

|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-#####   | Baseline      |
|   | Date: January 2017   | Page 1 of 166 |

# Core Flight System Command and Data Dictionary Utility User's Guide

Engineering Directorate  
Software, Robotics, and Simulation Division

Baseline  
January 2017



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



|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-#####   | Baseline      |
|   | Date: January 2017   | Page 2 of 166 |

## Contents

|             |                                    |     |
|-------------|------------------------------------|-----|
| 1.0         | Description .....                  | 5   |
| 2.0         | Requirements .....                 | 6   |
| 3.0         | Installation .....                 | 6   |
| 4.0         | Operation .....                    | 7   |
| 4.1         | Getting Started .....              | 7   |
| 4.2         | Program Preferences.....           | 12  |
| 4.3         | Mouse and Keyboard Navigation..... | 13  |
| 4.4         | PostgreSQL Setup.....              | 14  |
| 4.5         | Event Log .....                    | 15  |
| 4.6         | Data Tables .....                  | 16  |
| 4.6.1       | Table types.....                   | 16  |
| 4.6.2       | Table groups.....                  | 20  |
| 4.6.3       | Table tree.....                    | 20  |
| 4.6.4       | Data types .....                   | 22  |
| 4.6.5       | Enumerations.....                  | 24  |
| 4.6.6       | Macros .....                       | 25  |
| 4.7         | Data Fields .....                  | 26  |
| 4.7.1       | Data field editor .....            | 26  |
| 4.8         | Input Types .....                  | 29  |
| 4.9         | Data Streams .....                 | 30  |
| 4.10        | Command Menu .....                 | 31  |
| 4.10.1      | File menu .....                    | 31  |
| 4.10.2      | Project menu.....                  | 40  |
| 4.10.3      | Data menu .....                    | 49  |
| 4.10.4      | Scheduling.....                    | 80  |
| 4.10.5      | Script menu.....                   | 92  |
| 4.10.6      | Help menu.....                     | 99  |
| 4.11        | Scripts.....                       | 100 |
| 4.11.1      | JavaScript .....                   | 101 |
| 4.11.2      | Python.....                        | 102 |
| 4.11.3      | Ruby.....                          | 102 |
| 4.11.4      | Groovy .....                       | 103 |
| 4.11.5      | Command line execution.....        | 103 |
| 4.11.6      | Data access methods.....           | 104 |
| Appendix A. | Acronyms .....                     | 129 |
| Appendix B. | Definitions.....                   | 130 |
| Appendix C. | Error & Warning Messages .....     | 131 |
| Appendix D. | Known Issues.....                  | 155 |
| Appendix E. | Program Notes .....                | 157 |
| a.          | CCDD class files .....             | 157 |
| b.          | PostgreSQL tables .....            | 160 |

|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-#####   | Baseline      |
|   | Date: January 2017   | Page 3 of 166 |

## Figures

|            |   |    |
|------------|---|----|
| Figure 1.  | CCDD inputs and outputs .....   | 5  |
| Figure 2.  | CCDD main window .....  | 12 |
| Figure 3.  | Example array display.....  | 18 |
| Figure 4.  | Example dialog .....  | 18 |
| Figure 5.  | Table tree.....   | 20 |
| Figure 6.  | Table tree expansion .....  | 21 |
| Figure 7.  | Example of macro name display and pop-up dialog in a data table ..... | 25 |
| Figure 8.  | Data field editor .....   | 27 |
| Figure 9.  | Select User dialog (no server connection).....                        | 31 |
| Figure 10. | Select User dialog (server connected).....                            | 32 |
| Figure 11. | Database server dialog .....  | 32 |
| Figure 12. | Search event log dialog.....  | 33 |
| Figure 13. | Web Server dialog .....   | 39 |
| Figure 14. | Application Appearance dialog .....                                   | 39 |
| Figure 15. | Example look and feel differences .....                               | 40 |
| Figure 16. | Select Project dialog .....   | 41 |
| Figure 17. | Create Project dialog .....   | 42 |
| Figure 18. | Rename Project dialog.....  | 43 |
| Figure 19. | Copy Project dialog.....  | 44 |
| Figure 20. | Delete Project dialog .....   | 45 |
| Figure 21. | Backup Project dialog .....   | 45 |
| Figure 22. | Unlock Project(s) dialog .....  | 47 |
| Figure 23. | Example Perform Updates dialog.....                                   | 49 |
| Figure 24. | New Table dialog .....  | 50 |
| Figure 25. | Select Table dialog.....  | 51 |
| Figure 26. | Example table editor .....  | 51 |
| Figure 27. | Rename Table dialog .....   | 57 |
| Figure 28. | Copy Table dialog .....   | 58 |
| Figure 29. | Delete Table dialog.....  | 59 |
| Figure 30. | Import table(s) dialog .....  | 60 |
| Figure 31. | CSV export dialog .....   | 61 |
| Figure 32. | EDS export dialog .....   | 62 |
| Figure 33. | XTCE export dialog.....   | 63 |
| Figure 34. | Manage Groups dialog.....   | 64 |
| Figure 35. | New Group dialog.....   | 65 |
| Figure 36. | Table type editor .....   | 66 |
| Figure 37. | Data Type Editor dialog .....   | 72 |
| Figure 38. | Example pointer to a structure data type.....                         | 73 |
| Figure 39. | Structure name pop-up .....   | 73 |
| Figure 40. | Macro Editor dialog .....   | 74 |
| Figure 41. | Assign Table Message IDs dialog .....                                 | 76 |
| Figure 42. | Example Select Data Field(s) dialog.....                              | 77 |
| Figure 43. | Example Show/Edit Data Fields dialog .....                            | 79 |
| Figure 44. | Search tables dialog.....   | 80 |

|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-#####   | Baseline      |
|   | Date: January 2017   | Page 4 of 166 |

