed));
       ATM_CB_();
       return result;
   }
   ```

**Step 3** Compress the `nsync-1.22.0.tar.gz` source code package.

Compress the modified `nsync-1.22.0` folder and `pax_global_header` into a new `nsync-1.22.0.tar.gz` source code package (for example, `/tmp/nsync-1.22.0.tar.gz`).

**Step 4** Generate a sha256sum verification code for the `nsync-1.22.0.tar.gz` source code package.
sha256sum /tmp/nsync-1.22.0.tar.gz

Obtain the sha256sum verification code.

**Step 5** Change the **sha256sum** verification code and **urls**.

Go to the tensorflow tag v1.15.0 source code directory, open the tensorflow/workspace.bzl file, find the definition of tf_http_archive whose name is nsync, enter the verification code obtained in **Step 4** after sha256=, and enter the first line of the list after urls=, enter the file:// index for storing the nsync-1.22.0.tar.gz file.

```python
tf_http_archive(
    name = "nsync",
    sha256 = "caf32e6b3d478b78cf6c2ba009c3400f8251f646804bcb65465666a9cea93c4",
    strip_prefix = "nsync-1.22.0",
    system_build_file = clean_dep("third_party/systemlibs/nsync.BUILD"),
    urls = [
        "file:///tmp/nsync-1.22.0.tar.gz",
        "https://storage.googleapis.com/mirror.tensorflow.org/github.com/google/nsync/archive/1.22.0.tar.gz",
        "https://github.com/google/nsync/archive/1.22.0.tar.gz",
    ],
)
```

**Step 6** Continue to perform compilation from the official configuration build (https://www.tensorflow.org/install/source).

After the **./configure** command is executed, add the following build option to the **.tf_configure.bazelrc** configuration file.

```bash
build:opt --cxxopt=-D_GLIBCXX_USE_CXX11_ABI=0
```

Delete the following two lines.

```bash
build:opt --copt=-march=native
build:opt --host_copt=-march=native
```

**Step 7** Proceed with the official compilation procedure (https://www.tensorflow.org/install/source).

**Step 8** Install the compiled TensorFlow.

After the preceding steps are complete, the TensorFlow is packaged to the specified directory. Go to the specified directory and run the following command to install TensorFlow1.15 as the **root** user:

```bash
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
```

Run the following commands as a non-root user:

```bash
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
```

----End

### 3.5.5 Installing protobuf Python

If the training script depends on the Python version of protobuf to store data in the serialized structure (for example, the serialization interfaces of TensorFlow), you need to install protobuf Python. The following uses EulerOS as an example to describe the installation procedure.

**Step 1** Check whether the system contains the dynamic library /usr/local/python3.7.5/lib/python3.7/site-packages/google/protobuf/pyext/
message.cpython-37m-<arch>-linux-gnu.so. If not, perform the following steps to install it:

NOTE

/usr/local/python3.7.5/lib/python3.7/site-packages is the path for installing third-party libraries using pip. You can run the pip3.7 -V command to query the path.

If the system displays /usr/local/python3.7.5/lib/python3.7/site-packages/pip, the path for installing third-party libraries using pip is /usr/local/python3.7.5/lib/python3.7/site-packages.

arch indicates the system architecture type, which can also be x86_64.

Step 2 Uninstall protobuf.

pip3.7 uninstall protobuf

Step 3 Download the protobuf software package.

Download the protobuf-python-3.11.3.tar.gz software package (or another version that is compatible with TensorFlow installed in the current environment) from https://github.com/protocolbuffers/protobuf/releases/, upload the package to any directory on the Linux server as the root user, and decompress the package.

Step 4 Install protobuf as the root user.

Go to the protobuf software package directory.

1. Install the dependencies of protobuf.

   If the OS is Ubuntu, run the following command:
   apt-get install autoconf automake libtool curl make g++ unzip libffi-dev -y
   If the OS is CentOS or BClinux, run the following command:
   yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel -y

   2. Grant the execute permission on the autogen.sh script and execute the script.
      chmod +x autogen.sh
      ./autogen.sh

   3. Configure the installation path.
      ./configure
      To specify the installation path, run the following command:
      ./configure --prefix=/protobuf
      protobuf indicates the installation path specified by the user.

   4. Run the following commands to install protobuf:
      make
      make install

      NOTE
      Build using the make command could be time consuming. You can run the make -jx command to perform parallel build based on the number of CPU cores. x indicates the number of parallel tasks. For example, if there are two CPU cores, run the make -j4 command.

   5. Refresh the shared library.
      ldconfig

      After the protobuf is installed, the google/protobuf folder is generated in the include directory in Step 4.3. This folder stores the header files related to the
protobuf. Generate the protoc executable file in the bin directory in Step 4.3. The protoc executable file is used to compile the *.proto file and generate the C++ header file and implementation file of the protobuf.

6. Check whether the software package is successfully installed.

```bash
ln -s /protobuf/bin/protoc /usr/bin/protoc
protoc --version
```

/protobuf is the installation path configured by the user in Step 4.3. If the installation path is not configured, run the protoc --version command to check whether the installation is successful.

**Step 5** Install the runtime library of protobuf Python.

1. Go to the python subdirectory in the protobuf software package directory and compile the Python runtime library.

```bash
python3.7 setup.py build --cpp_implementation
```

**NOTE**

Build a runtime library of the binary version. If the runtime library of the binary version cannot be generated by running the python3.7 setup.py build command, the processing of the serialization structure is slow.

2. Install the dynamic library.

```bash
cd .. && make install
```

Go to the python subdirectory and install the Python runtime library.

```bash
python3.7 setup.py install --cpp_implementation
```

3. Check whether the library has been installed.

Check whether the dynamic library 

```
/usr/local/python3.7.5/lib/python3.7/site-packages/protobuf-3.11.3-py3.7-linux-aarch64.egg/google/protobuf/pyext/_message.cpython-37m-<arch>-linux-gnu.so
```

exists in the system.

**NOTE**

```
/usr/local/python3.7.5/lib/python3.7/site-packages
```

is the path for installing third-party libraries using pip. You can run the `pip3.7 -V` command to query the path.

If the system displays `/usr/local/python3.7.5/lib/python3.7/site-packages/pip`, the path for installing third-party libraries using pip is `/usr/local/python3.7.5/lib/python3.7/site-packages`.

`arch` indicates the system architecture type, which can also be `x86_64`.

4. Add the following environment variable setting to the startup script:

```bash
export LD_LIBRARY_PATH=/protobuf/lib
```

In the preceding command, `/protobuf` indicates the installation path set by the user in Step 4.3. The default installation path is `/usr/local`.

5. Establish a soft link.

If you configure the installation path, you need to establish a soft link. Otherwise, an error will be reported when TensorFlow is imported. The command is as follows:

```bash
ln -s /protobuf/lib/libprotobuf.so.22.0.3 /usr/lib/libprotobuf.so.22
```

/protobuf is the installation path configured by the user in Step 4.3.

----End
3.5.6 Changing NPU IP Addresses

Configuring NIC IP Address of a Device

- **SMP (symmetric multi-processor) mode**
  
  Log in to the AI Servers as the root user and configure the NIC IP address of each device. The configuration requirements are as follows:
  
  - NICs 0 and 4, 1 and 5, 2 and 6, and 3 and 7 of an AI Server must be in the same network segment respectively. NICs 0, 1, 2, and 3 must be in different network segments. NICs 4, 5, 6, and 7 must be in different network segments.
  
  - In the cluster scenario, the devices in the similar positions on AI Servers must be in the same network segment. For example, NIC 0 of AI Server 1 and AI Server 2 must be in the same network segment, and NIC 1 of AI Server 1 and AI Server 2 must be in the same network segment.

  ```
  hccn_tool -i 0 -ip -s address 192.168.0.2 netmask 255.255.255.0
  hccn_tool -i 1 -ip -s address 192.168.1.2 netmask 255.255.255.0
  hccn_tool -i 2 -ip -s address 192.168.2.2 netmask 255.255.255.0
  hccn_tool -i 3 -ip -s address 192.168.3.2 netmask 255.255.255.0
  hccn_tool -i 4 -ip -s address 192.168.0.3 netmask 255.255.255.0
  hccn_tool -i 5 -ip -s address 192.168.1.3 netmask 255.255.255.0
  hccn_tool -i 6 -ip -s address 192.168.2.3 netmask 255.255.255.0
  hccn_tool -i 7 -ip -s address 192.168.3.3 netmask 255.255.255.0
  ```

- **AMP (asymmetric multi-processor) mode**
  
  You do not need to restrict network segments. All NICs must be in the same network segment.

3.6 Verifying the Installation

You can use the Ascend-DMI tool to check the compatibility between the device health information and software and hardware. After configuring the environment variables, you can use the tool in any directory. You can use the tool as the root user or a non-root user. If you use the tool as a non-root user, perform the following steps to add the HwHiAiUser user group:

1. Log in to the server as the root user.
2. Run the `usermod -a -G HwHiAiUser {username}` command to add the user to the HwHiAiUser user group. `{username}` indicates the name of the non-root user. Replace it with the actual user name. The procedure is as follows:

To verify the installation, perform the following steps:

**Step 1** Add environment variables.

1. Log in to the server as the root user.
2. Run the `vi ~/.bashrc` command.
3. Add environment variables in **Table 3-16** to the `.bashrc` file based on the application scenario.
### Table 3-16 Environment Variables

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Environment Variables</th>
</tr>
</thead>
</table>
| **Training** (operating environment) | toolbox_install_path=/usr/local/Ascend/toolbox/{version}/xxx-{gcc-version} # Replace it with the actual installation path. {version} indicates the software package version, and xxx-{gcc-version} indicates the OS architecture and GCC version.  
nnae_install_path=/usr/local/Ascend/nnae/{version}/xxx-{gcc-version} # Replace it with the actual installation path. {version} indicates the software package version, and xxx-{gcc-version} indicates the OS architecture and GCC version.  
export PATH=${toolbox_install_path}/dmi/bin:${PATH}  
export LD_LIBRARY_PATH=/usr/local/dcmi:/usr/local/Ascend/add-ons:${toolbox_install_path}/dmi/lib64:${toolbox_install_path}/fwkacllib/lib64:/usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/driver:${LD_LIBRARY_PATH} |
| **Training** (development environment) | toolkit_install_path=/usr/local/Ascend/ascend-toolkit/{version}/xxx-{gcc-version} # Replace it with the actual installation path. {version} indicates the software package version.  
export PATH=${toolkit_install_path}/dmi/bin:${PATH}  
export LD_LIBRARY_PATH=/usr/local/dcmi:/usr/local/Ascend/add-ons:${toolkit_install_path}/dmi/lib64:${toolkit_install_path}/fwkacllib/lib64:/usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/driver:${LD_LIBRARY_PATH} |

4. Run `source .bashrc` to make the environment variables take effect.

**Step 2** Check the device health status.

1. Run the `ascend-dmi info` command and obtain the card number from the **Card** parameter in the displayed table, as shown in the red box in **Figure 3-1**.
2. Run the `ascend-dmi -dg -c {card-number} -l 1` command to query the health status. `{card-number}` indicates the card number. Replace it as required.

For details about how to check the device health status, see 7.4 Performing Fault Diagnostics.

Step 3  Check the software and hardware compatibility.

- If you use the default path when installing the software package:
  Run the `ascend-dmi -c` command to check the software and hardware compatibility.

- You need to set this parameter if the default installation path is not used. For example, if the software package is installed in the `/home/xxx/Ascend` directory, run the following command:

  `ascend-dmi -c -p /home/xxx/Ascend`

For details about how to check software and hardware compatibility, see 7.5 Performing a Software and Hardware Compatibility Test.

----End
Installing the Operating Environment

4.1 Before You Start
4.2 Obtaining Software Packages
4.3 Installation on a Physical Machine
4.4 Installation in a Container

4.1 Before You Start

The operating environment is the actual environment for model training. The Ascend AI device (such as the training server or training card) must be configured in the installation environment.

The running environment can be deployed on physical machines and containers. You can use easy installation tools or commands to install the operating environment on a physical machine.

The easy installation tool allows you to install software required by a specific service scenario by one click. For example, you can install the training operating environment and install Ascend chip drivers, firmware, deep learning engines, framework plug-ins, and utility kit at the same time by one click. Therefore, the easy installation tool is recommended for you to install the operating environment on a physical machine to improve the installation efficiency.
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Installation Mode</th>
<th>Installation Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical machine</td>
<td>Easy installation</td>
<td>1. 4.2 Obtaining Software Packages</td>
</tr>
<tr>
<td>scenario</td>
<td>tool</td>
<td>2. 4.3.1 Before You Start</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. 4.3.2 Installing the Ascend Chip Driver Firmware and Training Software (Easy Installation Tool)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. 4.3.4 Installing the Deep Learning Framework</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. 4.3.5 Installing protobuf Python</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. 4.3.6 Changing NPU IP Addresses</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4.3.3 Installing the Ascend Chip Driver Firmware and Training Software (CLI)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. 4.3.4 Installing the Deep Learning Framework</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. 4.3.5 Installing protobuf Python</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. 4.3.6 Changing NPU IP Addresses</td>
</tr>
<tr>
<td>Container</td>
<td>CLI</td>
<td>1. 4.2 Obtaining Software Packages</td>
</tr>
<tr>
<td>scenario</td>
<td></td>
<td>2. 4.4.1 Installing the Ascend Chip Driver and Firmware on the Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. 4.4.2 Installing the Toolbox on the Host</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. 4.4.3 Installing Ascend Docker on the Host Machine</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5. 4.4.4 Creating a Container Image Using a Docker File</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6. 4.4.5 Deploying a Training Container</td>
</tr>
</tbody>
</table>

### 4.2 Obtaining Software Packages

To obtain the software package, perform the following steps:

**Step 1** Log in to the **Resources** page and select the corresponding product series and product model.

**Step 2** Obtain the software packages listed in **Table 4-1**.
<table>
<thead>
<tr>
<th>Name</th>
<th>Software Package</th>
<th>OS Architecture</th>
<th>Deployment Scenario</th>
<th>Installation User</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascend chip driver</td>
<td>A800-9010-NPU_Driver-{version}-X86_64-Ubuntu18.04.run</td>
<td>Ubuntu X86</td>
<td>Physical machine</td>
<td>root</td>
<td>NPU driver</td>
</tr>
<tr>
<td>installation package</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A800-9010-NPU_Driver-{version}-X86_64-CentOS7.6_gcc7.3.0.run</td>
<td>CentOS X86</td>
<td></td>
<td>HwHiAi User</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A800-9010-NPU_Driver-{version}-X86_64-Debian9.9.run</td>
<td>Debian X86</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ascend chip firmware</td>
<td>A800-9010-NPU_Firmware-{version}.run</td>
<td>-</td>
<td></td>
<td></td>
<td>NPU firmware</td>
</tr>
<tr>
<td>installation package</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deep learning acceleration</td>
<td>Ascend-NNAE-{version}-x86_64-linux_gcc7.3.0.run</td>
<td>CentOS X86</td>
<td>Physical machine</td>
<td>root or non-root</td>
<td>Contains the FWK library Fwklib and the operator library OPP component, which are used to train models.</td>
</tr>
<tr>
<td>engine package</td>
<td></td>
<td>Ubuntu X86</td>
<td>or container</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Debian X86</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Software Package</td>
<td>OS Architecture</td>
<td>Deployment Scenario</td>
<td>Installation User</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------------------------------------</td>
<td>--------------------------</td>
<td>---------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Tool box</td>
<td>Ascend-Toolbox-{version}-x86_64-linux_gcc7.3.0.run</td>
<td>CentOS X86, Ubuntu X86, Debian X86</td>
<td>Physical machine</td>
<td>the same as that of the Ascend chip driver operating user HwHiAI User. If you use the easy installation tool to install the software, perform operation as the root user.</td>
<td>Contains AICPU operators, which are used for inference model calling.</td>
</tr>
<tr>
<td>Framework plug-in package</td>
<td>Ascend-TFPlugin-{version}-x86_64-linux_gcc7.3.0.run</td>
<td>CentOS X86, Ubuntu X86, Debian X86</td>
<td>Physical machine or container</td>
<td></td>
<td>Contains the framework adaptation plug-in, which is used to connect to the upper-layer framework, for example, the adaptation plug-in of TensorFlow.</td>
</tr>
<tr>
<td>Easy installation tool</td>
<td>ascend-sdk-manager-{version}-x86-{os}.tgz</td>
<td>Ubuntu X86, CentOS X86, Debian X86</td>
<td>Physical machine</td>
<td>root</td>
<td>Use an easy installation tool to install the operating environment and development environment.</td>
</tr>
<tr>
<td>AscendDocker</td>
<td>ascend-docker-runtime-{version}.x86_64.rpm</td>
<td>CentOS X86</td>
<td>Physical machine</td>
<td>root</td>
<td>The container engine plug-in Ascend Docker provides Ascend NPU-based containerization support for all AI training jobs so that AI jobs can run smoothly on Ascend devices as Docker containers.</td>
</tr>
<tr>
<td></td>
<td>ascend-docker-runtime_{version}.x86_64.deb</td>
<td>Ubuntu X86, Debian X86</td>
<td>Physical machine</td>
<td>root</td>
<td></td>
</tr>
</tbody>
</table>
4.3 Installation on a Physical Machine

4.3.1 Before You Start

4.3.1.1 Ubuntu (x86)

Environment Requirements

The following software or dependencies need to be installed in the operating environment: For details, see Installing Dependencies.

<table>
<thead>
<tr>
<th>Category</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
</tr>
<tr>
<td>cmake</td>
<td>3.5.1+</td>
</tr>
<tr>
<td>gcc</td>
<td>7.3.0 or later</td>
</tr>
<tr>
<td>g++</td>
<td>7.3.0 or later</td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
</tr>
<tr>
<td>Category</td>
<td>Version Restriction</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>make</td>
<td>There is no version requirement. The version to be installed is subject to the source provided by the OS.</td>
</tr>
<tr>
<td>zlib1g</td>
<td></td>
</tr>
<tr>
<td>zlib1g-dev</td>
<td></td>
</tr>
<tr>
<td>libbz2-dev</td>
<td></td>
</tr>
<tr>
<td>libsqlite3-dev</td>
<td></td>
</tr>
<tr>
<td>libssl-dev</td>
<td></td>
</tr>
<tr>
<td>libxslt1-dev</td>
<td></td>
</tr>
<tr>
<td>libffi-dev</td>
<td></td>
</tr>
<tr>
<td>unzip</td>
<td></td>
</tr>
<tr>
<td>pcoutils</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>grpcio</td>
<td></td>
</tr>
<tr>
<td>grpcio-tools</td>
<td></td>
</tr>
<tr>
<td>protobuf</td>
<td></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td></td>
</tr>
</tbody>
</table>

Check the umask of the root user.

1. Log in to the installation environment as the root user.
2. Check the umask value of the root user.

   ```bash
   umask
   ```
3. If the umask value is not 0022, append "umask 0022" to the file and save the file:

   ```bash
   vi ~/.bashrc
   source ~/.bashrc
   ```

Creating Installation and Running Users

Table 4-3 lists the users for installing and running the Ascend chip driver, firmware, and CANN software.
### Table 4-3 Installation and running user list

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Installation User</th>
<th>Running User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascend chip driver and</td>
<td>The user must be <strong>root</strong>.</td>
<td>The value must be <strong>HwHiAiUser</strong>, and <strong>HwHiAiUser</strong> must be created before</td>
</tr>
<tr>
<td>firmware installation package</td>
<td></td>
<td>the Ascend chip driver and firmware are installed.</td>
</tr>
<tr>
<td>CANN software package</td>
<td>The user can be <strong>root</strong> or non-root. For details, see</td>
<td>The user must be a non-root user. For details, see the following content</td>
</tr>
<tr>
<td></td>
<td>the following content below the table.</td>
<td>below the table.</td>
</tr>
</tbody>
</table>

You must create the **HwHiAiUser** user before installing the CANN software as the **root** user or a non-root user.

1. Create the **HwHiAiUser** user.
   Switch to the **root** user and run the following command to create the **HwHiAiUser** user:
   ```bash
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser
   ```
   The user group of the **HwHiAiUser** user must be **HwHiAiUser**.

2. Select an installation mode.
   - **For installation as the root user:**
     You can install the software as the **root** user, but you must create the **HwHiAiUser** user. After the CANN software is installed, you must run the software as a non-root user, that is, the running user can only be in the **HwHiAiUser** user group.
   - **For installation as a non-root user:**
     In this scenario, the installation and running users must be the same.
     - If a non-root user exists, add the user to the **HwHiAiUser** user group and switch to the user for normal installation. The default installation user and running user are the current user.
     - If you want to use a new non-root user, you need to create the user first.

To create a non-root user, run the following commands as the **root** user:

a. Create a non-root user.
   ```bash
   useradd -g HwHiAiUser -d /home/username -m username
   ```
   **username** is user-defined, but its owner group must be **HwHiAiUser**.

b. Set the password of the non-root user.
   ```bash
   passwd username
   ```
The created running user cannot belong to the root user group.

After the HwHiAiUser user is created, do not disable the login authentication function of the user.

The password validity period is 90 days. You can change the validity period in the /etc/login.defs file or using the chage command. For details, see 5.5 Parameters description.

Checking the Source Validity

Before installing the deep learning engine package, toolbox, and framework plug-in package, you need to download related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the root user to check whether the source is valid:

```
apt-get update
```

If an error is reported during command execution or dependency installation, check whether the network connection is normal, or replace the source in the /etc/apt/sources.list file with an available source or use an image source. For details about how to configure a network proxy, see 6.7 Configuring a System Network Proxy.

Configuring Permissions for the Installation User

If a non-root user is used for installation, perform the operations described in this section. If the root user is used for installation, you only need to perform step 1.

Before installing the deep learning engine package, toolbox, and framework plug-in package, you need to download the dependencies using the sudo apt-get permission. Perform the following operations as the root user:

1. Run the following command to install sudo:
```
apt-get install sudo
```

2. Open the /etc/sudoers file:
```
chmod u+w /etc/sudoers
vi /etc/sudoers
```

3. Add the following content below # User privilege specification of the file:

```
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

Replace username with the name of the common user who executes the installation script.

