Core Flight System
Command and Data Dictionary Utility
Installation Guide

Engineering Directorate
Software, Robotics, and Simulation Division

Version 2.1.8
April 2024

National Aeronautics and Space Administration
Lyndon B. Johnson Space Center
Houston, Texas 77058-3696

Table of Contents

1.0 Overview

The Core Flight System Command and Data Dictionary (CCDD) is a software tool for managing the command and telemetry data for CFS and CFS applications. CCDD is written in Java™ and interacts with a PostgreSQL database, so it can be used on any operating system that supports the Java Runtime Environment (JRE) and PostgreSQL. CCDD is released as open source software under the NASA Open Source Software Agreement, version 1.3, and is hosted on GitHub.

This document describes the installation of the CCDD and its software dependencies.

Questions or comments concerning this document or the CCDD application can be addressed to:

Johnson Space Center
Software, Robotics, and Simulation Division
Spacecraft Software Engineering Branch, Mail Code ER6
Houston, TX 77058

2.0 Software Dependencies

The following open source software components are used by the CCDD tool, and hence, must be installed in order to run the CCDD.

2.1 Java

To install Java, go to www.java.com and locate the installation instructions appropriate for the operating system on which the application is to be run.

2.2 PostgreSQL

The PostgreSQL relational database management system is available for download from www.postgresql.org. The format appropriate for the target operating system must be used.

Once installed, PostgreSQL must be configured prior to use by the application. Configuration includes setting up the PostgreSQL server as a background service, creating database users and roles within the PostgreSQL server, and setting the desired level of password authentication. Extensive information on configuring PostgreSQL is available from www.postgresql.org.

2.3 Scripting Languages

CCDD supports the use of JVM-based scripting languages. At least one of these languages must be installed for the application to make use of CCDD’s project-data-to-script-language interface. Only the scripting language(s) intended for use with the application need to be installed.

The tool was tested with five of the available languages: JavaScript, Python, Ruby, Groovy, and Scala. Details are provided in this in subsequent sections of the CCDD developers guide on the use of these five scripting languages; installation and use of other languages should be similar. The CCDD repository provides examples of common scripts in the JavaScript, Python, Ruby, and Groovy languages.

The scripting languages are not part of the CCDD package and must be installed separately on the platform from which the CCDD application is launched. The following links can be used to find further information on downloading and installing the scripting languages.

- *JavaScript®* is part of the JRE download and installation from www.java.com, so no further installation is necessary to use this scripting language. More information on JavaScript can be found at developer.mozilla.org.
- *Python™* scripting is implemented in one of two ways:
    - Using *Jython*, the Python implementation for Java. Jython can be downloaded from www.jython.org. Jython only supports Python 2.7.
    - Using *Py4J*, a bridge between CCDD and a Python script. Py4J can be downloaded from py4j.org. Py4J supports Python version 2.7 and 3.
- *Ruby* scripting is implemented using *JRuby*, which implements Ruby in Java. JRuby is available for download from jruby.org.
- *Groovy* can be downloaded from www.groovy-lang.org.
- *Scala* can be downloaded from www.scala-lang.org.

3.0 Installation & Setup

3.1 PostgreSQL

This section describes the installation and setup of PostgreSQL. This section covers installation and setup for Linux Redhat (including Fedora/CentOS/etc.) and Debian (including Ubuntu/etc.) distributions. For Windows an interactive installer can be downloaded from www.postgresql.org.

3.1.1 Installation

Execute the following commands as super user. Commands are provided for Redhat-based (indicated with ®) and Debian-based (indicated with ⊕) Linux versions. Commands that are unmarked work with any Linux version.

1. Pull the latest versions of all installed components

    ```
    sudo yum update  ®
    sudo apt update  ⊕
    ```

2. Install the required PostgreSQL packages

    ```
    sudo yum install postgresql##-server postgresql##-contrib ®
    sudo apt install postgresql##-server-dev-all postgresql##-contrib ⊕
    ```

    ## is the PostgreSQL version number (e.g., 12 or 15), which may need to be included in the package names

3. Initialize PostgreSQL

    ```
    sudo postgresql-setup initdb
    ```

4. Start the PostgreSQL service

    ```
    sudo systemctl start postgresql.service
    ```

5. Have the PostgreSQL service start at boot

```
sudo systemctl enable postgresql.service
```

3.1.2 Setup

Follow the steps below to setup a new PostgreSQL user and password.