|            |  |     |
|------------|--|-----|
| Figure 45. | Manage Links dialog .....                          | 81  |
| Figure 46. | New Link dialog .....                              | 82  |
| Figure 47. | Copy Link(s) dialog.....                           | 83  |
| Figure 48. | Example link copy failure dialog.....              | 83  |
| Figure 49. | Telemetry Scheduler dialog .....                   | 84  |
| Figure 50. | Assign telemetry message names and IDs dialog..... | 86  |
| Figure 51. | Application Scheduler dialog .....                 | 88  |
| Figure 52. | Rate Parameters dialog .....                       | 90  |
| Figure 53. | Application Parameters dialog.....                 | 91  |
| Figure 54. | Manage Script Associations dialog.....             | 93  |
| Figure 55. | Execute Script(s) dialog.....                      | 95  |
| Figure 56. | Script selection dialog.....                       | 96  |
| Figure 57. | Retrieve Script(s) dialog.....                     | 97  |
| Figure 58. | Delete Script(s) dialog.....                       | 98  |
| Figure 59. | Script search dialog .....                         | 99  |
| Figure 60. | About dialog.....                                  | 100 |

## Tables

|          |  |     |
|----------|--|-----|
| Table 1. | Command line arguments .....               | 10  |
| Table 2. | Structure column and input data types..... | 17  |
| Table 3. | Command column and input data types .....  | 19  |
| Table 4. | Variable tree icons .....                  | 22  |
| Table 5. | Default primitive data types .....         | 23  |
| Table 6. | Web data access commands.....              | 38  |
| Table 7. | Script Data Access Methods .....           | 128 |

|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-#####   | Baseline      |
|   | Date: January 2017   | Page 5 of 166 |

## 1.0 Description

The Core Flight System (CFS) Command and Data Dictionary (CDD) utility, or CCDD, is a software tool for managing the data structures for the 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.

The CCDD application uses tables, similar to a spreadsheet, to display and allow manipulation of telemetry data structures, command information, and other data pertinent to a CFS project. The data is stored in a PostgreSQL database for manipulation and data security. The database server can be run locally or centralized on a remote host for easier access by multiple users. Data can be imported into or exported from the application from files in comma-separated values (CSV), extensible markup language (XML) telemetric and command exchange (XTCE) XML, or electronic data sheet (EDS) XML formats. The CCDD tables also allow simple cut and paste operations from the host operating system's clipboard. To make use of the project's data, CCDD can interact with Java Virtual Machine (JVM)-based scripting languages via a set of supplied data access methods. Using scripts, the user can translate the data stored in the CCDD's database into output files. Example scripts for creating common CFS output files are provided in four of these scripting languages. AN embedded web server can be activated, allowing web-based application access to the data. Figure 1 shows the basic relation between CCDD and external sources.

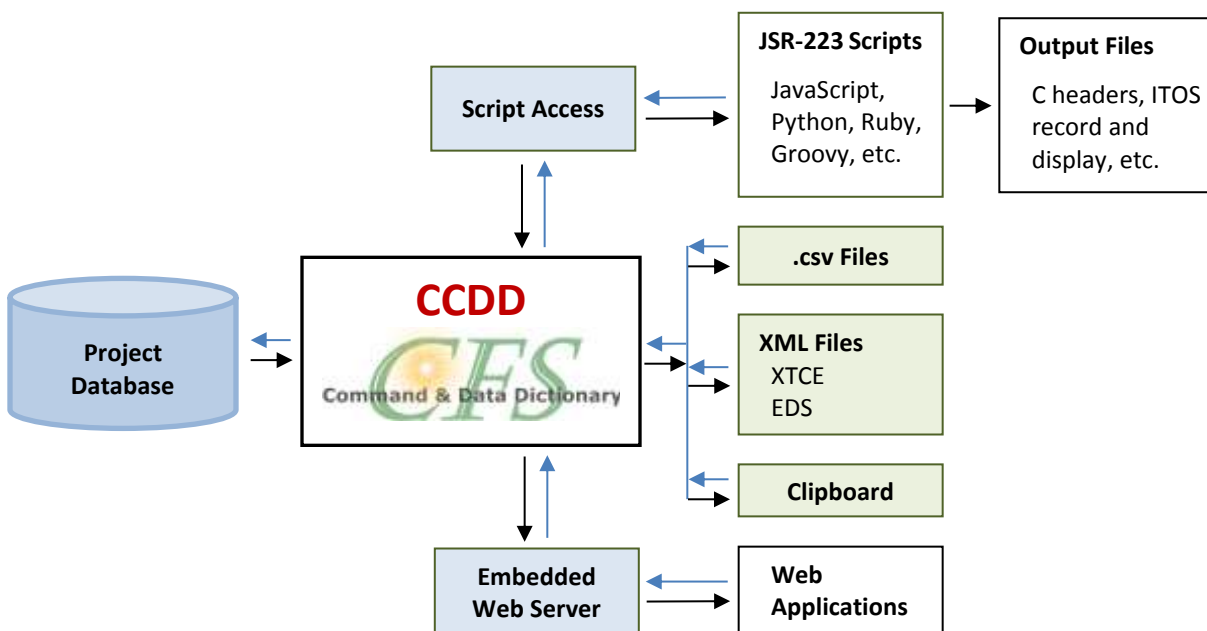


Figure 1. CCDD inputs and outputs

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

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

|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-#####   | Baseline      |
|   | Date: January 2017   | Page 6 of 166 |

## 2.0 Requirements

CCDD is written based on the following Java and PostgreSQL versions:

- JavaSE 1.7
  - Developed in Linux using JavaSE 1.7 and tested on Microsoft Windows and Apple OS X using JavaSE 1.8
- PostgreSQL 8.4
- Java Database Connectivity (JDBC) driver 8.4-703 4.0

CCDD uses the following embedded web server version:

- Jetty 9.2.18.v20160721

Scripting language testing was performed using the following languages and versions:

- JavaScript (ECMAScript) 1.8 (Mozilla Rhino 1.7 release 3)
- Python 2.7 (Jython 2.7 ), PyDev 4.2.0
- Ruby 9.0.1.0 (JRuby 9.0.1.0)
- Groovy 2.3.7 (Groovy Scripting Engine 2.0)

Compatibility with other versions, in particular earlier ones, is not guaranteed.