**NOTE**

Ensure that the last line of the /etc/sudoers file is #includedir /etc/sudoers.d. Otherwise, add it manually.

4. Run :wq! to save the file.

5. Run the following command to revoke the write permission on the /etc/sudoers file:
```
chmod u-w /etc/sudoers
```
Installing Dependencies

**NOTE**

- If you install Python and its dependencies as the root user, perform steps 1 to 3. Note that sudo needs to be deleted from the commands in steps 1 to 3.
- If python and its dependency are installed as a non-root user, run the `su - username` command to switch to the non-root user, and then perform steps 1 to 3.

**Step 1** Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

```bash
gcc --version
make --version
cmake --version
dpkg -l zlib1g| grep zlib1g| grep ii
dpkg -l zlib1g-dev| grep zlib1g-dev| grep ii
dpkg -l libbz2-dev| grep libbz2-dev| grep ii
dpkg -l libsllite3-dev| grep libsllite3-dev| grep ii
dpkg -l openssl| grep openssl| grep ii
dpkg -l libssl-dev| grep libssl-dev| grep ii
dpkg -l libxslt1-dev| grep libxslt1-dev| grep ii
dpkg -l libffi-dev| grep libffi-dev| grep ii
dpkg -l unzip| grep unzip| grep ii
dpkg -l pciutils| grep pciutils| grep ii
```

If the following information is displayed, the installation is complete. Go to the next step.

```
gcc (Ubuntu/Linaro 7.5.0-3ubuntu1~18.04) 7.5.0
GNU Make 4.1
cmake version 3.10.2
zlib1g:amd64   1:1.2.11.dfsg-0ubuntu2 amd64        compression library - runtime
zlib1g-dev:amd64  1:1.2.11.dfsg-0ubuntu2  amd64        compression library - development
libbz2-dev:amd64    1.0.6-8.1ubuntu0.2 amd64        high-quality block-sorting file compressor library - development
libsllite3-dev:amd64  3.22.0-1ubuntu0.3 amd64        SQLite 3 development files
openssl        1.1.1-1ubuntu2.1-18.04.6 amd64        Secure Sockets Layer toolkit - cryptographic utility
libssl-dev:amd64    1.1.1-1ubuntu2.1-18.04.6 amd64        Secure Sockets Layer toolkit - development files
libssl1-dev:amd64  1.1.29-Sububuntu0.2  amd64        XSLT 1.0 processing library - development kit
libffi-dev:amd64  3.2.1-8          amd64        Foreign Function Interface library (development files)
unzip          6.0-21ubuntu1 amd64        De-archiver for .zip files
pciutils       1.3.5.2-1ubuntu1 amd64        Linux PCI Utilities
net-tools      1.60+git20161116.90da8a0-1ubuntu1 amd64        NET-3 networking toolkit
```

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

```
sudo apt-get install gcc make cmake zlib1g zlib1g-dev libbz2-dev libssl-dev libsllite3-dev libssl-dev libssl1-dev libffi-dev unzip pciutils net-tools -y
```

**Step 2** Check whether the Python development environment is installed.

The development kit depends on the Python environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether Python has been installed. If the following information is displayed, Python has been installed. Go to the next step.

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:
1. Run the **wget** command to download the source code package of Python 3.7.5 to any directory of the installation environment. The command is as follows:
   
   ```bash
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```

2. Run the following command to go to the download directory and decompress the source code package:
   
   ```bash
   tar -zxvf Python-3.7.5.tgz
   ```

3. Go to the decompressed folder and run the following configuration, build, and installation commands:
   
   ```bash
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5
--enable-shared
make
sudo make install
```

   The **--prefix** parameter specifies the Python installation path. You can change it based on the site requirements. The **--enable-shared** parameter is used to compile the **libpython3.7m.so.1.0** dynamic library.

   This document uses **--prefix=/usr/local/python3.7.5** as an example. After the configuration, compilation, and installation commands are executed, the installation package is output to the **/usr/local/python3.7.5** directory, and the **libpython3.7m.so.1.0** dynamic library is output to the **/usr/local/python3.7.5/lib/libpython3.7m.so.1.0** directory.

4. Check whether **libpython3.7m.so.1.0** exists in **/usr/lib64** or **/usr/lib**. If yes, skip this step or back up the **libpython3.7m.so.1.0** file provided by the system and run the following command:

   ```bash
   Copy the compiled file libpython3.7m.so.1.0 to /usr/lib64:
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```

   If the following information is displayed, enter **y** to overwrite the **libpython3.7m.so.1.0** file provided by the system.

   ```bash
   cp: overwrite 'libpython3.7m.so.1.0'? [y/n]
   ```

   If the **/usr/lib64** directory does not exist in the environment, copy the file to the **/usr/lib** directory.

   ```bash
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```

   Replace the path of the **libpython3.7m.so.1.0** file as required.

5. Run the following commands to set the soft link:

   ```bash
   sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
   sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
   ```

   If a message indicating that the link already exists is displayed during the command execution, run the following command to delete the existing link and run the command again:

   ```bash
   sudo rm -rf /usr/bin/python3.7.5
   sudo rm -rf /usr/bin/pip3.7.5
   ```

6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

   ```bash
   python3.7.5 --version
   pip3.7.5 --version
   ```

**Step 3** Table 4-4 lists the dependencies to be installed. Before installing a software package, ensure that its dependencies have been installed. If you run the following
command as a non-root user, add \texttt{--user} to the end of the command. For example:

\texttt{pip3.7 install numpy --user}. The installation command can be run in any path.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
\textbf{Dependency} & \textbf{Version} & \textbf{Installation Command} \\
\hline
Python & 3.7.5 & For details, see \texttt{Step 5 Check whether the Python development environment has been installed.} \\
\hline
numpy & $\geq 1.13.3$ & \texttt{pip3.7 install numpy} \\
\hline
decorator & $\geq 4.4.0$ & \texttt{pip3.7 install decorator} \\
\hline
sympy & 1.4 & \texttt{pip3.7 install sympy=1.4} \\
\hline
cffi & 1.12.3 & \texttt{pip3.7 install cffi=1.12.3} \\
\hline
pyyaml & - & \texttt{pip3.7 install pyyaml} \\
\hline
pathlib2 & - & \texttt{pip3.7 install pathlib2} \\
\hline
grpcio & - & \texttt{pip3.7 install grpcio} \\
\hline
grpcio-tools & - & \texttt{pip3.7 install grpcio-tools} \\
\hline
protobuf & - & \texttt{pip3.7 install protobuf} \\
\hline
scipy & - & \texttt{pip3.7 install scipy} \\
\hline
requests & - & \texttt{pip3.7 install requests} \\
\hline
\end{tabular}
\end{table}

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, rectify the fault by referring to \texttt{6.2 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 install Is Run?}.

If the error message "subprocess.CalledProcessError: Command 'lsb_release', '-a')' return non-zero exit status 1" is displayed, rectify the fault by referring to \texttt{6.4 The error message "subprocess.CalledProcessError: Command 'lsb_release', '-a')' return non-zero exit status 1" is displayed during pip3.7.5 installation.}.

-----End

4.3.1.2 CentOS (x86)

Environment Requirements

The following software or dependencies need to be installed in the operating environment: For details, see \texttt{Installing Dependencies}.
**Table 4-5 Dependency information**

<table>
<thead>
<tr>
<th>Category</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
</tr>
<tr>
<td>cmake</td>
<td>3.5.1+</td>
</tr>
<tr>
<td>gcc</td>
<td>4.8.5</td>
</tr>
<tr>
<td>g++</td>
<td>NOTE: CentOS 7.6 uses GCC 4.8.5 by default, which needs to be upgraded to GCC 7.3.0.</td>
</tr>
<tr>
<td>numpy</td>
<td>1.17.2</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
</tr>
<tr>
<td>make</td>
<td></td>
</tr>
<tr>
<td>unzip</td>
<td></td>
</tr>
<tr>
<td>zlib-devel</td>
<td></td>
</tr>
<tr>
<td>libffi-devel</td>
<td></td>
</tr>
<tr>
<td>openssl-devel</td>
<td></td>
</tr>
<tr>
<td>pcutils</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>sqlite-devel</td>
<td></td>
</tr>
<tr>
<td>blas-devel</td>
<td></td>
</tr>
<tr>
<td>lapack-devel</td>
<td></td>
</tr>
<tr>
<td>openblas-devel</td>
<td></td>
</tr>
<tr>
<td>gcc-gfortran</td>
<td></td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>grpcio</td>
<td></td>
</tr>
<tr>
<td>grpcio-tools</td>
<td></td>
</tr>
<tr>
<td>protobuf</td>
<td></td>
</tr>
<tr>
<td>Scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td></td>
</tr>
<tr>
<td></td>
<td>There is no version requirement. The version to be installed is subject to the source provided by the OS.</td>
</tr>
</tbody>
</table>

**Check the umask of the root user.**

1. Log in to the installation environment as the root user.
2. Check the umask value of the root user.
   ```bash
   umask
   ```
3. If the umask value is not 0022, append "umask 0022" to the file and save the file:
Creating Installation and Running Users

Table 4-6 lists the users for installing and running the Ascend chip driver, firmware, and CANN software.

Table 4-6 Installation and running user list

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Installation User</th>
<th>Running User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascend chip driver and firmware installation package</td>
<td>The user must be root.</td>
<td>The value must be HwHiAiUser, and HwHiAiUser must be created before the Ascend chip driver and firmware are installed.</td>
</tr>
<tr>
<td>CANN software package</td>
<td>The user can be root or non-root. For details, see the following content below the table.</td>
<td>The user must be a non-root user. For details, see the following content below the table.</td>
</tr>
</tbody>
</table>

You must create the HwHiAiUser user before installing the CANN software as the root user or a non-root user.

1. Create the HwHiAiUser user.
   
   Switch to the root user and run the following command to create the HwHiAiUser user:
   
   ```bash
groupadd HwHiAiUser           //Create the HwHiAiUser user group.
useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser           // Create an HwHiAiUser user whose owner group is HwHiAiUser.
```

   The user group of the HwHiAiUser user must be HwHiAiUser.

2. Select an installation mode.
   
   - **For installation as the root user:**
     
     You can install the software as the root user, but you must create the HwHiAiUser user. After the CANN software is installed, you must run the software as a non-root user, that is, the running user can only be in the HwHiAiUser user group.

   - **For installation as a non-root user:**
     
     In this scenario, the installation and running users must be the same.
     
     - If a non-root user exists, add the user to the HwHiAiUser user group and switch to the user for normal installation. The default installation user and running user are the current user.
     
     - If you want to use a new non-root user, you need to create the user first.

   To create a non-root user, run the following commands as the root user:
a. Create a non-root user.

```
useradd -g HwHiAiUser -d /home/username -m username
```

`username` is user-defined, but its owner group must be `HwHiAiUser`.

b. Set the password of the non-root user.

```
passwd username
```

**NOTE**

- The created running user cannot belong to the **root** user group.
- After the `HwHiAiUser` user is created, do not disable the login authentication function of the user.
- The password validity period is 90 days. You can change the validity period in the `/etc/login.defs` file or using the `chage` command. For details, see 5.5 Parameters description.

### Setting the Source

Before installing the deep learning engine package, toolbox, and framework plugin package, you need to download related dependencies. Ensure that the installation environment can be connected to the network. Run the following command as the **root** user to check whether the source is valid:

```
yum makecache
```

If an error is reported during command execution or dependency installation, check whether the network connection is normal, or replace the source in the `/etc/yum.repos.d/CentOS-Base.repo` file with an available source or use an image source. For details about how to configure a network proxy, see 6.7 Configuring a System Network Proxy.

### Configuring Permissions for the Installation User

If a non-root user is used for installation, perform the operations described in this section. If the **root** user is used for installation, you only need to perform step 1.

Before installing the deep learning engine package, toolbox, and framework plugin package, you need to download the dependencies using the `sudo yum` permission. Perform the following operations as the **root** user:

1. Run the following command to install `sudo`:

   ```
yum install sudo
```

2. Open the `/etc/sudoers` file:

   ```
chmod u+w /etc/sudoers
vim /etc/sudoers
```

3. Add the following content below `# User privilege specification` of the file:

   ```
username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/yum, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3.7 /usr/bin/python3, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

Replace `username` with the name of the common user who executes the installation script.

**NOTE**

Ensure that the last line of the `/etc/sudoers` file is `#includedir /etc/sudoers.d`. Otherwise, add it manually.
4. Run `.w` to save the file.
5. Run the following command to revoke the write permission on the `/etc/sudoers` file:
   ```
   chmod u-w /etc/sudoers
   ```

### Configuring the Maximum Number of Threads

1. Log in to the installation environment as the root user.
2. Configure environment variables to change the maximum number of threads.
   In the `/etc/profile` file, add the following content to the end of the file, save the file, and exit:
   ```
   ulimit -u unlimited
   ```
3. Run the following command to make the environment variable take effect:
   ```
   source /etc/profile
   ```

### Installing Dependencies

**NOTE**

- If you install Python and its dependencies as the root user, perform steps 1 to 4. Note that sudo needs to be deleted from the commands in steps 1 to 4.
- If python and its dependency are installed as a non-root user, run the `su - username` command to switch to the non-root user, and then perform steps 1 to 4.

**Step 1** Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

```
gcc --version
gcc-c++ --version
make --version
cmake --version
rpm -qa | grep unzip
rpm -qa | grep zlib-devel
rpm -qa | grep libffi-devel
rpm -qa | grep openssl-devel
rpm -qa | grep pciutils
rpm -qa | grep net-tools
rpm -qa | grep sqlite-devel
rpm -qa | grep blas-devel
rpm -qa | grep lapack-devel
rpm -qa | grep openblas-devel
rpm -qa | grep gcc-gfortran
```

If the following information is displayed, the installation is complete. Go to the next step.

```
gcc (GCC) 4.8.5 20150623 (Red Hat 4.8.5-39)
GNU Make 3.82
cmake version 2.8.12.2
unzip-6.0-21.el7.x86_64
zlib-devel-1.2.7-18.el7.x86_64
libffi-devel-3.0.13-18.el7.x86_64
openssl-devel-1.0.2k-19.el7.x86_64
pciutils-3.5.1-3.el7.x86_64
net-tools-2.0-0.25.20131004git.el7.x86_64
sqlite-devel-3.7.17-8.el7.x86_64
blas-devel-3.4.2-8.el7.x86_64
lapack-devel-3.4.2-8.el7.x86_64
openblas-devel-0.3.3-2.el7.x86_64
gcc-gfortran-4.8.5-39.el7.x86_64
```
Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

```bash
sudo yum install -y gcc gcc-c++ make cmake unzip zlib-devel libffi-devel openssl-devel pciutils net-tools sqlite-devel blas-devel lapack-devel openblas-devel gcc-gfortran
```

In the preceding steps, the message "No package libopenblas available" is displayed during the installation of openblas-devel. Run the following command to install the Linux extension package of the enterprise edition:

```bash
sudo yum install epel-release
```

If the CMake version installed using the preceding method is earlier than 3.5.1, see 6.5 Installing CMake 3.5.2.

**Step 2** Check whether the Python development environment is installed.

The deep learning engine package, toolbox, and framework plug-in package depend on the Python environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether they have been installed. If the following information is displayed, they have been installed. Go to the next step.

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:

1. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the installation environment. The command is as follows:

   ```bash
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```

2. Run the following command to go to the download directory and decompress the source code package:

   ```bash
tar -zxvf Python-3.7.5.tgz
   ```

3. Go to the decompressed folder and run the following configuration, build, and installation commands:

   ```bash
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-shared
make
sudo make install
```

The `--prefix` parameter specifies the Python installation path. You can change it based on the site requirements. The `--enable-shared` parameter is used to compile the `libpython3.7m.so.1.0` dynamic library.

This document uses `--prefix=/usr/local/python3.7.5` as an example. After the configuration, compilation, and installation commands are executed, the installation package is output to the `/usr/local/python3.7.5` directory, and the `libpython3.7m.so.1.0` dynamic library is output to the `/usr/local/python3.7.5/lib/libpython3.7m.so.1.0` directory.

4. Check whether `libpython3.7m.so.1.0` exists in `/usr/lib64` or `/usr/lib`. If yes, skip this step or back up the `libpython3.7m.so.1.0` file provided by the system and run the following command:

   ```bash
   Copy the compiled file `libpython3.7m.so.1.0` to `/usr/lib64`:
   sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```

   If the following information is displayed, enter `y` to overwrite the `libpython3.7m.so.1.0` file provided by the system.

   ```bash
cp: overwrite 'libpython3.7m.so.1.0'?
   ```

   If the `/usr/lib64` directory does not exist in the environment, copy the file to the `/usr/lib` directory.
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib

Replace the path of the **libpython3.7m.so.1.0** file as required.

5. Run the following commands to set the soft link:

```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
```

If a message indicating that the link already exists is displayed during the command execution, run the following command to delete the existing link and run the command again:

```
sudo rm -rf  /usr/bin/python3.7.5
sudo rm -rf  /usr/bin/pip3.7.5
sudo rm -rf  /usr/bin/python3.7
sudo rm -rf  /usr/bin/pip3.7
```

6. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.

```
python3.7.5 --version
pip3.7.5 --version
```

**Step 3**  
Table 4-7 lists the dependencies to be installed. Before installing a software package, ensure that its dependencies have been installed. If you run the following command as a non-root user, add **--user** to the end of the command. For example:

```
pip3.7 install numpy --user
```

The installation command can be run in any path.

**Table 4-7 Dependency list**

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Version</th>
<th>Installation Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
<td>For details, see <a href="#">Step 5 Check whether the Python development environment has been installed.</a></td>
</tr>
</tbody>
</table>
| numpy      | 1.17.2  | export CFLAGS=-std=c99  
ip3.7 install numpy==1.17.2 |
| decorator  | >=4.4.0 | pip3.7 install decorator |
| sympy      | 1.4     | pip3.7 install sympy==1.4 |
| cffi       | 1.12.3  | pip3.7 install cffi==1.12.3 |
| pyyaml     | -       | pip3.7 install pyyaml |
| pathlib2   | -       | pip3.7 install pathlib2 |
| grpcio     | -       | pip3.7 install grpcio |
| grpcio-tools | -       | pip3.7 install grpcio-tools |
| protobuf   | -       | pip3.7 install protobuf |
| scipy      | -       | pip3.7 install scipy |
| requests   | -       | pip3.7 install requests |
During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, rectify the fault by referring to 6.2 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 install Is Run?.

If the error message "subprocess.CalledProcessError: Command '('/lsb_release', '-a')' return non-zero exit status 1 " is displayed, rectify the fault by referring to 6.4 The error message "subprocess.CalledProcessError: Command '('/lsb_release', '-a')' return non-zero exit status 1 " is displayed during pip3.7.5 installation.

Step 4 CentOS 7.6 uses GCC 4.8.5 by default, which needs to be upgraded to GCC 7.3.0. See 6.6 Installing GCC 7.3.0.

---End

4.3.1.3 Debian (x86)

Environment Requirements

The following software or dependencies need to be installed in the operating environment: For details, see Installing Dependencies.

<table>
<thead>
<tr>
<th>Category</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
</tr>
<tr>
<td>cmake</td>
<td>3.5.1+</td>
</tr>
<tr>
<td>gcc</td>
<td>6.3.0</td>
</tr>
<tr>
<td>g++</td>
<td>6.3.0</td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
</tr>
</tbody>
</table>

NOTE Debian x86 uses GCC 6.3.0 by default, which needs to be upgraded to GCC 7.3.0.
<table>
<thead>
<tr>
<th>Category</th>
<th>Version Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>make</td>
<td>There is no version requirement. The version to be installed is subject to the source provided by the OS.</td>
</tr>
<tr>
<td>zlib1g-dev</td>
<td></td>
</tr>
<tr>
<td>libbz2-dev</td>
<td></td>
</tr>
<tr>
<td>libsqlite3-dev</td>
<td></td>
</tr>
<tr>
<td>libssl-dev</td>
<td></td>
</tr>
<tr>
<td>libxslt1-dev</td>
<td></td>
</tr>
<tr>
<td>libffi-dev</td>
<td></td>
</tr>
<tr>
<td>unzip</td>
<td></td>
</tr>
<tr>
<td>pciutils</td>
<td></td>
</tr>
<tr>
<td>net-tools</td>
<td></td>
</tr>
<tr>
<td>pyyaml</td>
<td></td>
</tr>
<tr>
<td>pathlib2</td>
<td></td>
</tr>
<tr>
<td>grpcio</td>
<td></td>
</tr>
<tr>
<td>grpcio-tools</td>
<td></td>
</tr>
<tr>
<td>protobuf</td>
<td></td>
</tr>
<tr>
<td>scipy</td>
<td></td>
</tr>
<tr>
<td>requests</td>
<td></td>
</tr>
</tbody>
</table>

**Check the umask of the root user.**

1. Log in to the installation environment as the root user.
2. Check the umask value of the root user.
   ```bash
   umask
   ```
3. If the umask value is not **0022**, append "umask 0022" to the file and save the file:
   ```bash
   vi ~/.bashrc
   source ~/.bashrc
   ```

**Creating Installation and Running Users**

*Table 4-9* lists the users for installing and running the Ascend chip driver, firmware, and CANN software.
### Table 4-9 Installation and running user list

<table>
<thead>
<tr>
<th>Package Type</th>
<th>Installation User</th>
<th>Running User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascend chip driver and firmware installation package</td>
<td>The user must be <strong>root</strong>.</td>
<td>The value must be <strong>HwHiAiUser</strong>, and <strong>HwHiAiUser</strong> must be created before the Ascend chip driver and firmware are installed.</td>
</tr>
<tr>
<td>CANN software package</td>
<td>The user can be <strong>root</strong> or non-root. For details, see the following content below the table.</td>
<td>The user must be a non-root user. For details, see the following content below the table.</td>
</tr>
</tbody>
</table>

You must create the **HwHiAiUser** user before installing the CANN software as the **root** user or a non-root user.