1. Change the passwd for the "postgres" user

   ```
   sudo passwd postgres
   ```

2. Change the user to "postgres" user. Remain logged in as "postgres" user

   ```
   su – postgres
   ```

3. Enter the PostgreSQL command line shell. In this instance "postgres" refers to the default database name, not the user name

   ```
   psql postgres
   ```

4. Locate the authentication file. Note down the file path and name

   ```
   SHOW hba_file;
   ```

5. Exit `psql`

   ```
   \q
   ```

6. Create a new PostgreSQL user by executing

   ```
   createuser --interactive --pwprompt
   ```

   a. Enter the new user name, username

   b. Enter the desired password (or just press the Enter key for no password), then answer "*Y*" to the remaining questions

7. Enter the authentication information for the new user. This allows the user to log into the PostgreSQL server

   a. Use an editor (vi, gedit, etc.) to open the authentication file using the path and name from step 4

   b. Search the file for the line starting with

   ```
   # Type    DATABASE   USER       ADDRESS        METHOD
   ```

   c. Alter the subsequent uncommented lines (ones with no "#") for the following lines for "local", "IPv4", and "IPv6" connections so that the user's password is required to access the PostgreSQL server

   ```
   local   all        all                       md5
   host    all        all        127.0.0.1/32   md5
   host    all        all        ::1/128        md5
   ```

   If "trust" is used in place of "md5" for the authentication method then the users' passwords do not have to be entered when logging into the PostgreSQL server. Individual entries can be added for a new user to specify their authentication method – this overrides the method set for user "all" above as long as these entries precede the more contraining ones.

   ```
   local   all        newuser                   authentication
   ```

```
host    all        newuser   127.0.0.1/32  authentication
host    all        newuser   ::1/128       authentication
```

   d. Save and close the file

8. Type Ctrl-D to logout as “postgres” user

9. Restart the postgresql service by executing

   ```
   sudo systemctl restart postgresql.service
   ```

10. If the user is also a Linux login name then add the new user to the “postgres” group by executing

    ```
    sudo usermod -a -G postgres username
    ```

    PostgreSQL should now be fully configured for CCDD usage

11. Verify that the new user has been properly set up

    a. Log out of or restart your computer

    b. Log back in and create a test database for the new user by executing

    ```
    createdb ccdd_test -O <new username>
    ```

    c. Next, log into the psql shell by executing

    ```
    psql ccdd_test
    ```

    d. Verify that the new user is configured with the attributes *Superuser*, *Create role*, and *Create DB*

    ```
    \du
    ```

3.2 CCDD

This section describes the installation and setup of the CCDD tool. It assumes that **`git`** is installed.

3.2.1 Installation

1. Open a terminal and change to the git repository directory

2. Clone a copy from the NASA Github repository

   ```
   git clone https://github.com/nasa/ccdd.git
   ```

3. Go into the ccdd directory and checkout the “CCDD-2” branch

   ```
   cd ccdd

   git checkout CCDD-2
   ```

4. Launch CCDD

   ```
   java -jar CCDD.jar
   ```

5. Enter the user name and password created in section 3.1.2 in the login dialog

3.2.2 Setup

Set up the CCDD path preferences. This sets the storage path for CCDD’s event logs as well as for various items so that when a dialog is opened the file path points initially to the selected locations, reducing the need for folder navigation

1. Select **File** | **Preferences**, then select the **Path** tab in the **Preferences** dialog that appears
2. Select a path for each item using the respective **Select…** button. Folders may need to be created – this can be done within the selection dialog
3. Environment variables may be used in the path names (e.g,. `$HOME` for the user's home directory). This makes switching between users easier. An environment variable can be overridden in the **Other** tab

3.3 Py4J

This section describes installation and use of the Py4J Python to Java gateway server. Py4J allows Python scripts to access the CCDD script data access methods for any version of Python (Jython, the other supported scripting language for Python, is only valid with Python 2.7.x).

3.3.1 Installation

Detailed installation instructiona can be found on the Py4J web site at https://www.py4j.org/install.html. Following is a synosis of the instructions for Linux using an official release. Python 2.7.x or Python 3.x is assumed to be installed.

1. Download the source distribution .tar file from https://pypi.org/project/py4j/#files
2. Untar the file and change to the directory created
3. Install Py4J

   `python setup.py install` *if Python 2.7.x is installed*

   `python3 setup.py install` *if Python3.x is installed*

4. The set up script output indicates the location of the Py4J package. It should look similar to this:

   ```
   Installed /usr/local/lib/python3.6/site-packages/py4j-0.10.9.7-
   py3.6.egg
   ```

   Edit `.bash_profile` and update the `PYTHONPATH` environment variable with the package location. Using the example above:

   ```
   PYTHONPATH=$PYTHONPATH:/usr/local/lib/python3.6/site-packages/py4j-
   0.10.9.7-py3.6.egg
   ```

3.3.2 Running CCDD

CCDD needs access to the Py4J library file (as with any other scripting language). This library is a .jar file that is included in the .tar file downloaded and untarred in paragraph 3.3.1. The .jar file can be found in the py4j-java folder. The CCDD Users Guide, section 4.1 provides details on including the library in the class path.

3.3.3 Setup

Set up which version of Python CCDD uses when executing a Python script.

1. In CCDD select **File** | **Preferences**, then select the **Other** tab in the **Preferences** dialog that appears

2. Enter the **Python command name**

   `python` *if using Python 2.7.x*

   `python3` *if using Python 3.x*

   An environment variable can be entered for the Python command name (e.g., $PYTHON). This enviroment variable can then be set in the user's `.bashrc` or `.bash_profile` file:

   ```
   export PYTHON=python3
   ```

Appendix A. Acronyms

CCDD	CFS Command & Data Dictionary
JRE	Java Runtime Environment
SQL	Structured Query Language