## 3.0 Installation

To install CCDD copy the Java archive (jar) file CCDD.jar to a folder. The application requires read/write access to the folder it resides in so that event log file(s) can be created (see paragraph 4.5 for further information on event logs). **Java, PostgreSQL, and the JDBC driver must be installed before the application can be used.**

- To install Java, go to [www.java.com](http://www.java.com) and locate the installation instructions appropriate for the operating system on which the application is to be run.
- The PostgreSQL relational database management system is available for download from [www.postgresql.org](http://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](http://www.postgresql.org).
- The PostgreSQL JDBC driver is located at [jdbc.postgresql.org](http://jdbc.postgresql.org).

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, and only the scripting language(s) intended for use with the application need to be installed. The application was tested with four of the available languages: JavaScript, Python, Ruby, and Groovy. Details are provided in this and subsequent sections of this guide on the use of these four scripting languages; installation and use of other languages should be similar. The CCDD package provides versions of some common scripts in each of the four languages.

|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-####  | Baseline      |
|   | Date: January 2017   | Page 7 of 166 |

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](http://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](http://developer.mozilla.org).
- *Python*™ scripting is implemented using *Jython*, the Python implementation for Java. Jython can be downloaded from [www.jython.org](http://www.jython.org). *PyDev* is also required and can be found at [www.pydev.org](http://www.pydev.org).
- *Ruby* scripting is implemented using *JRuby*, which implements Ruby in Java. JRuby is available for download from [jruby.org](http://jruby.org).
- *Groovy* can be downloaded from [www.groovy-lang.org](http://www.groovy-lang.org).

## 4.0 Operation

### 4.1 Getting Started

To run the application open a command prompt window and type:

```
java -classpath class_paths CCDD.CcddMain [args...]
```

where *class\_paths* includes the path and .jar file name for the CCDD application, the JDBC driver, and the installed script languages, separated by colons with no intervening spaces (in the file names below <version> is the specific version number of the installed file that is included as part of the file's name):

```
<CCDD path>/CCDD.jar
<JDBC path>/postgresql-<version>.jar
<JRuby path>/lib/jruby.jar
```

**Note:** the *JRuby* reference must precede the other scripting language file references!

```
<Jython path>jython.jar:<PyDev path>/plugins/org.python.pydev.jython_<version>/jython.jar
<Groovy path>/lib/groovy-<version>.jar:<Groovy path>/lib/groovy-jsr223-
<version>.jar
```

and *args* are optional command line arguments in the form:

```
[[<- or />]command value [...]]
```

Each argument consists of a command, optionally preceded by either a '-' or '/', followed by a space, then the command value. The available commands and acceptable values are described in Table 1. The commands can be entered in any order. If a command is entered more than once the last instance's value is used. The commands are not case-sensitive, so "-user" is the same as "-USER", "-user", etc.

| Command | Description  | Value  | Default Value                   |
|---------|--|--|---------------------------------|
| project | Selects the project database to which to initially connect | Project database name. The project's database name is case sensitive | Previous session's project name |

|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-#####   | Baseline      |
|   | Date: January 2017   | Page 8 of 166 |

| Command  | Description   | Value  | Default Value  |
|----------|---|--|--|
| backup   | Sets the file path and name to which to automatically backup the project database once is successfully connected. The extension “.dbu” is automatically appended to the file name if not already present. Only applies to the first successful connection | File path and name of the project backup file  | <i>None</i>  |
| user     | Sets the user name to use when connecting to the PostgreSQL server  | User name for PostgreSQL. The user name is case sensitive  | <i>None or previous session's user name</i>            |
| password | Sets the user's PostgreSQL password   | Password for user name for PostgreSQL. The password is case sensitive  | <i>None</i>  |
| host     | Sets the name of the PostgreSQL server's host   | PostgreSQL server host name. The host name is case sensitive   | <i>localhost or previous session's PostgreSQL host</i> |
| port     | Sets the port of the PostgreSQL server's host   | PostgreSQL server port. The server port must be blank or a positive integer  | <i>Blank or previous session's PostgreSQL port</i>     |
| events   | Selects whether or not to display all event log messages  | “true” to display all event log messages in the main application window; “false” to hide all event log messages. The value text must be lower case         | true   |
| command  | Selects whether or not to display event log command messages  | “true” to display event log command messages in the main application window; “false” to hide event log command messages. The value text must be lower case | true   |
| success  | Selects whether or not to display event log success messages  | “true” to display event log success messages in the main application window; “false” to hide event log success messages. The value text must be lower case | true   |



|   |  |               |
|---|--|---------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |               |
|   | Doc. No. JSC-#####   | Baseline      |
|   | Date: January 2017   | Page 9 of 166 |

| Command   | Description   | Value   | Default Value  |
|-----------|---|---|--|
| fail      | Selects whether or not to display event log fail messages   | "true" to display event log fail messages in the main application window; "false" to hide event log fail messages. The value text must be lower case  | true   |
| status    | Selects whether or not to display event log status messages   | "true" to display event log status messages in the main application window; "false" to hide event log status messages. The value text must be lower case  | true   |
| server    | Selects whether or not to display event log web server messages   | "true" to display event log web server messages in the main application window; "false" to hide event log web server messages. The value text must be lower case  | true   |
| laf       | Sets the application look & feel  | "Look and feel" name (e.g., "Nimbus", "Windows", etc.). The names are case sensitive  | Metal or previous session's L&F                        |
| mainSize  | Sets the main application window's size   | Main application window size in pixels. The parameter format must be in the form <i>widthxheight</i> where <i>width</i> and <i>height</i> are positive integer values. A width or height less than the minimum allowed (750 for width, 400 for height) is replaced by the minimum value | 750x400  |
| validate  | Selects whether or not the telemetry and application schedulers' data is validated each time the scheduler is opened. See paragraph ... for more detail | "true" to validate the telemetry and application schedulers' data when the scheduler is opened  | false or previous session's automatic validation state |
| webserver | Enables the embedded web server. See paragraph 4.10.1.5 for more detail   | "nogui" to start the application and enable the web server without displaying the user interface; "gui" to start the application, enable the web server, and display the user interface   | nogui or gui   |
| webport   | Enables the embedded web server. See paragraph 4.10.1.5 for more detail   | Valid port number for the web server to listen to for queries   | 7070 or previous session's web server port             |