1. Create the **HwHiAiUser** user.
   
   Switch to the **root** user and run the following command to create the **HwHiAiUser** user:
   ```bash
   groupadd HwHiAiUser           //Create the HwHiAiUser user group.
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser           // Create an HwHiAiUser user whose owner group is HwHiAiUser.
   ``
   
   The user group of the **HwHiAiUser** user must be **HwHiAiUser**.

2. Select an installation mode.
   
   - **For installation as the root user:**
     
     You can install the software as the **root** user, but you must create the **HwHiAiUser** user. After the CANN software is installed, you must run the software as a non-root user, that is, the running user can only be in the **HwHiAiUser** user group.
     
   - **For installation as a non-root user:**
     
     In this scenario, the installation and running users must be the same.
     
     - If a non-root user exists, add the user to the **HwHiAiUser** user group and switch to the user for normal installation. The default installation user and running user are the current user.
     
     - If you want to use a new non-root user, you need to create the user first.

   To create a non-root user, run the following commands as the **root** user:
   
   a. Create a non-root user.
      ```bash
      useradd -g HwHiAiUser -d /home/username -m username
      ```
      
      **username** is user-defined, but its owner group must be **HwHiAiUser**.
   
   b. Set the password of the non-root user.
      ```bash
      passwd username
      ```
The created running user cannot belong to the root user group.

After the HwHiAiUser user is created, do not disable the login authentication function of the user.

The password validity period is 90 days. You can change the validity period in the /etc/login.defs file or using the chage command. For details, see 5.5 Parameters description.

Checking the Source Validity

Before installing the deep learning engine package, toolbox, and framework plug-in package, you need to download related dependencies. Ensure that the installation environment can be connected to the network.

Run the following command as the root user to check whether the source is valid:

```bash
apt-get update
```

If an error is reported during command execution or dependency installation, check whether the network connection is normal, or replace the source in the /etc/apt/sources.list file with an available source or use an image source. For details about how to configure a network proxy, see 6.7 Configuring a System Network Proxy.

Configuring Permissions for the Installation User

If a non-root user is used for installation, perform the operations described in this section. If the root user is used for installation, you only need to perform step 1.

Before installing the deep learning engine package, toolbox, and framework plug-in package, you need to download the dependencies using the sudo apt-get permission. Perform the following operations as the root user:

1. Run the following command to install sudo:

   ```bash
   apt-get install sudo
   ```

2. Open the /etc/sudoers file:

   ```bash
   chmod u+w /etc/sudoers
   vi /etc/sudoers
   ```

3. Add the following content below # User privilege specification of the file:

   ```bash
   username ALL=(ALL:ALL) NOPASSWD:SETENV:/usr/bin/apt-get, /usr/bin/unzip, /usr/bin/pip, /bin/tar, /bin/mkdir, /bin/rm, /bin/sh, /bin/cp, /bin/bash, /usr/bin/make install, /bin/ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7.5, /bin/ln -s /usr/local/python3.7.5/bin/python3.7 /usr/bin/python3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7, /bin/ln -s /usr/local/python3.7.5/bin/pip /usr/bin/pip3 /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7.5
   ```

   Replace username with the name of the common user who executes the installation script.

4. Run :wq! to save the file.

5. Run the following command to revoke the write permission on the /etc/sudoers file:

   ```bash
   chmod u-w /etc/sudoers
   ```
Installing Dependencies

**NOTE**

- If you install Python and its dependencies as the root user, perform steps 1 to 4. Note that sudo needs to be deleted from the commands in steps 1 to 4.
- If python and its dependency are installed as a non-root user, run the `su - username` command to switch to the non-root user, and then perform steps 1 to 4.

**Step 1** Check whether the Python dependencies and GCC software are installed.

Run the following commands to check whether the dependencies such as GCC, Make, and Python are installed:

```bash
gcc --version
g++ --version
make --version
cmake --version
dpkg -l zlib1g-dev| grep zlib1g-dev| grep ii
dpkg -l libbz2-dev| grep libbz2-dev| grep ii
dpkg -l libsqlite3-dev| grep libsqlite3-dev| grep ii
dpkg -l libssl-dev| grep libssl-dev| grep ii
dpkg -l libxslt1-dev| grep libxslt1-dev| grep ii
dpkg -l libffi-dev| grep libffi-dev| grep ii
dpkg -l unzip| grep unzip| grep ii
dpkg -l pchutils| grep pchutils| grep ii
dpkg -l net-tools| grep net-tools| grep ii
```

If the following information is displayed, the installation is complete. Go to the next step.

```bash
 gcc (Debian 6.3.0-18+deb9u1) 6.3.0 20170516
 GNU Make 4.7
 cmake version 3.7.2
 zlib1g-dev:amd64 1:1.2.8.dfsg-5 amd64  compression library - development
 libbz2-dev:amd64 1.0.6-8.1 amd64  high-quality block-sorting file compressor library - development
 libsqlite3-dev:amd64 3.16.2-5+deb9u1 amd64  SQLite 3 development files
 libssl-dev:amd64 1.1.0l-1+deb9u2 amd64  Secure Sockets Layer toolkit - development files
 libxslt1-dev:amd64 1.1.29-2.1+deb9u2 amd64  XSLT 1.0 processing library - development kit
 libffi-dev:amd64 3.2.1-6 amd64  Foreign Function Interface library (development files)
 unzip 6.0-21+deb9u2 amd64  De-archiver for .zip files
 pchutils 1.3.5.2-1 amd64  Linux PCI Utilities
 net-tools 1.60+git20161116.90da8a0-1 amd64  NET-3 networking toolkit
```

Otherwise, run the following command to install the software. You can change the following command to install only some of them as required.

```
sudo apt-get install -y gcc g++ cmake make zilib1g-dev libbz2-dev libsqlite3-dev libssl-dev libxslt1-dev libffi-dev unzip pchutils net-tools
```

**Step 2** Check whether the Python development environment is installed.

The development kit depends on the Python environment. Run the `python3.7.5 --version` and `pip3.7.5 --version` commands to check whether Python has been installed. If the following information is displayed, Python has been installed. Go to the next step.

```
Python 3.7.5
pip 19.2.3 from /usr/local/python3.7.5/lib/python3.7/site-packages/pip (python 3.7)
```

Otherwise, use the following procedure to install Python 3.7.5:

1. Run the `wget` command to download the source code package of Python 3.7.5 to any directory of the installation environment. The command is as follows:

   ```bash
   wget https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tgz
   ```
2. Run the following command to go to the download directory and decompress the source code package:
   ```
tar -zxvf Python-3.7.5.tgz
   ```

3. Prepare for the installation.
   Check the system time. If the system time is different from the actual time, adjust the system time to be the same as the actual time. The command is as follows:
   ```
date          // Check the system time and date.
date -s "20200620 11:21:30"    // Change the system time to the current time.
hwclock -w       // Write the current time and date to the BIOS to prevent them from becoming invalid after the restart.
   ```
   In the preceding command, 20200620 11:21:30 is an example of the actual time. Change it based on the actual situation.

4. Go to the decompressed folder and run the following configuration, build, and installation commands:
   ```
cd Python-3.7.5
./configure --prefix=/usr/local/python3.7.5 --enable-shared
make
sudo make install
   ```
   The --prefix parameter specifies the Python installation path. You can change it based on the site requirements. The --enable-shared parameter is used to compile the libpython3.7m.so.1.0 dynamic library.
   This document uses --prefix=/usr/local/python3.7.5 as an example. After the configuration, compilation, and installation commands are executed, the installation package is output to the /usr/local/python3.7.5 directory, and the libpython3.7m.so.1.0 dynamic library is output to the /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 directory.

5. Check whether libpython3.7m.so.1.0 exists in /usr/lib64 or /usr/lib. If yes, skip this step or back up the original file and run the following command:
   ```
Copy the following compiled files to /usr/lib64:
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib64
   ```
   If the /usr/lib64 directory does not exist in the environment, copy the file to the /usr/lib directory.
   ```
sudo cp /usr/local/python3.7.5/lib/libpython3.7m.so.1.0 /usr/lib
   ```
   Replace the path of the libpython3.7m.so.1.0 file as required.

6. Run the following commands to set the soft link:
   ```
sudo ln -s /usr/local/python3.7.5/bin/python3 /usr/bin/python3.7
sudo ln -s /usr/local/python3.7.5/bin/pip3 /usr/bin/pip3.7
sudo ln -s /usr/local/python3.7.5/bin/python3.7.5 /usr/bin/python3.7.5
sudo ln -s /usr/local/python3.7.5/bin/pip3.7.5 /usr/bin/pip3.7.5
   ```
   If a message indicating that the link already exists is displayed during the command execution, run the following command to delete the existing link and run the command again:
   ```
sudo rm -rf /usr/bin/python3.7.5
sudo rm -rf /usr/bin/pip3.7.5
sudo rm -rf /usr/bin/python3.7
sudo rm -rf /usr/bin/pip3.7
   ```

7. After the installation is complete, run the following commands to check the installation version. If the required version information is displayed, the installation is successful.
   ```
python3.7.5 --version
pip3.7.5 --version
   ```
Step 3  **Table 4-10** lists the dependencies to be installed. Before installing a software package, ensure that its dependencies have been installed. If you run the following command as a non-root user, add **--user** to the end of the command. For example:

```
pip3.7 install numpy --user
```

The installation command can be run in any path.

### Table 4-10 Dependency list

<table>
<thead>
<tr>
<th>Dependency</th>
<th>Version</th>
<th>Installation Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Python</td>
<td>3.7.5</td>
<td>For details, see Step 3 Check whether the Python development environment has been installed.</td>
</tr>
<tr>
<td>numpy</td>
<td>&gt;=1.13.3</td>
<td>pip3.7 install numpy</td>
</tr>
<tr>
<td>decorator</td>
<td>&gt;=4.4.0</td>
<td>pip3.7 install decorator</td>
</tr>
<tr>
<td>sympy</td>
<td>1.4</td>
<td>pip3.7 install sympy==1.4</td>
</tr>
<tr>
<td>cffi</td>
<td>1.12.3</td>
<td>pip3.7 install cffi==1.12.3</td>
</tr>
<tr>
<td>pyyaml</td>
<td>-</td>
<td>pip3.7 install pyyaml</td>
</tr>
<tr>
<td>pathlib2</td>
<td>-</td>
<td>pip3.7 install pathlib2</td>
</tr>
<tr>
<td>grpcio</td>
<td>-</td>
<td>pip3.7 install grpcio</td>
</tr>
<tr>
<td>grpcio-tools</td>
<td>-</td>
<td>pip3.7 install grpcio-tools</td>
</tr>
<tr>
<td>protobuf</td>
<td>-</td>
<td>pip3.7 install protobuf</td>
</tr>
<tr>
<td>scipy</td>
<td>-</td>
<td>pip3.7 install scipy</td>
</tr>
<tr>
<td>requests</td>
<td>-</td>
<td>pip3.7 install requests</td>
</tr>
</tbody>
</table>

During the command execution, if the network connection fails and the message "Could not find a version that satisfies the requirement xxx" is displayed, rectify the fault by referring to 6.2 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 install Is Run?.

If the error message "subprocess.CalledProcessError: Command ('lsb_release', '-a')' return non-zero exit status 1 " is displayed, rectify the fault by referring to 6.4 The error message "subprocess.CalledProcessError: Command ('lsb_release', '-a')' return non-zero exit status 1 " is displayed during pip3.7.5 installation..

Step 4  Debian x86 uses GCC 6.3.0 by default, which needs to be upgraded to GCC 7.3.0. See 6.6 Installing GCC 7.3.0.

----End
4.3.2 Installing the Ascend Chip Driver Firmware and Training Software (Easy Installation Tool)

You can use the one-click installation function to install the Ascend chip driver, firmware, deep learning engine, framework plug-in, and Toolbox.

Prerequisites

- Check the OS kernel version. For details, see "Checking the OS and Environment" in the Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9010).
- Obtain the driver package `{type}-NPU_Driver-{version}-xxx-{os}.run`, firmware package `{type} -NPU_Firmware-{version}.run`, deep learning engine package `{type}-NPU_Firmware-{version}.run`, toolbox package `{type}-NPU_Toolbox-{version}-xxx-gcc_version}.run`, and framework plug-in package `{type}-NPU_TFPlugin-{version}-xxx-gcc_version}.run` by referring to 4.2 Obtaining Software Packages.

Obtaining Easy Installation Tools

Download the easy installation tool `ascend-sdk-manager-{version}-{arch}-{os}.tgz` based on your OS architecture and upload the tool to any directory on the server, for example, `/home/HwHiAiUser`.

Run the `tar -xzvf ascend-sdk-manager-{version}-{arch}-{os}.tgz` command to decompress the package. The executable binary file `ascend-sdk-manager` is generated.

Installation Description

To use easy installation tools, you must be the root user.

If you use the easy installation to install the software, the software can be installed only in the default directory `/usr/local/Ascend`.

Procedure

**Step 1** Log in to the installation environment as the root user.

**Step 2** Upload the deep learning engine package, utility tool package, and framework plug-in package to any directory (for example, `/home/install`) in the installation environment.

**Step 3** Run the following commands to install the Ascend chip driver, firmware, deep learning engine, framework plug-in, and Toolbox:

```
/home/HwHiAiUser/ascend-sdk-manager app install /home/install train --mode=auto --silent=y
```

**NOTE**

The command format is `{sdk_path}/ascend-sdk-manager app install dir_path scene --mode=auto --silent=y`. The following table describes the parameters.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>{sdk_path}</td>
<td>Directory of the binary file <code>ascend-sdk-manager</code>, for example, <code>/home/HwHiAiUser</code>.</td>
</tr>
<tr>
<td>dir_path</td>
<td>Full path of the installation package, for example, <code>/home/install</code>.</td>
</tr>
</tbody>
</table>
| scene       | Installation scenario. The current version supports the following scenarios: `dev` (development environment), `infer` (inference operating environment), and `train` (training operating environment). The following lists the packages that can be installed in different scenarios.  
  - dev: Ascend-Toolkit  
  - infer: Ascend-NNRT, Ascend-Toolbox, NPU_Driver, NPU_Firmware  
  - train: Ascend-NNAE, Ascend-Toolbox, Ascend-TFPlugin, NPU_Driver, NPU_Firmware |
| mode        | The value is `auto`.                                                                                                                        |
| silent      | Silent installation option. In the current version, only `y` can be selected.                                                             |

**CAUTION**

- If a software package of an earlier version already exists in the installation environment, when you use the easy installation tool to install a software package of a later version in silent mode (that is, add `-silent=y` to the installation command), when you install the Toolbox, the `Aicpu_kernels` subpackage may have been installed and the installation is skipped. The details are as follows:
  
  Aicpu_kernels package has been installed on the path `/usr/local/Ascend`, the version is 1.73.T5.0.B050, and the version of this package is 1.73.T5.1.B050  
do you want to continue installing? [y/n]  
  stop installation!  
  Install aicpu_kernels end: 2020-07-10 15:07:19  
  [Toolbox] [20200710-15:07:19] [INFO] Ascend910-aicpu_kernels-1.73.T5.1.b050.run --full install success  
  [Toolbox] [20200710-15:07:19] [INFO] Ascend-Toolbox-20.0.0.B036-arm64-linux_gcc7.3.0.run install success  

In the preceding example, the later version `1.73.T5.1.B050` is not installed. In the actual environment, the earlier version `1.73.T5.0.B050` is still used. If you want to use the later aicpu version, you are advised to upgrade the Toolbox separately. For details, see 5.3 Upgrading the CANN Software Package (CLI).

---End

**4.3.3 Installing the Ascend Chip Driver Firmware and Training Software (CLI)**
4.3.3.1 Installing the Ascend Chip Driver and Firmware

Step 1  Check the OS kernel version. For details, see "Checking the OS and Environment" in the Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9010).

Step 2  Install the Ascend chip driver and firmware. For details, see "Installing the Driver and Firmware" in the Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9010).

----End

4.3.3.2 Installing the Training Software

Prerequisites

- Before installing the software, ensure that the Ascend chip driver has been installed in the installation environment.

Procedure

Obtain and install the deep learning engine package, Toolbox package, and framework plug-in package. The installation sequence of the software packages is not restricted, and the installation procedures are the same. The detailed installation procedure is as follows:

Step 1  Log in to the installation environment as the installation user of the software package.

Ensure that the installation user of the training software package is the same as the installation dependency user in 4.3.1 Before You Start.

Step 2  Upload the deep learning engine package, toolbox package, and framework plug-in package to any directory (for example, /home) in the installation environment.

Step 3  Go to the directory where the software packages are stored.

Step 4  Grant the execute permission on the software package.

```
chmod +x *.run
```

In the preceding command, * indicates the software package name. Replace it with the actual package name.

Step 5  Run the following command to check the consistency and integrity of the software package installation file:

```
./*.run --check
```

Step 6  Create a software installation path.

- If you need to specify the installation path, you need to create it first. For example, if the installation path is /home/work, run the `mkdir -p /home/`
work command to create an installation path and then select the path to install the software.

- If you do not specify an installation path, the software is installed in the default path. For details about the default path, see Table Software package installation path.

Table 4-11 Software package installation path

<table>
<thead>
<tr>
<th>Item</th>
<th>Path</th>
</tr>
</thead>
</table>
| Default installation path of a software package | - root user: `/usr/local/Ascend/${package_name}/(version)/(arch-linux-gccversion)`
| | - Non-root user: `${HOME}/Ascend/${package_name}/(version)/(arch-linux-gccversion)`
| Installation log path | `${install_path}/${package_name}/{version}/{arch-linux-gccversion}/ascend_install.log` |
| Path for recording information such as the software package version, CPU architecture, GCC version, and installation path after the installation | `${install_path}/${package_name}/{version}/{arch-linux-gccversion}/ascend-${package_name}_install.info` |

Table 4-11 describes the variables in the Table 3-15.

Table 4-12 Variable description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>$ {package_name}</code></td>
<td>Software package directory, which is named after the software package.</td>
</tr>
</tbody>
</table>
| `{version}` | Version directory, which is named after the version number of the software package. **NOTE**
| | Multiple versions can be installed, which are differentiated by version numbers. |
### Step 7 Install the software.

- If the installation path is specified, run the `./*.run --install --install-path=<path>` command to perform the installation.
  
  In the preceding command, `<path>` indicates the specified installation path, for example, `/home/work`.

- If the installation path is not specified, run the `./*.run --install` command to perform the installation.

#### CAUTION

- The AICPU operator package in the Toolbox must be installed as the root user. If you perform the installation as a non-root user, the AICPU operator package (`Ascend910-aicpu_kernels-{version}.run`) will be automatically decompressed to the `$install_path/toolbox/{version}/xxx-linux_gccx.x.x` directory during the installation. You need to add `sudo` before the following command or switch to the root user to install the operator package separately. Go to the `$install_path/toolbox/{version}/xxx-linux_gccx.x.x` directory and run the following command to perform the installation:
  
  ```
  ./Ascend910-aicpu_kernels-{version}.run --full
  ```

  The AICPU does not support the specified installation path and shares the installation path of the driver.

- The `--quiet` option is not supported when the Toolbox `Ascend-Toolbox-{version}-xxx-{gcc_version}.run` is installed in an x86 system.

- If the following information is displayed during the installation of the Toolbox `Ascend-Toolbox-{version}-xxx-{gcc_version}.run` in the x86 system, asking you whether to perform a hot reset, enter `n`. After the installation is complete, restart the OS for the setting to take effect. In the current version, only `n` is supported.

  The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n]

For more installation modes, see 7.1 Parameters.

If the following information is displayed, the software is successfully installed:

```
[INFO] xxx install success
[INFO] process end
```
4.3.4 Installing the Deep Learning Framework

This document uses TensorFlow as an example to describe how to install the deep learning framework. If you want to use the MindSpore framework, visit https://www.mindspore.cn/install/en to obtain the method of installing the MindSpore framework.

Installation Preparations

- For the x86 architecture, skip this step.
- In the AArch64 architecture, TensorFlow depends on H5py, and H5py depends on HDF5. Therefore, you need to compile and install HDF5 first. Otherwise, an error is reported when you use pip to install H5py. Perform the following operations as the root user:

**Step 1** Compile and install the HDF5.

1. Run the `wget` command to download the source code package of HDF5 to any directory of the installation environment. The command is as follows:
   ```
   wget https://support.hdfgroup.org/ftp/HDF5/releases/hdf5-1.10/hdf5-1.10.5/src/hdf5-1.10.5.tar.gz --no-check-certificate
   ```
2. Run the following command to go to the download directory and decompress the source code package:
   ```
   tar -zxvf hdf5-1.10.5.tar.gz
   ```
   Go to the decompressed folder and run the following configuration, build, and installation commands:
   ```
   cd hdf5-1.10.5/
   ./configure --prefix=/usr/include/hdf5
   make install
   ```

**Step 2** Configure environment variables and create a soft link to the dynamic link library (DLL).

1. Configure environment variables.
   ```
   export CPATH="/usr/include/hdf5/include/:/usr/include/hdf5/lib/"
   ```
2. Run the following command as the root user to create a soft link to the dynamic link library (DLL). Add sudo before the following commands as a non-root user:
   ```
   ln -s /usr/include/hdf5/lib/libhdf5.so /usr/lib/libhdf5.so
   ln -s /usr/include/hdf5/lib/libhdf5_hl.so /usr/lib/libhdf5_hl.so
   ```

**Step 3** Install h5py.

Run the following command as root user to install the h5py dependency:
```
pip3.7 install Cython
```
The h5py installation command is as follows:
```
pip3.7 install h5py==2.8.0
```
For the x86 architecture, download the software package from the pip source. For details, see https://www.tensorflow.org/install/pip?lang=python3. Note that the instructions provided by the TensorFlow website are incorrect. To download the CPU version from the pip source, you need to explicitly specify tensorflow-cpu. Otherwise, the GPU version is downloaded by default. That is, change tensorflow==1.15 —Release for CPU-only to tensorflow-gpu==1.15 —Release for CPU-only. In addition, the installation command pip3 install --user --upgrade tensorflow described on the official website needs to be changed to pip3.7 install Tensorflow-cpu==1.15 as the root user and to pip3.7 install Tensorflow-cpu==1.15 --user as a non-root user.

For the AArch64 architecture, the pip source does not provide the corresponding version. Therefore, you need to use GCC 7.3.0 to compile TensorFlow 1.15.0. For details about the compilation procedure, see https://www.tensorflow.org/install/source. Pay attention to the following points:

After downloading the tensorflow tag v1.15.0 source code, perform the following steps:

**Step 1** Download the nsync-1.22.0.tar.gz source code package.

1. Go to the tensorflow tag v1.15.0 source code directory, open the tensorflow/workspace.bzl file, and find the definition of tf_http_archive whose name is nsync:
   ```python
tf_http_archive(
    name = "nsync",
    sha256 = "caf32eb3b3d478b78c5f4c2b009c3400f8251f646b04cbdb5456566a9ce934c",
    strip_prefix = "nsync-1.22.0",
    system_build_file = clean_dep("//third_party/systemlibs:nsync.BUILD"),
    urls = [
      "https://storage.googleapis.com/mirror.tensorflow.org/github.com/google/nsync/archive/1.22.0.tar.gz",
      "https://github.com/google/nsync/archive/1.22.0.tar.gz",
    ],
  )
```
2. Download the nsync-1.22.0.tar.gz source code package from any path in urls and save it to any path.

**Step 2** Modify the nsync-1.22.0.tar.gz source code package:

1. Go to the directory where nsync-1.22.0.tar.gz is stored and decompress the source code package. Find the decompressed nsync-1.22.0 folder and the pax_global_header file.
2. Edit the nsync-1.22.0/platform/c++11/atomic.h file.
   Append the following information in red to NSYNC_CPP_START_.
   ```c
#include "nsync_cpp.h"
#include "nsync_atomic.h"
NSYNC_CPP_START_

#define ATM_CB_() __sync_synchronize()

static INLINE int atm_cas_nomb_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
  int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_relaxed, std::memory_order_relaxed));
  ATM_CB_();
  return result;
}
static INLINE int atm_cas_acq_u32_ (nsync_atomic_uint32_ *p, uint32_t o, uint32_t n) {
  int result = (std::atomic_compare_exchange_strong_explicit (NSYNC_ATOMIC_UINT32_PTR_ (p), &o, n, std::memory_order_acquire, std::memory_order_relaxed));
```
Step 3  Compress the nsync-1.22.0.tar.gz source code package.

Compress the modified nsync-1.22.0 folder and pax_global_header into a new nsync-1.22.0.tar.gz source code package (for example, /tmp/nsync-1.22.0.tar.gz).

Step 4  Generate a sha256sum verification code for the nsync-1.22.0.tar.gz source code package.