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 10 of 166 |

| Command | Description  | Value   | Default Value |
|---------|--|---|---------------|
| execute | Runs the supplied script(s) using the supplied table(s). The application graphical user interface (GUI) is not displayed; exits upon completion of the script(s). See paragraph 4.11.5 for more detail | Script name. Script file paths are required if the script is in a folder other than the one from which the application is executed. If the script requires one or more tables to be specified then the table name(s) are placed after the script name and a colon. If multiple tables are specified the table names must be separated by a plus (+) character. When multiple scripts are run each definition, as described above, is separated by a comma | None          |

Table 1. Command line arguments

The following is an example of starting the application in Linux. In this example the CCDD application is installed in the current folder and the supporting software is installed in the /opt folder. The project initially opened is "myProject" by user "userName":

```
java -classpath ./CCDD.jar:/opt/JDBC/postgresql-9.3-1102.jdbc4.jar:/opt/jruby-9.0.1.0/lib/jruby.jar:/opt/jython2.7.0/jython.jar:/opt/PyDev/plugins/org.python.pydev.jython_4.3.0.201508182223/jython.jar:/opt/groovy-2.4.4/lib/groovy-2.4.4.jar:/opt/groovy-2.4.4/lib/groovy-jsr223-2.4.4.jar CCDD.CcddMain -project myProject -user userName
```

To make execution easier an alias can be created. Using the example above the Linux alias command is as follows:

```
alias CCDD='java -classpath ./CCDD.jar:/opt/JDBC/postgresql-9.3-1102.jdbc4.jar:/opt/jruby-9.0.1.0/lib/jruby.jar:/opt/jython2.7.0/jython.jar:/opt/PyDev/plugins/org.python.pydev.jython_4.3.0.201508182223/jython.jar:/opt/groovy-2.4.4/lib/groovy-2.4.4.jar:/opt/groovy 2.4.4/lib/groovy jsr223 2.4.4.jar CCDD.CcddMain'
```

For Microsoft Windows, the doskey command can be used to create an alias (individual class paths must be separated by semi-colons instead of colons):

```
doskey CCDD=java -classpath "class_paths" CCDD.CcddMain $*
```

Having created an alias, the application can then be started by simply typing:

```
CCDD [args...]
```

An invalid command or command parameter results in program termination. An invalid parameter displays an error message at the command prompt. An invalid command or a valid command without an associated parameter produces the following output at the command prompt:

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-####  | Baseline       |
|   | Date: January 2017   | Page 11 of 166 |

usage:

```
java -classpath <class_paths> CCDD.CcddMain [[<- or />]<command> <value> [...]]
```

Command line arguments:

| Description                  | Command   | Value   |
|------------------------------|-----------|---|
| -----                        | -----     | -----   |
| Select CCDD project          | project   | <project name>                                      |
| Backup project on connecting | backup    | <backup file name>                                  |
| Set user name                | user      | <user name>   |
| Set user password            | password  | <user password>                                     |
| Set PostgreSQL server host   | host      | <host name>   |
| Set PostgreSQL server port   | port      | <port number>                                       |
| Show events                  | events    | <true or false>                                     |
| Show command events          | command   | <true or false>                                     |
| Show success events          | success   | <true or false>                                     |
| Show fail events             | fail      | <true or false>                                     |
| Show status events           | status    | <true or false>                                     |
| Show web server events       | server    | <true or false>                                     |
| Load look & feel             | laf       | <look & feel>                                       |
| Set main window size         | mainSize  | <widthxheight>                                      |
| Validate messages            | validate  | <true or false>                                     |
| Enable web server            | webserver | <nogui or gui>                                      |
| Set web server port          | webport   | <port number>                                       |
| Execute script(s)            | execute   | <script_name[:table[+table2[+...[+tableN]]]][,...]> |

Once the application is executed the CCDD main window appears as shown in Figure 2. If password authentication is enforced (see paragraph 4.4) and a password is not supplied on the command line then the Select User dialog appears (see paragraph 4.10.1.1), allowing the user and password to be entered. The graphical user interface (GUI) L&F can be selected by the user from a list of ones installed on the operating system. If the L&F is changed then the application window and dialogs may differ in appearance (but not function) from those shown in the figures below. See paragraph 4.10.1.5 on how to alter the L&F.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 12 of 166 |

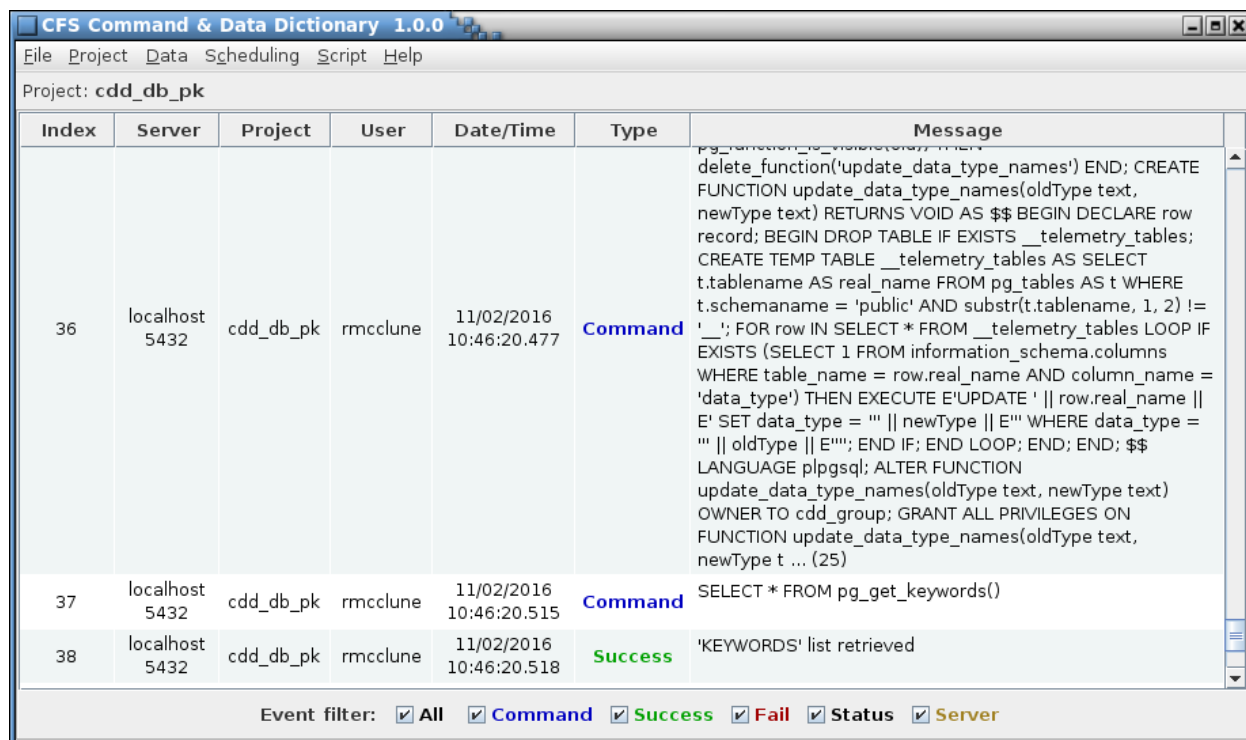


Figure 2. CCDD main window

The main window header contains the program name and version number. The main window is divided into a menu bar along the top and a session event log display area underneath. See paragraph 4.5 for further information on the event log. The window can be resized as desired. Each menu contains one or more menu items or sub-menus. A menu items that is grayed-out indicates that the affected item is not available at that time; for example, if no project database is open then the table commands are not available. A description of each of the menu items is provided in section 4.9.

## 4.2 Program Preferences

The programs preferences are stored in a location dependent on the operating system and are updated as needed by the CCDD application. For example, the Windows operating system stores the preferences in the system registry under the key name:

HKEY\_CURRENT\_USER\Software\JavaSoft\Prefs\C/C/D/D

In Linux the preferences are stored in the file:

/<user home directory>/ .java/.userPrefs/CCDD/prefs.xml

The user should have no need to manually edit these preferences; however, a description of the preference keys and associated values is provided below for reference purposes.

|                             |   |
|-----------------------------|---|
| <b>PostgreSQLServerHost</b> | The name of the PostgreSQL server host that was connected to most recently          |
| <b>PostgreSQLServerPort</b> | The PostgreSQL server port number of the server that was connected to most recently |
| <b>Database</b>             | The name of the project database that was connected to most recently                |

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 13 of 166 |

|                               |  |
|-------------------------------|--|
| <b>User</b>                   | The name of the latest user to attempt a server connection   |
| <b>LogFilePath</b>            | The full path name for the location where an event log was most recently opened for reading. This is not the path of the current session log |
| <b>LastSavedDataFile</b>      | The full path and file name of the latest file imported or exported  |
| <b>LastSavedDataPath</b>      | The full path of the latest file imported or exported  |
| <b>LastDatabaseBackupFile</b> | The full path and file name of the latest database backup file   |
| <b>LastDatabaseExportFile</b> | The full path and file name of the latest database export file   |
| <b>LastScriptFile</b>         | The full path and file name of the latest script file selected via the Script Manager dialog   |
| <b>LastScriptPath</b>         | The full path of the latest script file location selected via the Script storage or retrieval dialog   |
| <b>LookAndFeel</b>            | The name of the selected "look and feel"   |
| <b>AutoValidate</b>           | The state of the automatic scheduler data validation flag  |
| <b>WebServerPort</b>          | The web server port number of the server that was connected to most recently   |

### 4.3 Mouse and Keyboard Navigation

The application's menus, dialogs, and GUI components can be manipulated using the mouse pointer, mouse buttons, and mouse wheel, as well as with the keyboard. Keyboard mnemonics are provided for the menu items and dialog buttons. These are accessed by pressing the Alt key in addition to another key; i.e., Alt+key, where *key* is the underlined character in the menu or button text (the key case is ignored). For example, pressing Alt+F or Alt+f in the main application window opens the **F**ile menu. The Tab and arrow keys can be used to navigate between the components in a dialog or window, and the pressing the Enter or space key actuates a control (e.g., a button or check box).

When a dialog containing a table is initially displayed it has no row selected. A row can be selected by positioning the mouse pointer over a cell in the row and pressing the left mouse button, or by using the keyboard. To select an initial row with the keyboard press the Tab key then the Enter or Space key when the table has the keyboard focus (which it does initially); this selects the table's topmost visible row and sets the focus to that row's leftmost column. The up and down arrow keys can then be used to change the selected row and the left and right arrows can change the selected column. The selected cell is highlighted. Multiple cell selection behavior is dependent on the particular table, but in general behaves as follows. Multiple, contiguous cells can be selected using a combination of the mouse/keyboard and the Shift key. Highlight the starting cell, then either (a) continue to press the left mouse button and drag the pointer, (b) hold the Shift key and left-click the mouse on another row (the two rows, plus any in between, are highlighted), or (c) hold the Shift key and press the arrow key to highlight as many cells as desired. Individual cells can be selected/deselected by pressing the Ctrl key and selecting the cell with the mouse. The entire table may be selected by pressing Ctrl-A. For row operations (e.g., Move up or Delete row) the row(s) indicated by the highlighted cell(s) are affected. Similarly, for column operations (e.g., Move left) the column(s) indicated by the highlighted cell(s) are affected. Once one or more cells are selected the highlighted data can be copied by pressing Ctrl-C. To

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 14 of 166 |

paste the data into another application (e.g., spreadsheet or text document) or another table use the Ctrl-V key sequence.