```bash
sha256sum /tmp/nsync-1.22.0.tar.gz
```

Obtain the sha256sum verification code.

Step 5  Change the sha256sum verification code and urls.

Go to the tensorflow tag v1.15.0 source code directory, open the tensorflow/workspace.bzl file, find the definition of tf_http_archive whose name is nsync, enter the verification code obtained in Step 4 after sha256=, and enter the first line of the list after urls=, enter the file:// index for storing the nsync-1.22.0.tar.gz file.

```python
tf_http_archive(
    name = "nsync",
    sha256 = "caf32e6b3d478b78ccf6c2ba009c3400f8251f646804bcb65465666a9cea93c4",
    strip_prefix = "nsync-1.22.0",
    system_build_file = clean_dep("//third_party/systemlibs:nsync.BUILD"),
    urls = [
        "file:///tmp/nsync-1.22.0.tar.gz",
        "https://storage.googleapis.com/mirror.tensorflow.org/github.com/google/nsync/archive/1.22.0.tar.gz",
        "https://github.com/google/nsync/archive/1.22.0.tar.gz",
    ],
)
```

Step 6  Continue to perform compilation from the official configuration build (https://www.tensorflow.org/install/source).

After the ./configure command is executed, add the following build option to the .tf_configure.bazelrc configuration file.

```bash
build:opt --ccopt=-D_GLIBCXX_USE_CXX11_ABI=0
```

Delete the following two lines.

```bash
build:opt --copt=-march=native
build:opt --host_copt=-march=native
```

Step 7  Proceed with the official compilation procedure (https://www.tensorflow.org/install/source).

Step 8  Install the compiled TensorFlow.
After the preceding steps are complete, the TensorFlow is packaged to the specified directory. Go to the specified directory and run the following command to install TensorFlow1.15 as the root user:

```
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
```

Run the following commands as a non-root user:

```
pip3.7 install tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl --user
```

----End

### 4.3.5 Installing protobuf Python

If the training script depends on the Python version of protobuf to store data in the serialized structure (for example, the serialization interfaces of TensorFlow), you need to install protobuf Python. The following uses EulerOS as an example to describe the installation procedure.

**Step 1** Check whether the system contains the dynamic library `/usr/local/python3.7.5/lib/python3.7/site-packages/google/protobuf/pyext/_message.cpython-37m-<arch>-linux-gnu.so`. If not, perform the following steps to install it:

**NOTE**

`/usr/local/python3.7.5/lib/python3.7/site-packages` is the path for installing third-party libraries using pip. You can run the `pip3.7 -V` command to query the path.

If the system displays `/usr/local/python3.7.5/lib/python3.7/site-packages/pip`, the path for installing third-party libraries using pip is `/usr/local/python3.7.5/lib/python3.7/site-packages`.

`arch` indicates the system architecture type, which can also be `x86_64`.

**Step 2** Uninstall protobuf.

```
pip3.7 uninstall protobuf
```

**Step 3** Download the protobuf software package.

Download the `protobuf-python-3.11.3.tar.gz` software package (or another version that is compatible with TensorFlow installed in the current environment) from `https://github.com/protocolbuffers/protobuf/releases/`, upload the package to any directory on the Linux server as the root user, and decompress the package.

**Step 4** Install protobuf as the root user.

Go to the protobuf software package directory.

1. Install the dependencies of protobuf.
   - If the OS is Ubuntu, run the following command:
     ```
     apt-get install autoconf automake libtool curl make g++ unzip libffi-dev
     ```
   - If the OS is CentOS or BClinux, run the following command:
     ```
     yum install autoconf automake libtool curl make gcc-c++ unzip libffi-devel
     ```

2. Grant the execute permission on the `autogen.sh` script and execute the script.
   ```
   chmod +x autogen.sh
   ./autogen.sh
   ```

3. Configure the installation path.
   ```
   ./configure
   ```
To specify the installation path, run the following command:

```
./configure --prefix=/protobuf
```

/protobuf indicates the installation path specified by the user.

4. Run the following commands to install protobuf:

```
make
make install
```

**NOTE**

Build using the `make` command could be time consuming. You can run the `make -jx` command to perform parallel build based on the number of CPU cores. x indicates the number of parallel tasks. For example, if there are two CPU cores, run the `make -j4` command.

5. Refresh the shared library.

```
ldconfig
```

After the protobuf is installed, the google/protobuf folder is generated in the include directory in this step. This folder stores the header files related to the protobuf. Generate the protoc executable file in the bin directory in this step. The protoc executable file is used to compile the *.proto file and generate the C++ header file and implementation file of the protobuf.

6. Check whether the software package is successfully installed.

```
ln -s /protobuf/bin/protoc /usr/bin/protoc
protoc --version
```

/protobuf is the installation path configured by the user in this step. If the installation path is not configured, run the `protoc --version` command to check whether the installation is successful.

**Step 5** Install the runtime library of protobuf Python.

1. Go to the python subdirectory in the protobuf software package directory and compile the Python runtime library.

```
python3.7 setup.py build --cpp_implementation
```

**NOTE**

Build a runtime library of the binary version. If the runtime library of the binary version cannot be generated by running the `python3.7 setup.py build` command, the processing of the serialization structure is slow.

2. Install the dynamic library.

```
cd .. && make install
```

Go to the python subdirectory and install the Python runtime library.

```
python3.7 setup.py install --cpp_implementation
```

3. Check whether the library has been installed.

Check whether the dynamic library `/usr/local/python3.7.5/lib/python3.7/site-packages/protobuf-3.11.3-py3.7-linux-aarch64.egg/google/protobuf/pyext/_message.cpython-37m-<arch>-linux-gnu.so` exists in the system.
/usr/local/python3.7.5/lib/python3.7/site-packages is the path for installing third-party libraries using pip. You can run the `pip3.7 -V` command to query the path.

If the system displays `/usr/local/python3.7.5/lib/python3.7/site-packages/pip`, the path for installing third-party libraries using pip is `/usr/local/python3.7.5/lib/python3.7/site-packages`.

`arch` indicates the system architecture type, which can also be `x86_64`.

4. Add the following environment variable setting to the startup script:

```bash
export LD_PATH_LIBRARY=/protobuf/lib
```

In the preceding command, `/protobuf` indicates the installation path set by the user in Step 4.3. The default installation path is `/usr/local`.

5. Establish a soft link.

If you configure the installation path, you need to establish a soft link. Otherwise, an error will be reported when TensorFlow is imported. The command is as follows:

```bash
ln -s /protobuf/lib/libprotobuf.so.22.0.3 /usr/lib/libprotobuf.so.22
```

/protobuf is the installation path configured by the user in Step 4.3.

---End

### 4.3.6 Changing NPU IP Addresses

#### Configuring NIC IP Address of a Device

- **SMP (symmetric multi-processor) mode**

  Log in to the AI Servers as the root user and configure the NIC IP address of each device. The configuration requirements are as follows:
  - NICs 0 and 4, 1 and 5, 2 and 6, and 3 and 7 of an AI Server must be in the same network segment respectively. NICs 0, 1, 2, and 3 must be in different network segments. NICs 4, 5, 6, and 7 must be in different network segments.
  - In the cluster scenario, the devices in the similar positions on AI Servers must be in the same network segment. For example, NIC 0 of AI Server 1 and AI Server 2 must be in the same network segment, and NIC 1 of AI Server 1 and AI Server 2 must be in the same network segment.

```bash
hccn_tool -i 0 -ip -s address 192.168.0.2 netmask 255.255.255.0
hccn_tool -i 1 -ip -s address 192.168.1.2 netmask 255.255.255.0
hccn_tool -i 2 -ip -s address 192.168.2.2 netmask 255.255.255.0
hccn_tool -i 3 -ip -s address 192.168.3.2 netmask 255.255.255.0
hccn_tool -i 4 -ip -s address 192.168.0.3 netmask 255.255.255.0
hccn_tool -i 5 -ip -s address 192.168.1.3 netmask 255.255.255.0
hccn_tool -i 6 -ip -s address 192.168.2.3 netmask 255.255.255.0
hccn_tool -i 7 -ip -s address 192.168.3.3 netmask 255.255.255.0
```

- **AMP (asymmetric multi-processor) mode**

  You do not need to restrict network segments. All NICs must be in the same network segment.

#### 4.3.7 Verifying the Installation

You can use the Ascend-DMI tool to check the compatibility between the device health information and software and hardware. After configuring the environment
variables, you can use the tool in any directory. You can use the tool as the root user or a non-root user. If you use the tool as a non-root user, perform the following steps to add the HwHiAiUser user group:

1. Log in to the server as the root user.
2. Run the `usermod -a -G HwHiAiUser {username}` command to add the user to the HwHiAiUser user group. `{username}` indicates the name of the non-root user. Replace it with the actual user name. The procedure is as follows:

To verify the installation, perform the following steps:

**Step 1** Add environment variables.

1. Log in to the server as the root user.
2. Run the `vi ~/.bashrc` command.
3. Add environment variables in Table 4-13 to the `.bashrc` file based on the application scenario.

### Table 4-13 Environment Variables

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Environment Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training (operating environment)</td>
<td>toolbox_install_path=/usr/local/Ascend/toolbox/{version}/xxx-{gcc-version} # Replace it with the actual installation path. {version} indicates the software package version, and xxx-{gcc-version} indicates the OS architecture and GCC version. nnae_install_path=/usr/local/Ascend/nnae/{version}/xxx-{gcc-version} # Replace it with the actual installation path. {version} indicates the software package version, and xxx-{gcc-version} indicates the OS architecture and GCC version. export PATH=${toolbox_install_path}/dmi/bin:${PATH} export LD_LIBRARY_PATH=/usr/local/dcmi:/usr/local/Ascend/add-ons:${toolbox_install_path}/dmi/lib64:$ {nnae_install_path}/fwkacllib/lib64:/usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/driver:$ {LD_LIBRARY_PATH}</td>
</tr>
<tr>
<td>Training (development environment)</td>
<td>toolkit_install_path=/usr/local/Ascend/ascend-toolkit/{version}/xxx-{gcc-version} # Replace it with the actual installation path. {version} indicates the software package version. export PATH=${toolkit_install_path}/dmi/bin:${PATH} export LD_LIBRARY_PATH=/usr/local/dcmi:/usr/local/Ascend/add-ons:${toolkit_install_path}/dmi/lib64:$ {toolkit_install_path}/fwkacllib/lib64:/usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/driver:$ {LD_LIBRARY_PATH}</td>
</tr>
</tbody>
</table>

4. Run `source .bashrc` to make the environment variables take effect.

**Step 2** Check the device health status.

1. Run the `ascend-dmi info` command and obtain the card number from the Card parameter in the displayed table, as shown in the red box in Figure 4-1.
2. Run the `ascend-dmi -dg -c {card-number} -l 1` command to query the health status. `{card-number}` indicates the card number. Replace it as required.

For details about how to check the device health status, see 7.4 Performing Fault Diagnostics.

**Step 3** Check the software and hardware compatibility.

- If you use the default path when installing the software package:
  Run the `ascend-dmi -c` command to check the software and hardware compatibility.

- You need to set this parameter if the default installation path is not used. For example, if the software package is installed in the `/home/xxx/Ascend` directory, run the following command:

  `ascend-dmi -c -p /home/xxx/Ascend`

For details about how to check software and hardware compatibility, see 7.5 Performing a Software and Hardware Compatibility Test.

---End

### 4.4 Installation in a Container

#### 4.4.1 Installing the Ascend Chip Driver and Firmware on the Host
4.4.1.1 Creating a Running User

The driver and firmware must be installed as user root. However, after the installation is complete, a running user HwHiAiUser is required to run the driver and firmware (the running user cannot belong to the root user group). Therefore, you need to create the running user HwHiAiUser before the installation.

To create the HwHiAiUser user, perform the following steps as user root:

1. Create the HwHiAiUser user.
   ```
   groupadd HwHiAiUser
   useradd -g HwHiAiUser -d /home/HwHiAiUser -m HwHiAiUser
   ```
2. Set the password of the HwHiAiUser user.
   ```
   passwd HwHiAiUser
   ```

**NOTE**
- The owner group of the running user is HwHiAiUser.
- After the HwHiAiUser user is created, do not disable the login authentication function of the user.
- The password validity period is 90 days. You can change the validity period in the /etc/login.defs file or using the chage command. For details, see 5.5 Parameters description.

4.4.1.2 Installing the Ascend Chip Driver and Firmware

**Step 1** Check the OS kernel version. For details, see "Checking the OS and Environment" in the Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9010).

**Step 2** Install the Ascend chip driver and firmware. For details, see "Installing the Driver and Firmware" in the Atlas 800 AI Training Server Driver and Firmware Installation and Upgrade Guide (Model 9010).

----End

4.4.2 Installing the Toolbox on the Host

The Ascend-Toolbox-{version}-x.x.x-linux_gccx.x.x.run tool package cannot be installed in the container. Therefore, you need to install the tool package by referring to 4.3.3.2 Installing the Training Software. The tool package contains the AI CPU operator. Therefore, the AI CPU operator must be installed.

4.4.3 Installing Ascend Docker on the Host Machine

Ascend Docker provides Ascend NPU-based containerization support for all AI training and inference jobs so that AI jobs can run smoothly on Ascend devices as Docker containers.

**Prerequisites**
- Ensure that the Docker program has been installed in the packaging environment and the Docker version is 18.03 or later.
Obtain the Ascend Docker software package from **4.2 Obtaining Software Packages** based on the OS and platform type. For details about the software package, see **Table 4-14**.

**Table 4-14 Software packages**

<table>
<thead>
<tr>
<th>Software packages</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ascend-docker-runtime_{version}_&lt;arch&gt;.deb</td>
<td>Ascend Docker installation package for Ubuntu and /Debian.</td>
</tr>
<tr>
<td>ascend-docker-runtime-{version}.&lt;arch&gt;.rpm</td>
<td>Ascend Docker installation package for CentOS.</td>
</tr>
</tbody>
</table>

**NOTE**

{version} indicates the software version, and <arch> indicates the OS architecture.

### Installing Ascend Docker

**Step 1** Upload the obtained software package to the server.

**Step 2** Go to the directory where the software package is stored and run different commands to install Ascend Docker based on the OS type.

```
<table>
<thead>
<tr>
<th>OS</th>
<th>Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu/Debian</td>
<td><code>sudo dpkg -i ascend-docker-runtime_{version}_&lt;arch&gt;.deb</code></td>
</tr>
<tr>
<td>CentOS</td>
<td><code>sudo rpm -i ascend-docker-runtime-{version}.&lt;arch&gt;.rpm</code></td>
</tr>
</tbody>
</table>
```

**Step 3** Run the following command to restart the gaussdb process:

```
sudo systemctl restart docker
```

----End

### 4.4.4 Creating a Container Image Using a Docker File

**Mapping Between Host and Container OSs**

**Table 4-15** describes the mapping between host and container OSs.

<table>
<thead>
<tr>
<th>Host OS</th>
<th>Container OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ubuntu 18.04 X86</td>
<td>Ubuntu 18.04 X86</td>
</tr>
<tr>
<td>CentOS 7.6 X86</td>
<td>CentOS 7.6 X86</td>
</tr>
</tbody>
</table>
### Prerequisites

Obtain the software packages of the corresponding operating system and the Dockerfile and script files required for packaging images by referring to [Table 4-16](#).

In the names of the deep learning acceleration engine package and framework plug-in package, `{version}` indicates the version number, `xxx` indicates the OS, and `{gcc_version}` indicates the GCC version. This section uses Ubuntu ARM as an example. The Ascend software package has been installed on the host server.

#### Table 4-16 Required software

<table>
<thead>
<tr>
<th>Software Package</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ascend-NNAE-{version}-xxx-{gcc_version}.run</td>
<td>Deep learning acceleration engine package</td>
<td>See 4.2 Obtaining Software Packages.</td>
</tr>
<tr>
<td>Dockerfile</td>
<td>Required for creating an image</td>
<td>Prepared by users</td>
</tr>
<tr>
<td>ascend_install.info</td>
<td>Software package installation log file</td>
<td>Copy the <code>/etc/ascend_install.info</code> file from the host.</td>
</tr>
<tr>
<td>version.info</td>
<td>Driver package version information file</td>
<td>Copy the <code>/usr/local/Ascend/driver/version.info</code> file from the host.</td>
</tr>
</tbody>
</table>
### Software Package

<table>
<thead>
<tr>
<th>Software Package</th>
<th>Description</th>
<th>How to Obtain</th>
</tr>
</thead>
<tbody>
<tr>
<td>prebuild.sh</td>
<td>Script used to prepare for the installation of the training operating environment, for example, configure the agent.</td>
<td>Prepared by users</td>
</tr>
<tr>
<td>install_ascend_pkgs.sh</td>
<td>Script for installing the Ascend software package.</td>
<td></td>
</tr>
<tr>
<td>postbuild.sh</td>
<td>Delete the installation packages, scripts, and proxy configurations that do not need to be retained in the container.</td>
<td></td>
</tr>
</tbody>
</table>

### Procedure

**Step 1**  
Upload the software package, host Ascend software package installation information file, and driver package version information file to the same directory (for example, `/home/test`) on the server.
- Ascend-NNAE-{version}-xxx-{gcc_version}.run
- Ascend-TFPlugin-{version}-xxx-{gcc_version}.run
- ascend_install.info
- version.info

**Step 2**  
Log in to the server as the **root** user.

**Step 3**  
Perform the following steps to prepare the `prebuild.sh` file:
1. Go to the directory where the software package is stored and run the following command to create the `prebuild.sh` file:
   ```bash
   vi prebuild.sh
   ```
2. For details about the content to be written, see the *compilation example*. After writing the content, run the `:wq` command to save the content. The following uses Ubuntu ARM as an example.

**Step 4**  
Perform the following steps to prepare the `install_ascend_pkgs.sh` file:
1. Go to the directory where the software package is stored and run the following command to create the `install_ascend_pkgs.sh` file:
   ```bash
   vi install_ascend_pkgs.sh
   ```
2. For details about the content to be written, see the *compilation example*. After writing the content, run the `:wq` command to save the content. The following uses Ubuntu ARM as an example.

**Step 5**  
Perform the following steps to prepare the `postbuild.sh` file:
1. Go to the directory where the software package is stored and run the following command to create the `postbuild.sh` file:
   ```bash
   vi postbuild.sh
   ```

2. For details about the content to be written, see the compilation example. After writing the content, run the `:wq` command to save the content. The following uses Ubuntu ARM as an example.

**Step 6** Perform the following steps to prepare the `Dockerfile` file:

1. Go to the directory where the software package is stored and run the following command to create the `Dockerfile` file:
   ```bash
   vi Dockerfile
   ```

2. Write the following content and run the `:wq` command to save the content. The following uses Ubuntu ARM as an example.

   ```bash
   FROM ubuntu:18.04
   ARG NNAE_VERSION
   ARG NNAE_ARCH=arm64
   ARG TF_PKG=tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
   ARG HOST_ASCEND_BASE=/usr/local/Ascend
   ARG CNT_ASCEND_BASE=/usr/local/Ascend/nnae/$NNAE_VERSION/$NNAE_ARCH-linux_gcc7.3.0
   ARG INSTALL_ASCEND_PKGS_SH=install_ascend_pkgs.sh
   ARG PREBUILD_SH=prebuild.sh
   ARG POSTBUILD_SH=postbuild.sh
   WORKDIR /root
   COPY . ./
   # Trigger prebuild.sh.
   RUN bash -c "test -f $PREBUILD_SH && bash $PREBUILD_SH"
   # System package
   RUN apt update && \
       apt install --no-install-recommends python3.7 python3.7-dev curl g++ pkg-config unzip \
               libblas3 liblapack3 liblapack-dev libblas-dev gfortran libhdf5-dev \
               -y
   # pip3.7
   RUN curl -k https://bootstrap.pypa.io/get-pip.py -o get-pip.py && \
       cd /tmp && \
       apt-get download python3-distutils && \
       dpkg-deb -x python3-distutils_*.deb / && \
       rm python3-distutils_*.deb && \
       cd - && \
       python3.7 get-pip.py && \
       rm get-pip.py
   # HwHiAiUser
   RUN umask 0022 && \
       groupadd HwHiAiUser && \
       useradd -g HwHiAiUser -m -d /home/HwHiAiUser HwHiAiUser
   # Python package
   RUN pip3.7 install numpy && \
       pip3.7 install decorator && \
       pip3.7 install sympy==1.4 && \
       pip3.7 install cffi==1.12.3 && \
       pip3.7 install pyyaml && \
       pip3.7 install pathlib2 && \
       pip3.7 install grpcio && \
       pip3.7 install grpcio-tools && \
       pip3.7 install protobuf && \
       pip3.7 install scipy && \
       pip3.7 install requests
   # Ascend package
   RUN bash $INSTALL_ASCEND_PKGS_SH
   ```
# Matching package
RUN pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/topi-*py3-none-any.whl && \
pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/te-*py3-none-any.whl && \
pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/hccl-*py3-none-any.whl

# TF installation
ENV LD_LIBRARY_PATH=\ 
/usr/lib/aarch64-linux-gnu/hdf5/serial:\ 
$HOST_ASCEND_BASE/add-ons:\ 
$CNT_ASCEND_BASE/fwkacllib/lib64:\ 
$HOST_ASCEND_BASE/driver/lib64/common:\ 
$HOST_ASCEND_BASE/driver/lib64/driver:$LD_LIBRARY_PATH
RUN pip3.7 install $TF_PKG

# Environment variables
ENV GLOG_v=2
ENV TBE_IMPL_PATH=$CNT_ASCEND_BASE/opp/op_impl/built-in/ai_core/tbe
ENV PATH=$CNT_ASCEND_BASE/fwkacllib/ccec_compiler/bin/:$PATH
ENV PYTHONPATH=$TBE_IMPL_PATH:$PYTHONPATH

# Trigger postbuild.sh.
RUN bash -c "test -f $POSTBUILD_SH && bash $POSTBUILD_SH" && \
rm $POSTBUILD_SH

![NOTE](image)

To obtain the image **ubuntu:18.04**, you can also run the **docker pull ubuntu:18.04** command to obtain the image from Docker Hub.

**Step 7** Go to the directory where the software package is stored and run the following command to create a container image:

docker build -t test_train:v0.1 --build-arg http_proxy=http://user:password@proxyserverip:port \
--build-arg https_proxy=http://user:password@proxyserverip:port --build-arg NNAE_VERSION=20.0.0.8001 .

In the preceding command, **user** indicates the username on the intranet, **password** indicates the user password, **proxyserverip** indicates the IP address of the proxy server, and **port** indicates the port number.

**Table 4-17** describes the commands.

**Table 4-17** Command parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t</td>
<td>Specifies the image name.</td>
</tr>
<tr>
<td>-f</td>
<td>Specifies the dockerfile path.</td>
</tr>
<tr>
<td>--build-arg</td>
<td>Specifies parameters in the dockerfile file.</td>
</tr>
<tr>
<td>test_train:v0.1</td>
<td>Specifies the image name and tag. Change them based on the actual situation.</td>
</tr>
<tr>
<td>NNAE_VERSION</td>
<td>Version of the deep learning acceleration engine package. Change it based on the actual situation.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>http_proxy, https_proxy</td>
<td>Network proxy, which is configured by users. If the user server can directly connect to the external network, you do not need to set this parameter.</td>
</tr>
</tbody>
</table>

If "Successfully built xxx" is displayed, the image is successfully created. Do not omit the command at the end.

**Step 8** After the image is created, run the following command to view the image information:

```
docker images
```

Example:

<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_train</td>
<td>v1.0</td>
<td>d82746acd7f0</td>
<td>27 minutes ago</td>
<td>749MB</td>
</tr>
</tbody>
</table>

----End

**Compilation Examples**

1. Compilation example of `prebuild.sh`

   a. Ubuntu

```
#!/bin/bash
#--------------------------------------------------------------------------------
# Use the bash syntax to compile script code and prepare for the installation, for example, configuring the proxy.
# This script will be run before the formal creation process is started.
# The following content is only an example. You need to configure the information based on the server network.
# Note: After this script is run, it will not be automatically cleared. If it does not need to be retained in the image, clear it from the postbuild.sh script.
#--------------------------------------------------------------------------------
# Configure the DNS proxy. Add the following information in bold to the /etc/resolv.conf file based on the actual situation:
```
```
teet /etc/resolv.conf <<- EOF
nameserver 10.129.2.34
EOF
```

b. CentOS

```
#!/bin/bash
#--------------------------------------------------------------------------------
# VERSION: 20.0.0.RC1
# Use the bash syntax to compile script code and prepare for the installation, for example, configuring the proxy.
# This script will be run before the formal creation process is started.
# Note: After this script is run, it will not be automatically cleared. If it does not need to be retained in the image, clear it from the postbuild.sh script.
#--------------------------------------------------------------------------------
# https://bbs.huaweicloud.com/forum/thread-50114-1-1.html
# Modify the following file to resolve the problem that an error is reported when NumPy is installed in the CentOS image.
tee ./npy_math_internal.h.src.patch <<-EOF
480c480
< NPY_INPLACE @type@ npy_@kind@@c@(@type@ x, @type@ y)
```

```
2. Compilation example of install_ascend_pkgs.sh

```bash
#!/bin/bash
#--------------------------------------------------------------------------------
# Use the bash syntax to compile script code and install the Ascend software package.
# Note: After this script is run, it will not be automatically cleared. If it does not need to be retained in the image, clear it from the postbuild.sh script.
#--------------------------------------------------------------------------------
# Copy the /etc/ascend_install.info file on the host to the current directory before creating the container image.
cp ascend_install.info /etc/
# Copy the /usr/local/Ascend/driver/version.info file on the host to the current directory before creating the container image.
mkdir -p /usr/local/Ascend/driver/
cp version.info /usr/local/Ascend/driver/
# Ascend-NNAE-20.0.0.B001-arm64-linux_gcc7.3.0.run
chmod +x Ascend-NNAE-20.0.0.B001-arm64-linux_gcc7.3.0.run
./Ascend-NNAE-20.0.0.B001-arm64-linux_gcc7.3.0.run --install-path=/usr/local/Ascend/ --install --quiet
# Ascend-TFPlugin-20.0.0.B001-arm64-linux_gcc7.3.0.run
chmod +x Ascend-TFPlugin-20.0.0.B001-arm64-linux_gcc7.3.0.run
./Ascend-TFPlugin-20.0.0.B001-arm64-linux_gcc7.3.0.run --install-path=/usr/local/Ascend/ --install --quiet
# Only for the installation of the nnae package. Therefore, the nnae package needs to be cleared. When the container is started, the nnae package is mounted by the Ascend Docker.
rm -f version.info
rm -rf /usr/local/Ascend/driver/
```

3. Compilation example of postbuild.sh (Ubuntu)

```bash
#!/bin/bash
#--------------------------------------------------------------------------------
# Use the bash syntax to compile the script code and delete the installation packages, scripts, and proxy configurations that do not need to be retained in the container.
# This script will be run after the formal creation process ends.
# Note: After this script is run, it is automatically cleared and will not be left in the image. The scripts and Working Dir are stored in /root.
#--------------------------------------------------------------------------------
rm -f ascend_install.info
rm -f prebuild.sh
rm -f install_ascend_pkgs.sh
rm -f Dockerfile.tt.*
rm -f Ascend-NNAE-20.0.0.B001-arm64-linux_gcc7.3.0.run
rm -f Ascend-TFPlugin-20.0.0.B001-arm64-linux_gcc7.3.0.run
rm -f tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
rm -f /etc/apt/apt.conf.d/80proxy

tee /etc/resolv.conf <<- EOF
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
# Run "systemd-resolve --status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

options edns0
```

```bash
tee /etc/resolv.conf <<- EOF
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "systemd-resolve --status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

options edns0
EOF
```
4. Compilation example of **Dockerfile**

a. Docker file example for Ubuntu ARM

```plaintext
FROM ubuntu:18.04

ARG NNAE_VERSION
ARG NNAE_ARCH=arm64
ARG TF_PKG=tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
ARG HOST_ASCEND_BASE=/usr/local/Ascend
ARG CNT_ASCEND_BASE=/usr/local/Ascend/nnae/$NNAE_VERSION/$NNAE_ARCH-linux_gcc7.3.0
ARG INSTALL_ASCEND_PKGS_SH=install_ascend_pkgs.sh
ARG PREBUILD_SH=prebuild.sh
ARG POSTBUILD_SH=postbuild.sh
WORKDIR /root
COPY . ./

# Trigger prebuild.sh.
RUN bash -c "test -f $PREBUILD_SH && bash $PREBUILD_SH"

# System package
RUN apt update && \
    apt install --no-install-recommends python3.7 python3.7-dev curl g++ \
    libblas3 liblapack3 liblapack-dev libblas-dev gfortran libhdf5-dev libffi-dev -y

# pip3.7
RUN curl -k https://bootstrap.pypa.io/get-pip.py -o get-pip.py && \
    cd /tmp && \
    apt-get download python3-distutils && \
    dpkg-deb -x python3-distutils_*.deb / && \
    rm python3-distutils_*.deb && \
    cd - && \
    python3.7 get-pip.py && \
    rm get-pip.py

# HwHiAiUser
RUN umask 0022 && \
    groupadd HwHiAiUser && \
    useradd -g HwHiAiUser -m -d /home/HwHiAiUser HwHiAiUser

# Python package
RUN pip3.7 install numpy && \
    pip3.7 install decorator && \
    pip3.7 install sympy==1.4.0 && \
    pip3.7 install cffi==1.12.3 && \
    pip3.7 install pyyaml && \
    pip3.7 install pathlib2 && \
    pip3.7 install grpcio && \
    pip3.7 install grpcio-tools && \
    pip3.7 install protobuf && \
    pip3.7 install scipy && \
    pip3.7 install requests

# Ascend package
RUN bash $INSTALL_ASCEND_PKGS_SH

# Matching package
RUN pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/topi-*-py3-none-any.whl && \
    pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/te-*-py3-none-any.whl && \
    pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/hccl-*-py3-none-any.whl

# TF installation
ENV LD_LIBRARY_PATH=\n/usr/lib/aarch64-linux-gnu/hdf5/serial:/
$HOST_ASCEND_BASE/add-ons;\n$CNT_ASCEND_BASE/fwkacllib/lib64:
```

---

**NOTE:**

Nameserver 8.8.8.8
Nameserver 8.8.4.4
RUN pip3.7 install $TF_PKG

# Environment variables
ENV GLOG_v=2
ENV TBE_IMPL_PATH=$CNT_ASCEND_BASE/opp/op_impl/built-in/ai_core/tbe
ENV PATH=$CNT_ASCEND_BASE/fwkacllib/ccec_compiler/bin/:$PATH
ENV PYTHONPATH=$TBE_IMPL_PATH:$PYTHONPATH

# Trigger postbuild.sh.
RUN bash -c "test -f $POSTBUILD_SH && bash $POSTBUILD_SH" && rm $POSTBUILD_SH

b. Docker file example for Ubuntu x86
FROM ubuntu:18.04

ARG NNAE_VERSION
ARG NNAE_ARCH=x86_64
ARG TF_PKG=tensorflow-cpu==1.15.0
ARG HOST_ASCEND_BASE=/usr/local/Ascend
ARG CNT_ASCEND_BASE=/usr/local/Ascend/nnae/$NNAE_VERSION/$NNAE_ARCH-linux_gcc7.3.0
ARG INSTALL_ASCEND_PKGS_SH=install_ascend_pkgs.sh
ARG PREBUILD_SH=prebuild.sh
ARG POSTBUILD_SH=postbuild.sh
WORKDIR /root
COPY . ./

# System package
RUN apt update && apt install --no-install-recommends python3.7 python3.7-dev curl g++ pkg-config unzip libblas3 liblapack3 liblapack-dev libblas-dev gfortran libhdf5-dev libffi-dev -y

# pip3.7
RUN curl -k https://bootstrap.pypa.io/get-pip.py -o get-pip.py &&
    cd /tmp &&
    apt-get download python3.7-dev-distutils &
    dpkg-deb -x python3.7-distutils_* . deb &
    rm python3.7-distutils_* . deb &
    cd - &
    python3.7 get-pip.py &
    rm get-pip.py

# HwHiAiUser
RUN umask 0022 &&
    groupadd HwHiAiUser &&
    useradd -g HwHiAiUser -m -d /home/HwHiAiUser HwHiAiUser

# Python package
RUN pip3.7 install numpy &
    pip3.7 install decorator &
    pip3.7 install sympy &
    pip3.7 install cffi &
    pip3.7 install pyyaml &
    pip3.7 install pathlib2 &
    pip3.7 install grpcio &
    pip3.7 install grpcio-tools &
    pip3.7 install protobuf &
    pip3.7 install scipy &
    pip3.7 install requests

# Ascend package
RUN bash $INSTALL_ASCEND_PKGS_SH

# Matching package
RUN pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/topi-* . py3-none-any.whl &
pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/te-*py3-none-any.whl && 
pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/hccl-*py3-none-any.whl

# TF installation
ENV LD_LIBRARY_PATH=
/usr/lib/aarch64-linux-gnu/hdf5/serial:
$HOST_ASCEND_BASE/add-ons:
$CNT_ASCEND_BASE/fwkacllib/lib64:
$HOST_ASCEND_BASE/driver/lib64:
$HOST_ASCEND_BASE/driver/lib64/driver:$LD_LIBRARY_PATH

RUN pip3.7 install $TF_PKG

# Environment variables
ENV GLOG_v=2
ENV TBE_IMPL_PATH=$CNT_ASCEND_BASE/opp/op_impl/built-in/ai_core/tbe
ENV PATH=$CNT_ASCEND_BASE/fwkacllib/ccec_compiler/bin/:$PATH
ENV PYTHONPATH=$TBE_IMPL_PATH:$PYTHONPATH

# Trigger
postbuild.sh
RUN bash -c "test -f $POSTBUILD_SH && bash $POSTBUILD_SH" && 
rm $POSTBUILD_SH

c. Docker file example for CentOS ARM
FROM arm64v8/centos:7
ARG NNAE_VERSION
ARG NNAE_ARCH=arm64
ARG TF_PKG=tensorflow-1.15.0-cp37-cp37m-linux_aarch64.whl
ARG HOST_ASCEND_BASE=/usr/local/Ascend
ARG CNT_ASCEND_BASE=/usr/local/Ascend/nnae/$NNAE_VERSION/$NNAE_ARCH-linux_gcc7.3.0
ARG INSTALL_ASCEND_PKGS_SH=install_ascend_pkgs.sh
ARG PREBUILD_SH=prebuild.sh
ARG POSTBUILD_SH=postbuild.sh
WORKDIR /root
COPY . ./

# Trigger
prebuild.sh
RUN bash -c "test -f $PREBUILD_SH && bash $PREBUILD_SH"

# Install the compilation environment.
RUN yum makecache && 
  yum -y install centos-release-scl && 
  yum -y install devtoolset-7 && 
  yum -y install devtoolset-7-gcc-c++ && 
  yum -y install epel-release

# scl enable devtoolset-7 bash
SHELL ["/usr/bin/bash", "--compat", "4.4.0"]
# Install Python.
RUN yum -y install wget zlib-devel bzip2-devel openssl-devel ncreuses-devel 
  xz-devel libffi-devel hdf5-devel patch && 
  wget --no-check-certificate https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tar.xz && 
  tar -zxvf Python-3.7.5.tar.xz && 
  cd Python-3.7.5 && 
  make && make install && 
  cd .. && rm -rf build

# scl enable devtoolset-7 bash
SHELL ["/usr/bin/bash", "--compat", "4.4.0"]
# Install NumPy.
RUN python3.7 -m pip install cython && 
  yum install -y blas-devel lapack-devel atlas openblas-devel && 
  wget --no-check-certificate https://files.pythonhosted.org/packages/01/1b/ 
  d3dcbadb51f71be02df6e6ee20d64f77ff6d97f83b77f65e98c8a651981/numpy-1.18.5.zip && 
  unzip numpy-1.18.5.zip && rm -f numpy-1.18.5.zip && 
  cd numpy-1.18.5 && 

issue 01 (2020-08-03)  Copyright © Huawei Technologies Co., Ltd. 92
patch numpy/core/src/npymath/npy_math_internal.h.src ./npy_math_internal.h.src.patch && \
python3.7 setup.py build && \
python3.7 setup.py install -O1 --skip-build && \
cd -

# HwHiAiUser
RUN umask 0022 && \
groupadd HwHiAiUser && \
useradd -g HwHiAiUser -m -d /home/HwHiAiUser HwHiAiUser

# Python package
RUN pip3.7 install decorator && \
pip3.7 install sympy==1.4 && \
pip3.7 install cffi==1.12.3 && \
pip3.7 install pyyaml && \
pip3.7 install pathlib2 && \
pip3.7 install grpcio && \
pip3.7 install grpcio-tools && \
pip3.7 install protobuf && \
pip3.7 install scipy && \
pip3.7 install requests

# Ascend package
RUN bash $INSTALL_ASCEND_PKGS_SH

# Matching package
RUN pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/topi-*-py3-none-any.whl && \
pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/te-*-py3-none-any.whl && \
pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/hccl-*-py3-none-any.whl

# TF installation
ENV LD_LIBRARY_PATH=\n$HOST_ASCEND_BASE/add-ons:\n$CNT_ASCEND_BASE/fwkacllib/lib64:\n$HOST_ASCEND_BASE/driver/lib64/common:\n$HOST_ASCEND_BASE/driver/lib64/driver:$LD_LIBRARY_PATH
RUN pip3.7 install $TF_PKG

# Environment variables
ENV GLOG_v=2
ENV TBE_IMPL_PATH=$CNT_ASCEND_BASE/opp/op_impl/built-in/ai_core/tbe
ENV PATH=$CNT_ASCEND_BASE/fwkacllib/ccec_compiler/bin/:$PATH
ENV PYTHONPATH=$TBE_IMPL_PATH:$PYTHONPATH

# Trigger postbuild.sh
RUN bash -c "test -f $POSTBUILD_SH && bash $POSTBUILD_SH" && \
    rm $POSTBUILD_SH

d. Docker file example for CentOS x86
FROM centos:7.6.1810

ARG NNAE_VERSION
ARG NNAE_ARCH=x86_64
ARG TF_PKG=tensorflow-cpu==1.15.0
ARG HOST_ASCEND_BASE=/usr/local/Ascend
ARG CNT_ASCEND_BASE=/usr/local/Ascend/nnae/$NNAE_VERSION/$NNAE_ARCH-linux_gcc7.3.0
ARG INSTALL_ASCEND_PKGS_SH=install_ascend_pkgs.sh
ARG PREBUILD_SH=prebuild.sh
ARG POSTBUILD_SH=postbuild.sh
WORKDIR /root
COPY . .