Navigation within a table can be accomplished via mouse or keyboard. Note that some of these keys perform different functions if a cell is actively being edited. The Insert key inserts a row at the current selection point and the Delete key erases the contents of the currently selected cell(s) (see above paragraph concerning cell selection). Pressing the Ctrl-Delete deletes the currently selected rows. The Home and End keys change the cell selection to the first or last column, respectively, of the currently selected row. The Page Up and Page Down keys scroll the table up or down one page, respectively, (unless the entire table is already visible) changing the cell selection to the currently selected column, with the row one page up or down from its previous position.

Table data entry is initiated by double clicking the left mouse button while the mouse pointer is over the cell to be edited. The Enter or Space keys may also be used to initiate editing on the currently selected cell (the Space key initiates editing as well as inserts a space into the cell at the end of any existing text). Pressing the Enter key while editing a cell stores the text in the cell and initiates editing in the next cell, moving left to right until the last column is reached, at which point editing moves to the first column in the next row below unless the end of the table is reached. Protected cells, denoted by a gray background color, are skipped. A cell containing a check box does not allow moving to the next cell via the Enter key; instead, the check box state is toggles with each press of the Enter key.

While cell editing is active the Insert key inserts a space to the right of the text cursor, and the Delete key deletes the character to the right of the text cursor. The Home and End keys move the text cursor to the beginning and end of the cell, respectively.

Pressing the Escape key while editing terminates editing of the cell and removes any changes made to the cell.

Details specific to navigation in certain windows and dialogs are provided in the components' descriptions in later sections.

## 4.4 PostgreSQL Setup

A description of installing and configuring the PostgreSQL software and server is beyond the scope of this document; see the PostgreSQL web site at [www.postgresql.org](http://www.postgresql.org) for this information.

The password authentication configuration for the PostgreSQL server affects the behavior of the CCDD application. Password authentication is controlled via the pg\_hba.conf file. Super user status is required in order to make changes to this file. The location of this file can be determined by executing the "SHOW hba\_file;" command in the PostgreSQL server command line utility **psql**. There are a number of authentication methods described in the documentation on the PostgreSQL web site; e.g. "trust", "password", "md5", etc. The methods can be applied to all users or to individual users based on the connection type (local or remote). When set to "trust" no password is required to log into the server and access a database. The methods "password" and "md5" are similar in that the user must provide a password to log into the server. This is preferred in multi-user scenarios to control who may access the server and databases.

PostgreSQL allows only the owner of a database element (table, sequence, etc.) to make changes to that object. This would be problematic if multiple users require the capability to make updates. The restriction is overcome by means of *roles*. Every user login is a role in the server. Group roles can be

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 15 of 166 |

created to which other roles (e.g., users) are assigned membership; any role belonging to the group inherits the privileges assigned to the group role. The PostgreSQL administrator must create a role for each user (the user's login identity), and one or more group roles that are used as the owner role when a project is created. The administrator must also assign membership in the group role to the appropriate users. Role creation and maintenance is performed outside the CCDD application, and must be completed prior to creating a project database. When a project database is created, one of the group roles is assigned as the owner (see paragraph 4.10.2.3; note that for a single-user project the user's role can be selected as the owner). Since all elements of the database are owned by the selected group, all members of the group have write privileges to these elements. Other roles (users) not in the group are prevented from changing the project elements.

## 4.5 Event Log

The application automatically records all interactions with the PostgreSQL and web servers. The information includes the exact commands issued to the server and the server responses (success, or failure with supporting information). All events are logged to the session's log file, even if the GUI is disabled.

When the GUI is enabled the main application windows displays the current session's event log. Previous sessions' event logs can be reviewed using the **Read log** command; see paragraph 4.10.1.3. The log automatically scrolls to the latest entry when it is logged. Each log entry contains the following information arranged in a tabular format:

|                  |  |
|------------------|--|
| <b>Index</b>     | This is a sequential number assigned to each log entry, beginning with 1 and incrementing by one as transactions occur with the database   |
| <b>Server</b>    | Name or address of the computer hosting the PostgreSQL server  |
| <b>Project</b>   | Name of the open database to which the transaction applies   |
| <b>User</b>      | Name of the user that initiated the transaction  |
| <b>Data/Time</b> | Date (month, day, year) and time stamp (hours. minutes, seconds) when the transaction occurred   |
| <b>Type</b>      | One of five log entry types: <ul style="list-style-type: none"> <li><b>Command</b> Indicates a PostgreSQL command issued to the database</li> <li><b>Success</b> Indicates the database transaction completed successfully</li> <li><b>Fail</b> Indicates the database transaction failed</li> <li><b>Status</b> Indicates the log entry provides the application status</li> <li><b>Server</b> Indicates a web server command</li> </ul>  |
| <b>Message</b>   | The text in this column is dependent on the message <b>Type</b> . For a <b>Command</b> type the text is the PostgreSQL command issued to the database. If a data base transaction succeeds then a <b>Success</b> type message indicates what was accomplished by the database command. A <b>Fail</b> type message provides details on the cause of the transaction failure. Failed transactions are rolled back so that no change is made to the database. The <b>Status</b> type message shows the results for an application operation (for example, the database table consistency check). The message length displayed is limited to 2000 characters in order to prevent bogging |

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 16 of 166 |

down the application. Truncated messages are denoted by a trailing ellipsis (...) followed by the number of truncated characters in parentheses. The full text of the message can be viewed by double clicking the right mouse button while the mouse pointer is over a log entry row – a log entry viewer is opened showing the full message text for that row.

The log columns can be sorted by selecting the column header with the mouse pointer and pressing the left mouse button. The rows are first sorted in ascending order, depending on the selected column's contents. Selecting the column again sorts in descending order, and a third selection restores the rows to their original order.

Beneath the logged entries are entry filter check boxes that can be used to determine which messages are displayed based on the message type(s). If a message type's check box is unchecked then messages of that type are hidden. Checking the box restores the messages. Messages for hidden types are still logged even if not currently displayed. The **Server** check box only appears once the web server is activated. The **All** check box affects the remaining check boxes – unchecking it clears the other check boxes, and checking it selects the others. If none of the check boxes are selected then the log appears empty. Note that for the single log entry viewer the filter check boxes are not displayed.

## 4.6 Data Tables

The CCDD data is stored in the project's database in the form of tables. The tables consist of a two-dimensional array of columns and rows. The columns define the content of the cell in each row, much like the data in a spreadsheet. For example, a table may have a column titled "Description" which indicates that the cells in that column contain descriptive text concerning the parameter defined in each specific row. There is no constraint on the number of tables in the project's database, nor is there a constraint on the table's number of columns and rows.

### 4.6.1 Table types

Two types of tables are immediately available upon creation of a project: *Structure* and *Command*. Structure tables represent C-program data structures containing information on variables. Command tables are designed to contain information pertinent to CFS commands. Other table types may be created by the user to contain data that doesn't fit into the predefined types (see paragraph 4.10.3.6 for information on the table type editor). All tables of a given type share the same column definitions. Data in tables of any type are accessible via the scripts (see paragraph 4.11 for information regarding script access).

Every table that is created is considered a *prototype*. A table's prototype determines the columns and default data for all *instances* of that table. Each prototype table itself constitutes an instance of that table, and in many cases the prototype is the only instance. However, in the case of structure tables, multiple instances can exist – one for every reference to the structure from within another structure. Each of these derive their columns and initial data values from their prototype table.

A prototype table that is not referenced from within another table is considered a *parent* table. A table referenced by another table is a *child* of that table. It's common for structure tables to have a parent-child relationship. It's possible for tables of other types to have such a relationship as well, though less likely.



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 17 of 166 |

#### 4.6.1.1 Data Structure tables

Structure table rows represent C-program variables. The variables can either be of a primitive data type (e.g., int32, uint8, double) or can be a reference to another structure. These child structures can in turn reference other structures, and so on, to any depth required by the user's needs. The only constraint is that no circular references are allowed, wherein a structure references itself somewhere in its hierarchy. Ultimately only references to primitive data types exist as the end point of any path from the parent structure, through its child structures, to a variable.

Certain columns are inherent to structures and must be present for the table to be recognized as a structure. The default names for these columns are Variable Name, Data Type, Array Size, Bit Length, Rate, and Enumeration. The column names can be changed if desired; it's the column's input type that identifies the column (see paragraph 4.10.3.8 for more information on input types). Therefore, a structure table must include columns with the input types shown in Table 2. Other columns, Description and Units, are automatically included for structure tables; these additional columns can be altered, or even deleted. Columns containing other variable information can be added at the user's discretion.

| Default Column Name | Input Type            |
|---------------------|-----------------------|
| Variable Name       | Variable name         |
| Data Type           | Primitive & Structure |
| Array Size          | Array index           |
| Bit Length          | Bit length            |
| Rate                | Rate                  |
| Enumeration         | Enumeration           |

Table 2. Structure column and input data types

Only one variable name, data type, array size, and bit length column is allowed per table type definition. The table can have multiple rate and enumeration columns.

The array size and bit length cell values are mutually exclusive for a variable; only one can be assigned (or neither). If array size is specified then the bit length cell for that variable is grayed out and cannot be selected. Conversely, if a bit length is entered then the array size cell is grayed out.

The bit length and enumeration cells are valid only for integer data types (signed and unsigned, including boolean). If a non-integer data type is selected in the data type column then the bit length and enumeration cells for that row are grayed out and cannot be edited. Conversely, if the bit length or enumeration cell is not empty then the data type for that row only displays integer data types.

If an array size is specified a row is inserted automatically into the table for each array member. Arrays may have one or more dimensions. For multi-dimensional arrays the size of each dimension is specified in the array size column, separated from one another by commas. For example, a three dimensional array, **n**, with dimension sizes of 2, 3, and 4 would have the array size specified as "2, 3, 4". The first array member would be **n[0][0][0]**, the second **n[0][0][1]**, and so on until the last member, **n[1][2][3]**, is reached; this array would have a total of 24 members (= 2 x 3 x 4). When a structure table is open any arrays are initially collapsed; in other words only the *array definition* row is shown. The display of the *array member* rows can be toggled in one of two manners, via the **Expand arrays** command (see

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 18 of 166 |

paragraph 4.10.3.2.3.5) or by positioning the mouse pointer over any cell in the array size column and double-clicking the right mouse button. When expanded, the array members for all arrays are displayed beneath their respective array definition row. The variable name column shows the variable name with the array index (or indices) appended, and the overall array dimension size(s) is displayed in the array size column. See Figure 3 for an example of array size input and array expansion.

|                         |                      |                  |                   |
|-------------------------|----------------------|------------------|-------------------|
|                         | <b>Variable Name</b> | <b>Data Type</b> | <b>Array Size</b> |
| <i>array definition</i> | cmdCtr               | uint16           | 3, 2              |
|                         | cmdCtr[0][0]         | uint16           | 3, 2              |
|                         | cmdCtr[0][1]         | uint16           | 3, 2              |
| <i>array members</i>    | cmdCtr[1][0]         | uint16           | 3, 2              |
|                         | cmdCtr[1][1]         | uint16           | 3, 2              |
|                         | cmdCtr[2][0]         | uint16           | 3, 2              |
|                         | cmdCtr[2][1]         | uint16           | 3, 2              |

Figure 3. Example array display

Note that the variable name, data type, and array size are grayed out and cannot be altered in the array member rows; however, individual values may be assigned to a member for the other columns in the table. To change the array member names or data type make the change to the array definition row; the member rows are automatically changed as well. Changing the array definition row's array size value increases or decreases the member rows as needed, and clearing the array size cell removes all of the member rows for that variable.

The string data type is special instance of the char (character) data type. If no array size is specified then the string variable is simply a single character. The first array dimension supplied determines the string length in characters. A string behaves as other array variables except that column values (e.g., description) may not be assigned to array members of the string other than the first one (i.e., one ending with an array index of zero). Arrays of string variables are allowed, as with other data types.