# Trigger prebuild.sh
RUN bash -c "test -f $PREBUILD_SH && bash $PREBUILD_SH"

# Install the compilation environment.
RUN yum makercache && \
yum -y install centos-release-scl && \
yum -y install devtoolset-7 && \

---

4 Installing the Operating Environment
yum -y install devtoolset-7-gcc-c++ &&
yum -y install epel-release

# scl enable devtoolset-7 bash
SHELL ["/usr/bin/scl", "enable", "devtoolset-7"]

# Install Python.
RUN yum -y install wget zlib-devel bzip2-devel openssl-devel ncurses-devel 
    sqlite-devel readline-devel tk-devel gdbm-devel db4-devel libpcap-devel 
    xz-devel libffi-devel hdf5-devel patch &&
    wget --no-check-certificate https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tar.xz &&
    cd Python-3.7.5 &&
    mkdir build && cd build &&
    ../configure &&
    make -j && make install &&
    cd .. && rm -rf build

# scl enable devtoolset-7 bash
SHELL ["/usr/bin/scl", "enable", "devtoolset-7"]

# Install NumPy.
RUN python3.7 -m pip install cython &&
    yum install -y blas-devel lapack-devel atlas openblas-devel &&
    wget --no-check-certificate https://files.pythonhosted.org/packages/01/1b/d3ddcabd5817be02df0ee6ee20d6f477ff6d0d97f83b77f65e98c8a651981/numpy-1.18.5.zip &&
    unzip numpy-1.18.5.zip && rm -f numpy-1.18.5.zip &&
    cd numpy-1.18.5 &&
    patch numpy/core/src/numpymath/npy_math_internal.h.src ../npy_math_internal.h.src.patch &&
    python3.7 setup.py build &&
    python3.7 setup.py install -O1 --skip-build &&
    cd -

# HwHiAiUser
RUN umask 0022 &&
    groupadd HwHiAiUser &&
    useradd -g HwHiAiUser -m -d /home/HwHiAiUser HwHiAiUser

# Python package
RUN pip3.7 install decorator &&
    pip3.7 install sympy==1.4 &&
    pip3.7 install cffi==1.12.3 &&
    pip3.7 install pyyaml &&
    pip3.7 install pathlib2 &&
    pip3.7 install scipy &&
    pip3.7 install requests

# Ascend package
RUN bash $INSTALL_ASCEND_PKGS_SH

# Matching package
RUN pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/topi-*-py3-none-any.whl &&
    pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/te-*-py3-none-any.whl &&
    pip3.7 install $CNT_ASCEND_BASE/fwkacllib/lib64/hccl-*-py3-none-any.whl

# TF installation
ENV LD_LIBRARY_PATH= LD_LIBRARY_PATH=/host_ascend_base/add-ons:
$CNT_ASCEND_BASE/fwkacllib/lib64:
$CNT_ASCEND_BASE/fwkacllib/lib64/hccl-*-py3-none-any.whl

# TF PKG
RUN pip3.7 install $TF_PKG

# Environment variables
ENV GLOG_v=2
ENV TBE_IMPL_PATH=$CNT_ASCEND_BASE/opp/op_impl/built-in/ai_core/tbe
ENV PATH=$CNT_ASCEND_BASE/fwkacllib/ccce Compiler/bin/$PATH
ENV PYTHONPATH=${TBE_IMPL_PATH}:${PYTHONPATH}

# Trigger postbuild.sh.
RUN bash -c "test -f $POSTBUILD_SH && bash $POSTBUILD_SH" && 
    rm $POSTBUILD_SH

e. Docker file example for Debian x86

FROM gcc:7.3.0

ARG VERSION
ARG ARCH=x86_64
ARG TF_PKG=tensorflow-cpu==1.15.0
ARG HOST_ASCEND_BASE=/usr/local/Ascend
ARG NNAE_PATH=/usr/local/Ascend/nnae/$VERSION/$ARCH-linux_gcc7.3.0
ARG TF_PLUGIN_PATH=/usr/local/Ascend/tfplugin/$VERSION/$ARCH-linux_gcc7.3.0
ARG INSTALL_ASCEND_PKGS_SH=install_ascend_pkgs.sh
ARG PREBUILD_SH=prebuild.sh
ARG POSTBUILD_SH=postbuild.sh
WORKDIR /tmp
COPY . ./

# Trigger prebuild.sh.
RUN bash -c "test -f $PREBUILD_SH && bash $PREBUILD_SH || true"

# Python 3.7.5
RUN apt update && 
    apt install --no-install-recommends wget unzip zlib1g-dev libncurses5-dev libgdbm-dev 
    libnss3-dev libssl-dev 
    libreadline-dev libffi-dev curl libbz2-dev libhdf5-dev -y

RUN wget --no-check-certificate https://www.python.org/ftp/python/3.7.5/Python-3.7.5.tar.xz 
    && 
    tar -xf Python-3.7.5.tar.xz 
    cd Python-3.7.5 
    mkdir build && cd build 
    ../configure --enable-shared && 
    make -j && make install && 
    cd .. && rm -rf build 
    cd .. 
    ldconfig

# HwHiAiUser
RUN umask 0022 && 
    groupadd HwHiAiUser && 
    useradd -g HwHiAiUser -m -d /home/HwHiAiUser HwHiAiUser

# Python package
RUN pip3.7 install numpy && 
    pip3.7 install decorator && 
    pip3.7 install sympy==1.4 && 
    pip3.7 install cffi==1.12.3 && 
    pip3.7 install pyyaml && 
    pip3.7 install pathlib2 && 
    pip3.7 install grpcio && 
    pip3.7 install grpcio-tools && 
    pip3.7 install protobuf && 
    pip3.7 install scipy && 
    pip3.7 install requests

# Ascend package
RUN bash $INSTALL_ASCEND_PKGS_SH

# TF installation
ENV LD_LIBRARY_PATH=\
4.4.5 Deploying a Training Container

Prerequisites

- Ensure that the Docker program and Ascend Docker software package have been installed in the packaging environment.
- Before starting the container, ensure that the Ascend chip driver, firmware, and toolbox package `Ascend-Toolbox-{version}-xxx-{gcc_version}.run` have been installed on the host.
- If multiple servers are used for cluster training, ensure that the NPU IP address has been configured on the host before deploying the training container. For details, see 4.3.6 Changing NPU IP Addresses.

Deploying the Image

⚠️ CAUTION

When the host environment is CentOS, the SELinux security module of CentOS is enabled by default. As a result, the local directory mounted to the container does not have the execute permission. Therefore, you need to run the `su -c "setenforce 0"` command to temporarily disable the SELinux. After related services are complete, run the `su -c "setenforce 0"` command to enable SELinux again.

Step 1 Run the following command in the operating environment to run a container based on the new image:

docker run -it -e ASCEND_VISIBLE_DEVICES=xxx -d --pids-limit 409600 docker_image /bin/bash
Table 4-18 Argument description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| -e ASCEND_VISIBLE_DEVICES=xxx | Use the `ASCEND_VISIBLE_DEVICES` environment variable to specify the NPU device to be mounted to the container, and use the device sequence number to specify the device. You can specify a single device or a device range. The two types of devices can be used together. Example:  
  1. `-e ASCEND_VISIBLE_DEVICES=0` indicates that device 0 (/dev/davinci0) is mounted to the container.  
  2. `-e ASCEND_VISIBLE_DEVICES=1,3` indicates that devices 1 and 3 are mounted to the container.  
  3. `-e ASCEND_VISIBLE_DEVICES=0-2` indicates that devices 0 to 2 (including devices 0 and 2) are mounted to the container. The effect is the same as that of `-e ASCEND_VISIBLE_DEVICES=0,1,2`.  
  4. `ASCEND_VISIBLE_DEVICES=0-2,4:` indicates that devices 0 to 2 and device 4 are mounted to the container. The effect is the same as that of `-e ASCEND_VISIBLE_DEVICES=0,1,2,4`. |
| -d --pids-limit 409600     | If the host OS is CentOS or BC-linux, the maximum number of threads in Docker is 4092, which cannot meet the training requirements. This parameter needs to be increased to configure the maximum number of Docker threads in CentOS/BC-linux when the container is started. |
| docker_image               | Generated image file, for example, `ubuntu:example`.                                                                                     |

In addition to NPUs and management devices, Ascend Docker Runtime mounts the following directories and files to the container in read-only mode by default.

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/local/Ascend/driver/lib64</td>
<td>Directory for storing user-mode libraries provided by the driver.</td>
</tr>
<tr>
<td>/usr/local/Ascend/driver/tools</td>
<td>Directory for storing tools provided by the driver. (Not all tools support the container scenario.)</td>
</tr>
</tbody>
</table>
### Directory Description

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/local/Ascend/driver/include</td>
<td>Directory for storing the header file <code>dsmi_common_interface.h</code> provided by the driver.</td>
</tr>
<tr>
<td>/usr/local/dcmi</td>
<td>Directory for storing DCMI header files and libraries.</td>
</tr>
<tr>
<td>/usr/local/sbin/npu-smi</td>
<td>Directory for storing files and the npu-smi tool.</td>
</tr>
<tr>
<td>/var/log/npu/conf/slog/slog.conf</td>
<td>Directory for storing the configuration file of the log service.</td>
</tr>
</tbody>
</table>

If the container ID (3330d6524117 in this example) is displayed, the container is running and you have accessed the container.

```
root@xxx:~# docker run -it -e ASCEND_VISIBLE_DEVICES=xxx docker_image /bin/bash
root@3330d6524117:/#
```

**NOTE**

- You can use the `ASCEND_RUNTIME_OPTIONS` environment variable to adjust the mounted targets. If the value is set to `NODRV`, non-device directories and files are not mounted. For example:

  ```bash
docker run --rm -it -e ASCEND_VISIBLE_DEVICES=xxx -e ASCEND_RUNTIME_OPTIONS=NODRV docker_image /bin/bash
  ```

  Only NPU devices and management devices are mounted to the container to support driver installation in the container.

- If the container image OS is Ubuntu ARM and you want to use the npu-smi tool in the container, run the following commands in the container:

  ```bash
  mkdir /lib64
  cd /lib64
  ln -sf /lib/ld-linux-aarch64.so.1 ld-linux-aarch64.so.1
  ```

**Step 2**

Set the following environment variable. In the command, `/usr/local/Ascend` is the default installation path. Replace it as required.

```bash
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/local/Ascend/driver/lib64/common:/usr/local/Ascend/driver/lib64/driver:/usr/local/Ascend/add-ons
```

**Step 3**

Run the following command to start the `slogd` daemon process:

```bash
su HwHiAiUser --command "/usr/local/Ascend/driver/tools/slogd &"
```

**Step 4**

Run the following command to check whether the daemon process is successfully run:

```bash
ps -ef | grep -v grep | grep "tools/slogd"
```

If the following information is displayed, the process has been started and the image file has been deployed successfully:
HwHiAiU+  13  1  0 09:15 ?  00:00:00 /usr/local/Ascend/driver/tools/slogd

----End
5 Common Operations

5.1 Uninstalling the Ascend Chip Driver Firmware and Training Software (Easy Installation Tool)

Uninstallation Description

- If you only need to uninstall the specified software packages, including deep learning engine, deep learning framework plug-in, development kit, or Toolbox, you can uninstall them in any sequence. However, if you want to uninstall the Ascend chip driver and firmware, you need to uninstall other software packages and then uninstall the driver and firmware. There is no sequence requirement for uninstalling the driver and firmware.
- If you use the one-click uninstallation function to uninstall all software packages, you do not need to consider the sequence.

Uninstalling a Specified Software Package

**Step 1** Log in to the environment as user `root`.

**Step 2** Run the following commands to uninstall a specified component. For details about related parameters, see Table 5-1.

```
./ascend-sdk-manager app uninstall shortname --version=version --gcc_type=gcc_version --silent=y or
./ascend-sdk-manager app uninstall shortname version gcc_type y
```
**Table 5-1** Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortname</td>
<td>Component short name, for example, 'driver', 'firmware', 'nnae', 'nnrt', 'toolbox', 'tfplugin', 'toolkit'. When auto is entered, all components are uninstalled in one-click mode.</td>
</tr>
<tr>
<td>version</td>
<td>Component version. You can run the ./ascend-sdk-manager app display command to query the version information as required.</td>
</tr>
<tr>
<td>gcc_type</td>
<td>GCC version. You can run the ./ascend-sdk-manager app display command to query the GCC version as required.</td>
</tr>
<tr>
<td>silent</td>
<td>Silent uninstallation. Enter y or n to replace y or n that the system requires during the uninstallation.</td>
</tr>
</tbody>
</table>

---

**One-click Uninstallation**

**Step 1** Log in to the environment as user root.

**Step 2** Run the following command to uninstall all software packages in one-click mode. For details about related parameters, see Table 5-1.

```
./ascend-sdk-manager app uninstall auto --silent=y
```

```
./ascend-sdk-manager app uninstall auto y
```

---

**Uninstallation Example**

- Run the following commands to uninstall all versions of a specified component based on the component short names:

  ```
  ./ascend-sdk-manager app uninstall nnae --silent=y
  [NNAE] [20200713-19:13:12] [INFO] Ascend-fwkacllib-1.73.t5.0.b050-centos7.6.aarch64.run uninstall start
  [NNAE] [20200713:13:12]------------------------------------------
  uninstall /usr/local/Ascend/nnae/20.0.0.B036/arm64-linux_gcc7.3.0 success
  [NNAE] [20200713-19:13:19] [INFO] Ascend-fwkacllib-1.73.5.1.b050-centos7.6.aarch64.run uninstall start
  [NNAE] [20200713-19:13:19]------------------------------------------
  uninstall /usr/local/Ascend/nnae/20.0.0.RC1/arm64-linux_gcc7.3.0 success
  
  ./ascend-sdk-manager app uninstall toolbox --version=20.0.0.RC1 --silent=y
  [Toolbox] [20200713-19:16:08] [INFO] Ascend910-aicpu_kernels-1.73.5.1.b050.run uninstall start
  [Toolbox] [20200713-19:16:08]------------------------------------------
  uninstall /usr/local/Ascend/toolbox/20.0.0.RC1/arm64-linux_gcc7.3.0 success
  
  ./ascend-sdk-manager app uninstall tfplugin --version=20.0.0.RC1 --gcc_type=gcc7.3.0 --silent=y
  [TFPlugin] [20200713-19:18:02] [INFO] Ascend-tfplugin-1.73.5.1.b050-centos7.6.aarch64.run uninstall start
  ```
5.2 Upgrading the Ascend Chip Driver Firmware and Training Software (Easy Installation Tool)

Upgrade Description

- Before upgrading the CANN software, you need to upgrade the Ascend chip firmware and driver to ensure that the CANN software matches the Ascend chip firmware and driver versions.
- Upgrade the Ascend chip firmware first, and then the driver.

Upgrading a Specified Software Package

Step 1 Log in to the environment as user root.

Step 2 Run the following commands to update a specified component based on file_path in the installation package. For details about related parameters, see Table 5-2.

```bash
./ascend-sdk-manager app upgrade file_path --install_path=install_path --silent=y or
./ascend-sdk-manager app upgrade file_path install_path y
```

Table 5-2 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_path</td>
<td>Complete path of the installation package.</td>
</tr>
<tr>
<td>install_path</td>
<td>Installation path of the target version.</td>
</tr>
<tr>
<td>silent</td>
<td>Silent upgrade. Enter y or n to replace y or n that the system requires during the upgrade as required.</td>
</tr>
</tbody>
</table>

**CAUTION**

When the easy installation tool is used to upgrade the development kit Ascend-Toolkit-{version}-xxx-{gcc_version}.run and Toolbox Ascend-Toolbox-{version}-xxx-{gcc_version}.run in the x86 system, the silent upgrade option supports only --silent=n. After the installation is complete, restart the OS for the settings to take effect.

----End

Upgrade Example

The following is an example of upgrading the TFPlugin software package:
5.3 Upgrading the CANN Software Package (CLI)

NOTE

After the software package is upgraded, services are not affected.

Before upgrading the CANN software, you need to upgrade the Ascend chip firmware and driver to ensure that the CANN software matches the Ascend chip firmware and driver versions.

Step 1 Log in to the installation environment as the installation user of the software package.

Step 2 Upgrade the software.

- If you specify a path when installing the software package, run the `<package-path>/*.run --upgrade --install-path=<path>`. 

In the preceding command, `<package-path>` indicates the directory where the software package is stored, `<path>` indicates the software package installation directory specified by the user, and * indicates the software package name. Replace them with the actual ones.

For example, if the software package is stored in `/home/package` and the software package installation directory is `/home/work`, run the following command:

```
/home/package/*.run --upgrade --install-path=/home/work
```

- If you do not specify a path when installing the software package, run the `<package-path>/*.run --upgrade` command to upgrade the software package.

For example, if the software package is stored in the `/home/package` directory, run the following command:

```
/home/package/*.run --upgrade
```
When the development kit `Ascend-Toolkit-{version}-xxx-{gcc_version}.run` and Toolbox `Ascend-Toolbox-{version}-xxx-{gcc_version}.run` are upgraded in the x86 system, if the following information is displayed, asking you whether to perform hot reset, enter n. After the installation is complete, restart the OS for the settings to take effect. In the current version, only n is supported.

The installation of aicpu_kernels needs to restart the device to take effect, do you want to hot_reset the device? [y/n]

5.4 Uninstalling the CANN Software (CLI)

If you uninstall only the inference engine, development kit, or Toolbox, you can uninstall them in any sequence. However, if you also need to uninstall the inference card driver, you need to uninstall other software packages before uninstalling the driver.

Uninstallation

To uninstall an installed software package, perform the following steps:

**Step 1** Log in to the installation environment as the installation user of the software package.

**Step 2** Uninstall the software package.

- If you specify a path when installing the software package, run the `<package-path>/*.run --uninstall --install-path=<path>` to uninstall the software package.

  In the preceding command, `<package-path>` indicates the directory where the software package is stored, and `<path>` indicates the specified software package installation directory. In the preceding command, * indicates the software package name. Replace it with the actual package name.

  For example, if the software package is stored in `/home/package` and the software package installation directory is `/home/work`, run the following command:

  `/home/package/*.run --uninstall --install-path=/home/work`

- If you do not specify a path when installing the software package, run the `<package-path>/*.run --uninstall` command to uninstall the software package.

  For example, if the software package is stored in the `/home/package` directory, run the following command:

  `/home/package/*.run --uninstall`

If the following information is displayed, the software is successfully uninstalled:

```
[INFO] xxx uninstall success
[INFO] process end
```
Uninstallation Using a Script

You can also use the uninstallation script to uninstall the. The following describes how to uninstall the development suite package.

Step 1 Go to the directory where the uninstallation script of the software is stored. Generally, the script is stored in the script directory.

```
cd {package-path}/ascend-toolkit/20.0.0.xxx/xxx-linux_gccx.x.x/script
```

{path} indicates the tool installation directory. If the software package is installed in a specified directory, replace it with the specified directory. If this parameter is not specified, the default directory is used. Replace it by referring to Table 3-14.

20.0.0.xxx indicates the software package version, and xxx-linux_gccx.x.x indicates the OS architecture and GCC version. Replace them as required.

Step 2 Run the ./uninstall.sh command to run the script.

5.5 Parameters description

To ensure user security, you need to set the validity period of a user. You can run the chage command to set the validity period of a user.

Command:

```
```

Table 5-3 describes the parameters.

Table 5-3 Parameters description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m</td>
<td>Minimum time (in days) for which the password must be used. The password cannot be changed during this period. The value 0 indicates that the password can be changed at any time.</td>
</tr>
<tr>
<td>-M</td>
<td>Maximum validity period (days) of a password. The value 1 indicates that the validity check of the password can be disabled.</td>
</tr>
<tr>
<td>-d</td>
<td>Date when the password was changed the last time.</td>
</tr>
<tr>
<td>-I</td>
<td>Maximum idle period (in days) after which the user account will be disabled. After the specified time period has expired, the password will be invalid.</td>
</tr>
<tr>
<td>-E</td>
<td>Date when the user account expires. The user account is unavailable when the account validity period has expired.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Parameters</td>
</tr>
<tr>
<td>-----------</td>
<td>------------</td>
</tr>
<tr>
<td>-W</td>
<td>Number of days in advance users are notified that their passwords are about to expire.</td>
</tr>
<tr>
<td>-l</td>
<td>Lists the current settings. It helps non-privileged users to determine the time when their passwords or accounts expire.</td>
</tr>
</tbody>
</table>

**NOTE**

- Table 5-3 lists only common parameters. You can run the `chage --help` command to display detailed parameter description.
- The date format is YYYY-MM-DD. For example, `chage -E 2017-12-01 test` indicates that the password of user test expires on December 1, 2017.
- User is mandatory. Replace it with the actual user name. The default user name is root.

For example, to change the validity period of user *test* to December 31, 2017, run the following command:

chage -E 2017-12-31 test
6.1 What Do I Do If the "Software Has Been Installed" During RUN Package Installation?

6.2 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 install Is Run?

6.3 pip3.7.5 intall numply Error

6.4 The error message "subprocess.CalledProcessError: Command '((lsb_release', '-a')' return non-zero exit status 1 " is displayed during pip3.7.5 installation.

6.5 Installing CMake 3.5.2

6.6 Installing GCC 7.3.0

6.7 Configuring a System Network Proxy

6.1 What Do I Do If the "Software Has Been Installed" During RUN Package Installation?

Symptom

The following message is displayed during installation:

run package is already installed, install failed

Solution

The software package cannot be installed repeatedly. You need to uninstall the software package and then install it. For details, see 5.4 Uninstalling the CANN Software (CLI).
6.2 What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 install Is Run?

Symptom

What Do I Do If "Could not find a version that satisfies the requirement xxx" Is Displayed When pip3.7.5 install Is Run?

Possible Causes

The pip source is not configured.

Procedure

Configure the pip source as follows:

Step 1 Run the following command as the installation user of the software package:

cd ~/.pip

If a message indicating that the directory does not exist is displayed, run the following command to create the directory:

mkdir ~/.pip

cd ~/.pip

Create a pip.conf file in the .pip directory:

touch pip.conf

Step 2 Edit the pip.conf file.

Run the vi pip.conf command to open the pip.conf file and edit the file as follows.

```
[install]
#Configure the trusted host as required.
trusted-host=cmc-cd-mirror.rnd.huawei.com

[global]
#Configure the sources as required.
index-url=http://cmc-cd-mirror.rnd.huawei.com/pypi/simple/
```

Step 3 Run the :wq! command to save the file and exit.

-----End
6.3 pip3.7.5 install numpy Error

Symptom
When you run the pip3.7.5 install numpy command to install the dependency, the error message "Could not build wheels for numpy which use PEP 517 and cannot be install directly" is displayed, as shown in the following figure.

Possible Causes
The default gcc version installed in the CentOS system is too early. As a result, the NumPy fails to be installed.

Procedure
Run the following commands to install virt-manager-common and virt-manager:

export CFLAGS=-std=c99
pip3.7 install numpy==1.17.2

6.4 The error message "subprocess.CalledProcessError: Command '('/lsb_release', '-a')' return non-zero exit status 1 " is displayed during pip3.7.5 installation.

Symptom
During dependency installation, the error message "subprocess.CalledProcessError: Command '('/lsb_release', '-a')' return non-zero exit status 1 " is displayed when you run the pip3.7.5 install xxx command to install related software. The error message is as follows:
Possible Causes

When the subprocess module of Python 3.7.5 is executed, the system displays a message indicating that the lsb_release.py module cannot be found when the lsb_release -a command is executed. The lib path of Python 3.7.5 is /usr/local/python3.7.5/lib/python3.7/. The lsb_release.py module does not exist in the path. Therefore, an error is reported.

Procedure

**Step 1** Run the following command to search for the missing file lsb_release.py:
```bash
find / -name lsb_release
```
After the preceding command is executed, the following path is obtained. The path is only an example and may vary according to the actual situation.
```
/usr/bin/lsb_release
```

**Step 2** Run the following command to delete the /usr/bin/lsb_release file in **Step 1**:
```bash
rm /usr/bin/lsb_release
```

**Step 3** Run the pip3.7.5 list command to check whether the fault is rectified.

-----End

6.5 Installing CMake 3.5.2

1. Run the wget command to download the source code package of CMake to any directory on the server:
   ```bash
   wget https://cmake.org/files/v3.5/cmake-3.5.2.tar.gz --no-check-certificate
   ```
2. Run the following command to go to the download directory and decompress the source code package:
   ```bash
   tar -zxvf cmake-3.5.2.tar.gz
   ```
3. Go to the decompressed folder and run the following configuration, compilation, and installation commands:
   ```bash
   cd cmake-3.5.2
   ./bootstrap --prefix=/usr
   make
   sudo make install
   ```
4. After the installation is complete, run the `cmake --version` command again to check the version number.

6.6 Installing GCC 7.3.0

Perform the following steps as the root user:

**Step 1** Download gcc-7.3.0.tar.gz from [https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7.3.0/gcc-7.3.0.tar.gz](https://mirrors.tuna.tsinghua.edu.cn/gnu/gcc/gcc-7.3.0/gcc-7.3.0.tar.gz).

**Step 2** The GCC installation occupies a large amount of temporary space. Therefore, run the following command to clear the `/tmp` directory:

```
sudo rm -rf /tmp/*
```

**Step 3** Install the dependency.

For CentOS/BCLinux, run the following command:

```
yum install bzip2
```

For Ubuntu or Debian, run the following command:

```
apt-get install bzip2
```

**Step 4** Compile and install GCC.

1. Go to the directory where the `gcc-7.3.0.tar.gz` source code package is located and run the following command to decompress the source code package:

```
tar -zxvf gcc-7.3.0.tar.gz
```

2. Go to the decompressed folder and run the following command to download the GCC dependency package:

```
cd gcc-7.3.0
./contrib/download_prerequisites
```

If an error is reported during the execution of the preceding command, run the following command to download the dependency package from the `gcc-7.3.0` folder:

```
wget http://gcc.gnu.org/pub/gcc/infrastructure/mpc-1.0.3.tar.gz
wget http://gcc.gnu.org/pub/gcc/infrastructure/isl-0.16.1.tar.bz2
```

After the preceding dependency packages are downloaded, run the following command again:

```
./contrib/download_prerequisites
```

If the preceding command fails to be verified, ensure that the dependency package is successfully downloaded at a time and no repeated download occurs.

3. Run the configuration, compilation, and installation commands.

```
./configure --enable-languages=c,c++ --disable-multilib --with-system-zlib --prefix=/usr/local/gcc7.3.0
make -j15 #Check the number of CPUs by using grep -w processor /proc/cpuinfo|wc -l. In this example, the number is 15. You can set the parameters as required.
make install
```
Step 5 Configure environment variables.

When you perform training, the compilation environment after the GCC upgrade is required. Therefore, you need to configure environment variables in the training script by running the following command:

```bash
export LD_LIBRARY_PATH=.../xxx/xxx/xxx/lib64
```

In the preceding command, `.../xxx/xxx/xxx/` is the GCC installation path configured in 3. You need to configure the path. In this example, the path is `/usr/local/gcc7.3.0`.

**NOTE**

This step is performed only when the compilation environment after the GCC upgrade is required.

----End

### 6.7 Configuring a System Network Proxy

The following procedure is a general method for configuring a network proxy. It may not be applicable to all network environments. The method of configuring the network proxy depends on the actual network environment.

**Prerequisites**

- Ensure that the network cable of the server is connected and the proxy server can connect to the external network.
- The configuration proxy is based on the condition that the server is located on an intranet and cannot be directly connected to the external network.

**Configuring a System Network Proxy**

**Step 1** Log in to the user environment as the root user.

**Step 2** Run the following command to edit the `/etc/profile` file:

```bash
vi /etc/profile
```

Add the following content to the file, save the file, and exit:

```bash
export http_proxy="http://user:password@proxyserverip:port"
export https_proxy="http://user:password@proxyserverip:port"
```

In the preceding commands, `user` indicates the username on the intranet, `password` indicates the user password, `proxyserverip` indicates the IP address of the proxy server, and `port` indicates the port number.
Step 3  Run the following command to make the configuration take effect.
source /etc/profile

Step 4  Run the following command to check whether the external network is connected:
wget www.baidu.com

If the HTML file can be downloaded, the server is connected to the external network successfully.

----End
7 Reference

7.1 Parameters

One-click installation is supported in the command line. You can combine the commands and select arguments as required. All arguments are optional.

The installation command format is: ./*.run [options]

For details, see Table 7-1.

NOTICE

If the parameters queried by running the ./*.run --help command are not described in the following table, they are reserved or applicable to other chip versions and therefore can be ignored.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--help</td>
<td>Queries help information.</td>
</tr>
<tr>
<td>--version</td>
<td>Queries version information.</td>
</tr>
<tr>
<td>--info</td>
<td>Queries software package construction information.</td>
</tr>
<tr>
<td>--list</td>
<td>Queries the software package list.</td>
</tr>
</tbody>
</table>

Table 7-1  Parameters supported by the installation package
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--check</td>
<td>Checks the consistency and integrity of software packages.</td>
</tr>
<tr>
<td>--quiet</td>
<td>Silent installation, skipping interactive messages.</td>
</tr>
<tr>
<td>--noexec</td>
<td>Decompresses a software package to the current directory without running the installation script. This parameter is used together with <code>--extract=&lt;path&gt;</code>. The format is as follows: <code>--noexec --extract=&lt;path&gt;</code></td>
</tr>
<tr>
<td>--extract=&lt;path&gt;</td>
<td>Decompresses a software package to a specified directory.</td>
</tr>
<tr>
<td>--tar arg1 [arg2 ...]</td>
<td>Runs the <code>tar</code> command on the software package. Use the arguments following <code>tar</code> as the command arguments. For example, the <code>--tar xvf</code> command indicates that the RUN package will be decompressed to the current directory.</td>
</tr>
<tr>
<td>--install</td>
<td>Installs a software package. You can specify the installation path <code>--install-path=&lt;path&gt;</code> or use the default installation path.</td>
</tr>
<tr>
<td>--install-path=&lt;path&gt;</td>
<td>Specifies the installation path. If you do not specify the installation path, the default installation path is used.</td>
</tr>
<tr>
<td>--uninstall</td>
<td>Uninstalls the software that has been installed.</td>
</tr>
<tr>
<td>--upgrade</td>
<td>Upgrades the software that has been installed. The system automatically checks the version number. If the version number is not in ascending order, the upgrade cannot be performed.</td>
</tr>
<tr>
<td>--devel</td>
<td>Installs a software package in development mode, that is, install only the files required by the development environment.</td>
</tr>
</tbody>
</table>

### 7.2 Ascend Chip Driver and Firmware Related Information
7.2.1 Commands and Arguments

Description

One-click installation is supported in the command line. You can select parameters as required to complete the installation. All parameters are optional.

Installation command format: `./*.run [options]`

For details, see Table 7-2.

---

**NOTICE**

If the parameters queried by running the `./*.run --help` command are not described in the following table, this parameter is reserved or applies to other chip versions. You do not need to pay attention to this parameter.

---

Table 7-2 Parameters supported by the installation packages

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--run</td>
<td>Indicates the running mode, which installs only the files required in a running scenario.</td>
</tr>
<tr>
<td>--devel</td>
<td>Indicates the development mode, which contains the header files that users need to use for development. The firmware subpackage does not support this parameter.</td>
</tr>
<tr>
<td>--full</td>
<td>Indicates the full installation mode, which installs all files.</td>
</tr>
</tbody>
</table>
| --install-username=| Initial installation: You can specify the running user name. Otherwise, HwHiAiUser is used by default. Overwrite: The user name used in the last installation is adopted. **NOTE**  
  - You are not advised to specify user root for security considerations.  
  - The user name and user group cannot be specified during firmware installation. The user name and user group of the driver is used.  
  - This parameter must be used together with --install-usergroup=, and the value of username must be the same as that of the created user (4.3.1 Before You Start). |
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--install-usergroup=&lt;usergroup&gt;</td>
<td>Initial installation: You can specify the running user group name. Otherwise, HwHiAiUser is used by default. Overwrite: The user group name used in the last installation is adopted. <strong>NOTE</strong> This parameter must be used together with --install-username=&lt;username&gt;, and the value of usergroup must be the same as that of the created user group created ([4.3.1 Before You Start]).</td>
</tr>
</tbody>
</table>
| --install-path=<path> | Specifies the installation directory. Otherwise:  
  - For installation as user root, the default installation path is /usr/local/Ascend.  
  - For installation as a non-root user, the default installation path is ${HOME}/Ascend.  
  The running user must have the read and write permissions on the specified installation path. **NOTE** The firmware installation path cannot be specified. The driver installation path is shared. |
| --reset             | Restores the initial configuration. (This parameter applies only to the firmware subpackage.)  
  - The **--full --reset** parameter is restored to the initial configuration through installation and takes effect immediately after the system is restarted. If the installation fails, the initial configuration fails to be restored.  
  - The **--upgrade --reset** parameter is restored to the initial configuration through upgrade and takes effect immediately after the system is restarted. If the upgrade fails, the initial configuration fails to be restored.  
  After the initial configuration is restored, the original personal configuration data on the device becomes invalid and needs to be reconfigured. **NOTE**  
  - The **reset** parameter can be used only in **--full --reset** and **--upgrade --reset** modes. It cannot be used together with other parameters or independently.  
  - After the command with the **reset** parameter is executed and before the restart takes effect, do not access the device through SSH or SCP. Otherwise, the initialization configuration fails to be restored. |
<p>| --uninstall         | Uninstalls.                                                                                                                                 |
| --noexec            | Does not run the installation script. This parameter is used together with the <strong>--extract=path</strong> parameter. Format: --noexec --extract=path |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--extract=path</td>
<td>Decompresses the installation package to a specified directory.</td>
</tr>
<tr>
<td>--upgrade</td>
<td>Performs upgrade, which takes effect immediately. The upgrade can be performed only in the path where a software package is stored.</td>
</tr>
<tr>
<td>--help/-h</td>
<td>Displays the help information.</td>
</tr>
<tr>
<td>--check</td>
<td>Check the integrity of a software package.</td>
</tr>
<tr>
<td>--version</td>
<td>Display the version information.</td>
</tr>
<tr>
<td>--tar arg1 [arg2 ...]</td>
<td>Runs the <code>tar</code> command on the software package. Use the arguments following <code>tar</code> as the command arguments. For example, the <code>--tar xvf</code> command indicates that the .run package will be decompressed to the current directory.</td>
</tr>
<tr>
<td>--list</td>
<td>Lists the files in a software package.</td>
</tr>
<tr>
<td>--info</td>
<td>Displays detailed information of a package.</td>
</tr>
<tr>
<td>--quiet</td>
<td>Indicates the silent installation, which skips interactive messages.</td>
</tr>
<tr>
<td>--pylocal</td>
<td>Installs Python to a local path.</td>
</tr>
</tbody>
</table>

Example:

- Installation in **full** mode
  - Unspecified installation directory: `./*.run --full`
  - Specified installation directory: `./*.run --full --install-path=installation directory`

- Installation in **run** mode
  - Unspecified installation directory: `./*.run--run`
  - Specified installation directory: `./*.run --full --install-path=installation directory`

### 7.2.2 Scripts

The scripts in **Table 7-3** are automatically invoked to install, uninstall, and update the .run software package. The script storage path is subject to the path specified during the installation. The following uses the default path `/usr/local/Ascend` as an example.
### Table 7-3 Scripts

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>run_firmware_install.sh</td>
<td>/usr/local/Ascend/firmware/</td>
<td>Installs firmware.</td>
</tr>
<tr>
<td>run_firmware_uninstall.sh</td>
<td>/usr/local/Ascend/firmware/</td>
<td>Uninstalls firmware.</td>
</tr>
<tr>
<td>run_firmware_upgrade.sh</td>
<td>/usr/local/Ascend/firmware/</td>
<td>Updates firmware (in immediate or delayed mode).</td>
</tr>
<tr>
<td>install_common_parser.sh</td>
<td>/usr/local/Ascend/firmware/</td>
<td>Stops five background processes, including the ada process.</td>
</tr>
<tr>
<td>install.sh</td>
<td>/usr/local/Ascend/firmware/</td>
<td>Parses the filelist.csv file.</td>
</tr>
<tr>
<td>host_sys_stop.sh</td>
<td>/usr/local/Ascend/</td>
<td>Entry script for installing the firmware installation package.</td>
</tr>
<tr>
<td>host_sys_init.sh</td>
<td>/usr/local/Ascend/</td>
<td>Stops or starts five background processes, including the ada process.</td>
</tr>
<tr>
<td>host_server_setup.sh</td>
<td>/usr/local/Ascend/</td>
<td>Obtains the system configuration information and invokes the host_sys_init.sh script.</td>
</tr>
<tr>
<td>host_servers_remove.sh</td>
<td>/usr/local/Ascend/</td>
<td>Removes startup configuration including host_sys_init.sh.</td>
</tr>
<tr>
<td>install.sh</td>
<td>/usr/local/Ascend/driver/script/</td>
<td>Entry script for installing the driver installation package</td>
</tr>
<tr>
<td>run_driver_upgrade.sh</td>
<td>/usr/local/Ascend/driver/script/</td>
<td>Updates driver (in immediate or delayed mode).</td>
</tr>
<tr>
<td>run_driver_dkms_install.sh</td>
<td>/usr/local/Ascend/driver/script/</td>
<td>Loads modules to DKMS.</td>
</tr>
<tr>
<td>run_driver_dkms_uninstall.sh</td>
<td>/usr/local/Ascend/driver/script/</td>
<td>Uninstalls modules from DKMS.</td>
</tr>
<tr>
<td>run_driver_install.sh</td>
<td>/usr/local/Ascend/driver/script/</td>
<td>Installs, uninstalls, or updates (in immediate or delayed mode) driver.</td>
</tr>
<tr>
<td>run_driver_uninstall.sh</td>
<td>/usr/local/Ascend/driver/script/</td>
<td>Uninstalls driver.</td>
</tr>
</tbody>
</table>
7.2.3 Tools

**Step 1** Log in to the operating environment as the running user (default user: HwHiAiUser) specified during the software package installation.

**Step 2** Switch to the installation path of the software package and run the commands in Table 7-4 to invoke tools. The following uses the default installation path of the driver as an example.

### Table 7-4 Related tools

<table>
<thead>
<tr>
<th>Name</th>
<th>Path</th>
<th>Function</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>hdcd</td>
<td>/usr/local/Ascend/driver/tools</td>
<td>Starts the background process for file transfer between the host and device.</td>
<td>./hdcd</td>
</tr>
<tr>
<td>ADA</td>
<td>/usr/local/Ascend/driver/tools</td>
<td>Starts the IDE daemon process.</td>
<td>./ada</td>
</tr>
<tr>
<td>Name</td>
<td>Path</td>
<td>Function</td>
<td>Command</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>sklogd</td>
<td>/usr/local/Ascend/driver/tools</td>
<td>Starts background processes and records kernel-mode logs. This command can be executed only by user root.</td>
<td>./sklogd</td>
</tr>
<tr>
<td>slogd</td>
<td>/usr/local/Ascend/driver/tools</td>
<td>Starts background process and record user-mode logs.</td>
<td>./slogd</td>
</tr>
<tr>
<td>Name</td>
<td>Path</td>
<td>Function</td>
<td>Command</td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| upgrade-tool| /usr/local/Ascend/driver/tools | Views firmware versions and upgrades one or more pieces of firmware. This command can be executed only by user root. | • Lists all devices: `./upgrade-tool --mini_devices`
• Obtains the version of a specified device: `./upgrade-tool --device_index <dev_id> --system_version`
• Obtains the component information about a specified device: `./upgrade-tool --device_index <dev_id> --components`
• Queries the version of a device component: `./upgrade-tool --device_index <dev_id> --component <type> --version`
• Queries the device status: `./upgrade-tool --device_index <dev_id> --status`
• Updates the firmware package of a device: Switch to the `/usr/local/Ascend/firmware` directory and run the following command: `./upgrade-tool`  
  `--device_index <dev_id> --component <type> --path <firmwareee_path>`
  The parameters involved in the above command are as follows:
  - `--mini_devices`: list of all devices.
  - `--device_index`: device ID. A value ranges from 0 to 63 or -1. The value ranging from 0 to 63 indicates the device with the corresponding ID.
### 7.3 HCCN Tool Configuration Command Reference

#### 7.3.1 Introduction

This document describes the external interfaces of the cluster network tool HCCN, including interfaces for configuring the IP address and gateway, configuring the IP address of the network detection object, and querying the Link Layer Discovery Protocol (LLDP) of an RDMA over Converged Ethernet (RoCE) network interface card (NIC).

Before using this tool, ensure that the Driver component has been installed and the following requirements are met.

Before using this tool, ensure that the Driver component has been installed and you have logged in as the root user.
7.3.2 Interfaces

7.3.2.1 Displays HCCN help information

Command Prototype

```
hccn_tool -h
```

Command Function
Displays HCCN help information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h</td>
<td>Help information obtaining</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -h
```

7.3.2.2 Sets the IP address and subnet mask of an RoCE NIC

Command Prototype

```
hccn_tool [-i %d] -ip -s [address %s] [netmask %s]
```

Command Function
Sets the IP address and subnet mask of an RoCE NIC.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-ip</td>
<td>IP attribute</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>address</td>
<td>IP address</td>
</tr>
<tr>
<td>netmask</td>
<td>Subnet mask</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```
hccn_tool -i 0 -ip -s address 192.168.1.10 netmask 255.255.255.0
```

**Restrictions**

The OS on each device of the AI Server manages four Ascend AI Processors. You need to configure different IP address segments for the four NICs on each OS.

When the IP address is configured for the first time, the link status changes to Down and then to Up in 15 seconds.

### 7.3.2.3 Sets the default gateway of an RoCE NIC

**Command Prototype**

```
hccn_tool [-i %d] -gateway -s [gateway %s]
```

**Command Function**

Sets the default gateway of an RoCE NIC.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-gateway</td>
<td>Gateway attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>gateway</td>
<td>Gateway IP address</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
Other values: failure

Example

hccn_tool -i 0 -gateway -s gateway 192.168.1.1

7.3.2.4 Obtains the default gateway of the RoCE NIC

Command Prototype

hccn_tool [-i %d] -gateway -g

Command Function

Obtains the default gateway of the RoCE NIC.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-gateway</td>
<td>Gateway attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

hccn_tool -i 0 -gateway -g

7.3.2.5 Sets the IP address of the network detection object

Command Prototype

hccn_tool [-i %d] -netdetect -s [address %s]

Command Function

Sets the IP address of the network detection object.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>-netdetect</td>
<td>IP address attribute of the network detection object</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>address</td>
<td>IP address</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```bash
hccn_tool -i 0 -netdetect -s address 192.168.1.11
```

### 7.3.2.6 Obtains the MAC address of an RoCE NIC

**Command Prototype**

```bash
hccn_tool [-i %d] -mac -g
```

**Command Function**

Obtains the MAC address of an RoCE NIC.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-mac</td>
<td>MAC attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```bash
hccn_tool -i 0 -mac -g
```
7.3.2.7 Queries the LLDP information

**Command Prototype**

```
hccn_tool [-i %d] -lldp -g
```

**Command Function**

Queries the LLDP information.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-lldp</td>
<td>LLDP attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```
hccn_tool -i 0 -lldp -g
```

7.3.2.8 Restores the configuration of the 100GE NIC

**Command Prototype**

```
hccn_tool [-i %d][ -cfg recovery]
```

**Command Function**

Restores the configuration of the 100GE NIC, including the IP address, subnet mask, gateway address, and network detection IP address.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-cfg</td>
<td>CFG attribute</td>
</tr>
<tr>
<td>-recovery</td>
<td>Restoration flag</td>
</tr>
</tbody>
</table>
Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -cfg recovery
```

7.3.2.9 Shuts down a link

Command Prototype

```
hccn_tool [-i %d] -link -s down
```

Command Function

Shuts down a link.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-link</td>
<td>Link attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>down</td>
<td>Shutdown attribute.</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -link -s down
```

7.3.2.10 Obtains the status of a link

Command Prototype

```
hccn_tool [-i %d] -link -g
```

Command Function

Obtains the status of a link.
**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-link</td>
<td>Link attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```
hccn_tool -i 0 -link -g
```

### 7.3.2.11 Sets the MTU of an RoCE NIC

**Command Prototype**

```
hccn_tool [-i %d] -mtu -s [size %d]
```

**Command Function**

Sets the maximum transmission unit (MTU) of an RoCE NIC.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-mtu</td>
<td>MTU attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>size</td>
<td>MTU value. Value range: 68–9702</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```
hccn_tool -i 0 -mtu -s size 4096
```
7.3.2.12 Obtains the MTU of an RoCE NIC

Command Prototype

```
hccn_tool [-i %d] -mtu -g
```

Command Function

Obtains the MTU of an RoCE NIC.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-mtu</td>
<td>MTU attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -mtu -g
```

7.3.2.13 Obtains the network health status

Command Prototype

```
hccn_tool [-i %d] -net_health -g
```

Command Function

Obtains the network health status.

Status mapping:

- 0: health
- 1: socket failed
- 2: RX timeout
- 3: unreachable
- 4: sock response timeout
- 5: detection error
- 6: initialization in progress
- 7: thread error
- Other values: failure

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-net_health</td>
<td>Network health status</td>
</tr>
</tbody>
</table>
### 7.3.2.14 Sets the MAC address of an RoCE NIC

#### Command Prototype

```
hccn_tool [-i %d] -mac -s [mac %s]
```

#### Command Function

Sets the MAC address of an RoCE NIC.

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-mac</td>
<td>MAC address attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>mac</td>
<td>MAC address</td>
</tr>
</tbody>
</table>

#### Returns

- **0**: success
- Other values: failure

#### Example

```
hccn_tool -i 0 -net_health -g
```

```
hccn_tool -i 0 -mac -s mac 26:e5:f0:1c:39:c9
```

### 7.3.2.15 Sets the DSCP to TC mapping

#### Command Prototype

```
hccn_tool [-i %d] -dscp_to_tc -s [dscp %u] [tc %u]
```

#### Example

```
hccn_tool -i 0 -mac -s mac 26:e5:f0:1c:39:c9
```
Command Function

Sets the differentiated services code point (DSCP) to traffic class (TC) mapping.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-dscp_to_tc</td>
<td>DSCP to TC mapping attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>dscp</td>
<td>DSCP value. Value range: 0–63</td>
</tr>
<tr>
<td>tc</td>
<td>TC value. Value range: 0–7</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -dscp_to_tc -s dscp 0 tc 0
```

7.3.2.16 Obtains the DSCP to TC mapping

Command Prototype

```
hccn_tool [-i %d] -dscp_to_tc -g [dscp %u]
```

Command Function

Obtains the DSCP to TC mapping.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-dscp_to_tc</td>
<td>DSCP to TC mapping attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
<tr>
<td>dscp</td>
<td>DSCP value. Value range: 0–63</td>
</tr>
</tbody>
</table>
Returns

- 0: success
- Other values: failure

Examples

Obtains all DSCP to TC mappings: `HCCN tool -i 0 -dscp_to_tc -g`

Obtains the TC mapping of a specified DSCP: `HCCN tool -i 0 -dscp_to_tc -g dscp 0`

7.3.2.17 Obtains the ARP table

Command Prototype

```
hccn_tool [-i %d] -arp -g
```

Command Function

Obtains the Address Resolution Protocol (ARP) table.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-arp</td>
<td>ARP attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -arp -g
```

7.3.2.18 Obtains the routing table

Command Prototype

```
hccn_tool [-i %d] -route -g
```

Command Function

Obtains the routing table.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-route</td>
<td>Routing table attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -route -g
```

7.3.2.19 Obtains PCI device information.

Command Prototype

```
hccn_tool [-i %d] -pci -g
```

Command Function

Obtains PCI device information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-pci</td>
<td>PCI device attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -pci -g
```
7.3.2.20 Obtains QPC Information

Command Prototype

```
hccn_tool [-i %d] -context -g [qpc %d]
```

Command Function

Obtains queue pair context (QPC) information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-context</td>
<td>RoCE context attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
<tr>
<td>qpc</td>
<td>number.</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -context -g qpc 0
```

7.3.2.21 Obtains AEQC information

Command Prototype

```
hccn_tool [-i %d] -context -g [aeqc %d]
```

Command Function

Obtains Asynchronous Event Queue Context (AEQC) information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-context</td>
<td>RoCE context attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
<tr>
<td>aeqc</td>
<td>number.</td>
</tr>
</tbody>
</table>
Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -context -g aeqc 0
```

7.3.2.22 Obtains CEQC information

Command Prototype

```
hccn_tool [-i %d] -context -g [ceqc %d]
```

Command Function

Obtains Completion Event Queue Context (CEQC) information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-context</td>
<td>RoCE context attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
<tr>
<td>ceqc</td>
<td>number.</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -context -g ceqc 0
```

7.3.2.23 Obtains CQC information

Command Prototype

```
hccn_tool [-i %d] -context -g [cqc %d]
```

Command Function

Obtains Completion Queue Context (CQC) information.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-context</td>
<td>RoCE context attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
<tr>
<td>cqc</td>
<td>number.</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -context -g cqc 0
```

7.3.2.24 Obtains MPT information

Command Prototype

```
hccn_tool [-i %d] -context -g [mpt %d]
```

Command Function

Obtains Memory Protection Table (MPT) information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-context</td>
<td>RoCE context attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
<tr>
<td>mpt</td>
<td>number.</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -context -g mpt 0
```
7.3.2.25 Obtains optical module information

Command Prototype

```
hccn_tool [-i %d] -optical -g
```

Command Function

Obtains optical module information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-optical</td>
<td>Optical module attribute.</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -optical -g
```

Restrictions

For copper cables, the temperature and power information is unavailable.

7.3.2.26 Obtains register information

Command Prototype

```
hccn_tool [-i %d] -reg -g
```

Command Function

Obtains register information.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-reg</td>
<td>Register attribute</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```
hccn_tool -i 0 -reg -g
```

### 7.3.2.27 Obtains the IP address and subnet mask

**Command Prototype**

```
hccn_tool [-i %d] -ip -g
```

**Command Function**

Obtains the IP address and subnet mask.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-ip</td>
<td>IP attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```
hccn_tool -i 0 -ip -g
```

### 7.3.2.28 Obtains statistics

**Command Prototype**

```
hccn_tool [-i %d] -stat -g
```
**Command Function**

Obtains statistics.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-stat</td>
<td>Statistics attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```bash
hccn_tool -i 0 -stat -g
```

### 7.3.2.29 Sets the bandwidth limit of an RoCE NIC

**Command Prototype**

```bash
hccn_tool [-i %d] -shaping -s [bw_limit %d]
```

**Command Function**

Sets the bandwidth limit of an RoCE NIC.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-shaping</td>
<td>Bandwidth limit attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>-bw_limit</td>
<td>Bandwidth (Mbit/s). Value range: 10000–100000</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure
Example

hccn_tool -i 0 -shaping -s bw_limit 100000

7.3.2.30 Obtains the bandwidth limit of an RoCE NIC

Command Prototype

hccn_tool [-i %d] -shaping -g

Command Function

Obtains the bandwidth limit of an RoCE NIC.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-shaping</td>
<td>Bandwidth limit attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

hccn_tool -i 0 -shaping -g

7.3.2.31 Restores the configuration of all RoCE NICs

Command Prototype

hccn_tool -a -cfg recovery

Command Function

Restores the configuration of all RoCE NICs.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>All RoCE NICs</td>
</tr>
<tr>
<td>-cfg</td>
<td>CFG attribute</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-recovery</td>
<td>Restoration flag</td>
</tr>
</tbody>
</table>

### Returns

- 0: success
- Other values: failure

### Example

```
hccn_tool -a -cfg recovery
```

### 7.3.2.32 Sets TLS

#### 7.3.2.32.1 Presets or replaces the certificate suite

**Command Prototype**

```
hccn_tool [-i %d] -tls -s [path %s] [pri %s] [pub %s] [ca1 %s] [ca2 %s] ... [ca14 %s] [crl %s]
```

**Command Function**

Presets or replaces the certificate suite.

**NOTE**

- Initial presetting typically happens in the equipment test phase. You can run this command to replace the existing certificates despite their sources.
- The certificate level is configurable. A minimum of two levels (including only the device and root certificates) and a maximum of 15 levels (including the device and levels 1–14 root certificate) can be configured. By default, 3-level certificates (including the device and levels 1–2 root certificates) are configured.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-tls</td>
<td>Transport Layer Security (TLS) attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>path</td>
<td>Path for storing the certificates, private keys, and certificate revocation list (CRL). You are advised to pass an absolute path.</td>
</tr>
<tr>
<td>pri</td>
<td>Name of the private key file</td>
</tr>
<tr>
<td>pub</td>
<td>Name of the device certificate file</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>ca1</td>
<td>Name of the level-1 root certificate file</td>
</tr>
<tr>
<td>ca2</td>
<td>Name of the level-2 root certificate file</td>
</tr>
<tr>
<td>caN</td>
<td>Name of the level-(N) root certificate file ((0 &lt; N &lt; 15))</td>
</tr>
<tr>
<td>crl</td>
<td>CRL file name</td>
</tr>
</tbody>
</table>

**Returns**

- **0**: success
- Other values: failure

**Example**

```
hccn_tool -i 0 -tls -s path /home/HwHiAiUser pri.pem pub pem ca1 ca1.pem ca2 pem crl xxx.crl
```

This example attempts to set the TLS certificate suite for device 0, where all certificate suite files are stored in the `/home/HwHiAiUser` directory, the private key file is `pri.pem`, the device certificate file is `pub.pem`, the level-1 root certificate file is `ca1.pem`, the level-2 root certificate file is `ca2.pem`, the CRL file is `xxx.crl`, and the password file is `xxx`. After this command is run, the message "Please enter pass phrase for pri key" is displayed. You need to enter the password of the private key. The password must be the same as that used for generating the private key.

**Restrictions**

1. A file name allows a maximum of 255 bytes, including the suffix. For example, the length of file name `pub.pem` is 7 bytes.
2. File size:
   - A certificate file is up to 2048 bytes.
   - The private key is up to 4096 bytes.
   - A CRL is up to 20,480 bytes.
3. A Certificate Authority (CA) name is up to 64 bytes.
4. Only certificates and private keys must be PEM-encoded and the CRL must be in the CRL format.
5. The certificates must be in X.509v3 format. For a root CA or intermediate CA certificate, the "Basic Constraints" extensions must be "CA", and the "Key Usage" extensions must contain the "Certificate Signature". The digital certificates and the CRL must be within the validity periods.
6. A RSA private key with the length of [2048, 4096] is supported. The private key must be in ciphertext, and the encryption algorithm of the private key can only be AES-256.
7. The private key, certificates, and CRL must match each other. Otherwise, replacement fails.
8. You can replace only a certificate or the CRL. Ensure that the certificate matches the CRL before using TLS.

9. The private key and device certificate must be replaced together.

10. If only some certificates need to be replaced (usually device certificates and some low-level certificates). A larger N in caN indicates a lower level. Ensure each lower-level certificate is certified by its higher-level certificate. Otherwise, the certificate chain verification fails, causing configuration failures.

11. The password is entered in interactive mode and is not displayed in plaintext. The password is a string of 8 to 15 characters and must contain at least two types of the following characters: uppercase letters, lowercase letters, digits, and special characters.

12. When presetting or updating a private key, ensure that the password is the same as that entered for generating the private key.

13. After the configuration is complete, delete the plaintext private key. [warning] Please delete the pri key and pwd file after updating for security considerations." will be displayed.

14. The TLS certificate suite configured using this command is used for TLS authentication during cluster communication. If TLS authentication fails, a log message is recorded. The product side needs to analyze the log message and report an alarm.

7.3.2.32.2 Enables or disables TLS

Command Prototype

```
hccn_tool [-i %d] -tls -s [enable %d]
```

Command Function

Enables or disables TLS.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-tls</td>
<td>TLS attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>enable</td>
<td>Enable 0: disabled; 1: enabled</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure
Example

```
hccn_tool -i 0 -tls -s enable 1
```

Restrictions

This command can be used only after the certificate suite is preset. The configuration is non-volatile and will not be lost after reboot or power-off.

The TLS function is enabled by default before delivery. If you run this command to disable the TLS function, all information is transferred in plaintext in cluster communication, which is easy to be eavesdropped, tampered, or forged.

7.3.2.32.3 Sets the TLS certificate expiry alarm threshold

Command Prototype

```
hccn_tool [-i %d] -tls -s [alarm %d]
```

Command Function

Sets the TLS certificate expiry alarm threshold.

**NOTE**

The system checks all certification expiration dates on a daily basis. If any certificate is about to expire in the specified days (7 to 180 days, 60 days by default), an alarm is recorded in the log, including the certificate name, certificate expiration date, and error description.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-tls</td>
<td>TLS attribute</td>
</tr>
<tr>
<td>-s</td>
<td>Setting flag</td>
</tr>
<tr>
<td>alarm</td>
<td>Number of days before any certificate expires. Value range: 7–180</td>
</tr>
</tbody>
</table>

Returns

- 0: success
- Other values: failure

Example

```
hccn_tool -i 0 -tls -s alarm 60
```
**Restrictions**

This command can be used only after the certificate suite is preset. The configuration is non-volatile and will not be lost after reboot or power-off.

If a certificate is about to expire, an alarm message is recorded. The product side needs to analyze the log message and report an alarm.

If a certificate expires, TLS link establishment fails and a log message is recorded.

### 7.3.2.32.4 Obtains TLS information

**Command Prototype**

```
hccn_tool [-i %d] -tls -g
```

**Command Function**

Obtains TLS enable status and certificate expiry alarm threshold.

Obtains basic information about each TLS certificate, including the CA, start date, and end date.

The certificate information is displayed in the sequence of the certificate hierarchy: device certificate, level-1 root certificate, level-2 root certificate, and so on.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-i</td>
<td>Device ID. Value range: 0–7</td>
</tr>
<tr>
<td>-tls</td>
<td>TLS attribute</td>
</tr>
<tr>
<td>-g</td>
<td>Obtaining flag</td>
</tr>
</tbody>
</table>

**Returns**

- 0: success
- Other values: failure

**Example**

```
hccn_tool -i 0 -tls -g
```

**Restrictions**

This command can be used only after the certificate suite is preset.

If a certificate or CRL is about to expire, a warning message is displayed. If a certificate or CRL as expired, an error message is displayed.
7.4 Performing Fault Diagnostics

Function

Obtain the chip health information and perform computing power, power consumption, and bandwidth tests on the chip, and determine the health status of the current product based on the test results.

Commands for Querying Test Parameters

You can run either of the following commands to list the parameters of the fault diagnostics command:

```
ascend-dmi -dg -h
ascend-dmi -dg --help
```

Table 7-5 describes the parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory / Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>[-dg, diagnosis]</td>
<td>Performs a fault diagnostic test of the entire card.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>[-c, --card]</td>
<td>Specifies a card for diagnostics. If this parameter is not specified, fault diagnostics will be performed on all cards by default. You can run the <code>ascend-dmi info</code> command to obtain the numbers of the installed cards.</td>
<td>Optional</td>
</tr>
<tr>
<td>[-l, --level]</td>
<td>Specifies the diagnosis level. If this parameter is not specified, a level 0 diagnosis will be performed by default. The following options are available: • 0: performs a diagnosis test on the chip health. • 1: performs a diagnosis test on the power consumption, bandwidth, and computing power of the card.</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Leave `-c` and `-l` unspecified

Performs a level 0 diagnosis of all cards. Optional
NOTE

To ensure the correctness and accuracy of the test result, perform the fault diagnostic test separately.

Example

The command output on an inference server is similar to that on a training server. The following uses the screenshots on an inference server as an example.

- Perform a level 0 diagnosis test on card No. 65.
  
  \texttt{ascend-dmi -dg -c 65 -l 0}

  In the command, 65 is the number of the card to be tested. You can run the \texttt{ascend-dmi info} command to obtain the card number information.

  If information shown in Figure 7-1 is displayed, the tool is running properly. For details about the parameters, see Table 7-6.

  \textbf{Figure 7-1} Fault diagnosis example (level 0)

  \begin{verbatim}
  Card : 65
  Device
  Health : OK
  Device
  Health : OK
  Device
  Health : OK
  Device
  Health : OK
  
  \end{verbatim}

- Perform a level 1 diagnosis test on card No. 65.
  
  \texttt{ascend-dmi -dg -c 65 -l 1}

  In the command, 65 is the number of the card to be tested. You can run the \texttt{ascend-dmi info} command to obtain the card number information.

  If information shown in Figure 7-2 is displayed, the tool is running properly. For details about the parameters, see Table 7-6.
Figure 7-2 Fault diagnosis example (level 1)

Card: 05
Device: 4
Health: OK
Flops Test
  Duration(ms): 3766
  Computing Power(TFLOPS): 11.137291
Bandwidth Test
  Bandwidth(MB/s): Device To Device
  Bandwidth(MB/s): 45397.267791
  Bandwidth(MB/s): Device To Host
  Bandwidth(MB/s): 2936.311216
  Bandwidth Test
  Bandwidth(MB/s): Host To Device
  Bandwidth(MB/s): 2557.242812

Device: 5
Health: OK
Flops Test
  Duration(ms): 3766
  Computing Power(TFLOPS): 11.137291
Bandwidth Test
  Bandwidth(MB/s): Device To Device
  Bandwidth(MB/s): 45483.661086
  Bandwidth Test
  Bandwidth(MB/s): Device To Host
  Bandwidth(MB/s): 2921.020144
  Bandwidth Test
  Bandwidth(MB/s): Host To Device
  Bandwidth(MB/s): 2571.231763

Device: 6
Health: OK
Flops Test
  Duration(ms): 3766
  Computing Power(TFLOPS): 11.137291
Bandwidth Test
  Bandwidth(MB/s): Device To Device
  Bandwidth(MB/s): 45477.239995
  Bandwidth Test
  Bandwidth(MB/s): Device To Host
  Bandwidth(MB/s): 2928.073737
  Bandwidth Test
  Bandwidth(MB/s): Host To Device
  Bandwidth(MB/s): 2525.945231

Device: 7
Health: OK
Flops Test
  Duration(ms): 3766
  Computing Power(TFLOPS): 11.137291
Bandwidth Test
  Bandwidth(MB/s): Device To Device
  Bandwidth(MB/s): 45415.762753
  Bandwidth Test
  Bandwidth(MB/s): Device To Host
  Bandwidth(MB/s): 2884.494700
  Bandwidth Test
  Bandwidth(MB/s): Host To Device
  Bandwidth(MB/s): 2512.327214

Power Test
Max Power(W): 62,600,002
Average Power(W): 60,729,111
Max AI Core(%): 100
Average AI Core(%): 89
Max Temp(°C): 83
Average Temp(°C): 80
Max Voltage(V): 75
Average Voltage(V): 75
Table 7-6 Parameter description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Card</td>
<td>Indicates the card number.</td>
</tr>
<tr>
<td>Device</td>
<td>Indicates the chip number.</td>
</tr>
<tr>
<td>Health</td>
<td>Indicates the chip health status.</td>
</tr>
<tr>
<td>Flops Test</td>
<td>Indicates the computing power test.</td>
</tr>
<tr>
<td>Duration(ms)</td>
<td>Indicates the test duration in ms.</td>
</tr>
<tr>
<td>Computing Power(TFLOPS)</td>
<td>Indicates the computing power.</td>
</tr>
<tr>
<td>Bandwidth Test</td>
<td>Indicates the bandwidth test type.</td>
</tr>
<tr>
<td>Bandwidth(MB/s)</td>
<td>Indicates the bandwidth in MB/s.</td>
</tr>
<tr>
<td>Power Test</td>
<td>Indicates the power consumption test.</td>
</tr>
<tr>
<td>Max Power(W)</td>
<td>Indicates the maximum power consumption.</td>
</tr>
<tr>
<td>Average Power(W)</td>
<td>Indicates the average power consumption.</td>
</tr>
<tr>
<td>Max AI Core(%)</td>
<td>Indicates the maximum AI core usage.</td>
</tr>
<tr>
<td>Average AI Core(%)</td>
<td>Indicates the average AI core usage.</td>
</tr>
<tr>
<td>Max Temp(C)</td>
<td>Indicates the maximum temperature.</td>
</tr>
<tr>
<td>Average Temp(C)</td>
<td>Indicates the average temperature.</td>
</tr>
<tr>
<td>Max Voltage(V)</td>
<td>Indicates the maximum voltage.</td>
</tr>
<tr>
<td>Average Voltage(V)</td>
<td>Indicates the average voltage.</td>
</tr>
</tbody>
</table>

7.5 Performing a Software and Hardware Compatibility Test

Function

Test the hardware and software compatibility by obtaining the hardware information, architecture, driver version, firmware version, and software version information.

Commands for Querying Test Parameters

You can run either of the following commands to list the parameters of the compatibility test command:

```
ascend-dmi -c -h
```
ascend-dmi -c --help

Table 7-7 describes the parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>[-c, compatible]</td>
<td>Checks the compatibility of software and hardware versions.</td>
<td>Mandatory</td>
</tr>
<tr>
<td>[-p, --path]</td>
<td>Specifies the installation path of the software package to be checked.</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>The default path is used when the software package is installed by the root user. Therefore, you do not need to set this parameter. You need to set this parameter if the default installation path is not used. For example, if the software package is installed in the /home/xxx/Ascend directory, run the following command: ascend-dmi -c -p /home/xxx/Ascend</td>
<td></td>
</tr>
</tbody>
</table>

**NOTE**

The compatibility check tool checks the following software packages:
- Toolkit: development kit
- NNRT: offline inference engine
- NNAE: deep learning acceleration engine
- TFPlugin: framework plug-in
- Toolbox: utility
- NPU_Driver: driver
- NPU_Firmware: firmware

**Example**

Perform a software and hardware version compatibility test.

ascend-dmi -c
If the information shown in Figure 7-3 is returned by an inference server or the information shown in Figure 7-4 is returned by a training server, the tool is running properly. For details about the parameters, see Table 7-8.

**Figure 7-3** Software and hardware compatibility test example (inference server)

**Figure 7-4** Software and hardware compatibility test example (training server)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Information</td>
<td>Displays the system information.</td>
</tr>
<tr>
<td>Architecture</td>
<td>Indicates the architecture.</td>
</tr>
<tr>
<td>Product Type</td>
<td>Indicates the hardware type.</td>
</tr>
<tr>
<td>Package Information</td>
<td>Displays the package information.</td>
</tr>
<tr>
<td>NPU_Driver</td>
<td>Indicates the NPU driver version.</td>
</tr>
<tr>
<td>NPU_Firmware</td>
<td>Indicates the NPU firmware version.</td>
</tr>
<tr>
<td>Installed Software</td>
<td>Lists the software installed.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------</td>
</tr>
<tr>
<td>Compatibility Check Result</td>
<td>Displays the compatibility check result.</td>
</tr>
<tr>
<td>Package</td>
<td>Indicates the package name.</td>
</tr>
<tr>
<td>Version</td>
<td>Indicates the version.</td>
</tr>
<tr>
<td>Status</td>
<td>Indicates the status.</td>
</tr>
<tr>
<td>Dependencies</td>
<td>Lists the dependency.</td>
</tr>
<tr>
<td>Unknown Packages</td>
<td>Lists the packages that cannot be identified.</td>
</tr>
<tr>
<td>WARNING</td>
<td>Warning information.</td>
</tr>
</tbody>
</table>

**NOTE**

When an error is detected, the command output may contain the following information:

- **Unknown Packages**: lists all packages that do not comply with the formats recorded in the system.
- **WARNING**: The software package of an unknown version is installed or the Ascend-DMI tool needs to be updated.
<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020-07-21</td>
<td>This issue is the first official release.</td>
</tr>
</tbody>
</table>