If a data type cell references a structure then the specific instance of the structure table it represents can be opened by double-clicking the right mouse button while the mouse pointer is positioned over the data type cell. The table is opened in its own tab in the same editor window (see paragraph 4.10.3.2 for more information on the table editor). If this is attempted on a structure reference in a prototype table, and the prototype table is itself referenced in another structure table, then the dialog in Figure 4 is displayed indicating that the prototype of the selected structure, and not a specific instance, was opened (*a\_structure\_table* and *a\_child\_table* in the example are replaced by the prototype and child table names respectively). Once a structure is referenced by another one it can no longer be a top-level structure. Therefore, it can't have its own child tables, only those that are part of the hierarchy of the top-level structure to which the prototype belongs.

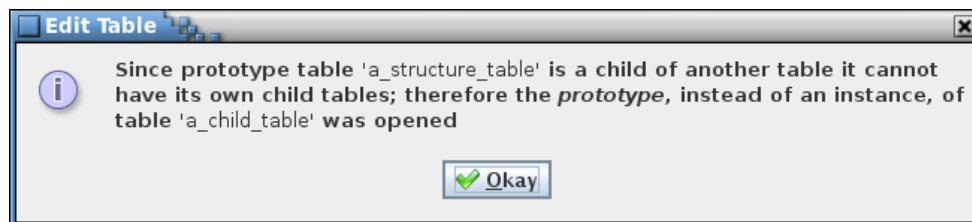


Figure 4. Example dialog

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 19 of 166 |

The user may create other table types that also represent a structure table. If a table contains the default structure input data types then the table is treated as a structure table.

#### 4.6.1.2 Command tables

Command tables contain CFS command information. Certain columns are inherent to command tables and must be present for the table to be recognized as a command table. The default names for these columns are Command ID, Command Name, Command Code, Arg 1 Name, Arg 1 Data Type, Arg 1 Enumeration, Arg 1 Minimum, and Arg 1 Maximum. The column names can be changed if desired; the columns' input type identifies the column (see paragraph 4.10.3.8 for more information on input types). Therefore, a command table must include columns with the input types shown in Table 3. Other columns (Description, Arg 1 Description, and Arg 1 units) are automatically included for command tables; these additional columns can be altered, or even deleted. Columns containing other command information, such as those to describe more command arguments (name, data type, enumeration, minimum, and maximum) can be added at the user's discretion.

| Default Column Name | Input Type    |
|---------------------|---------------|
| Command Name        | Command name  |
| Command Code        | Command code  |
| Arg 1 Name          | Argument name |
| Arg 1 Data Type     | Primitive     |
| Arg 1 Enumeration   | Enumeration   |
| Arg 1 Minimum       | Minimum       |
| Arg 1 Maximum       | Maximum       |

Table 3. Command column and input data types

Only one command ID, command name, and command code column is allowed per table type definition. The table can have multiple argument columns (argument name, data type, enumeration, minimum, and maximum).

Command argument name, data type, enumeration, minimum, and maximum columns are linked. The first data type column in the Command table type definition (i.e., a column designated as having a Primitive input type) is automatically associated with the first enumeration, argument name, minimum value, and maximum value columns (these columns are identified by their respective input types). The second data type column is associated with the second enumeration, etc. columns, and so on. It is suggested that any extra command argument sets are named in a manner similar to the default argument 1 names (e.g., Arg 2 Name, Arg 2 Data Type, etc.) so that the association is apparent to the user.

The command argument enumeration cells are valid only for integer data types (signed and unsigned). If a non-integer data type is selected in the argument's data type column then the argument's associated enumeration cell in the same row is grayed out and cannot be edited. Conversely, if the argument's enumeration cell is not empty then the argument's associated data type for that row only displays integer data types.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 20 of 166 |

## 4.6.2 Table groups

Data tables can be assigned to user-defined groups (see paragraph 4.10.3.8 for details on assigning tables to a group). These groups are a method of relating tables to each other. For example, all of the tables for a specific CFS application or subsystem can be assigned to a group. The groups are used in filtering the table tree (see paragraph 4.6.3 concerning filtering). A table can be assigned to more than one group, or to none. The application scheduler uses the groups designated as CFS applications when producing the scheduler table (see paragraph 4.10.3.8 for details on designating a group as an application). Groups can also be assigned data fields. If a group is specified as representing a CFS application a number of data fields are automatically assigned (these can be edited, removed, or additional fields added as desired). See paragraph 4.7 for more details on data fields.

## 4.6.3 Table tree

The table tree displays the data tables using a tree representation. Depending on the operation (e.g., Edit, Rename, etc.) there are one or two top levels in the tree. The first, labeled *Prototypes*, is an alphabetical arrangement of the prototype tables. Since it displays prototypes only it is a single level in depth (not including any filtering; see below). The other level that may be displayed, *Parents & Children*, shows the parent tables and, if applicable, their children as sub-levels, and the children of those tables as further sub-levels, etc. See Figure 5 and Figure 6.

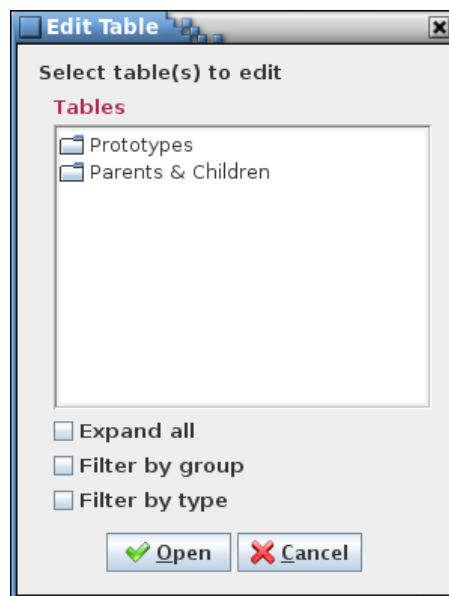


Figure 5. Table tree

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 21 of 166 |

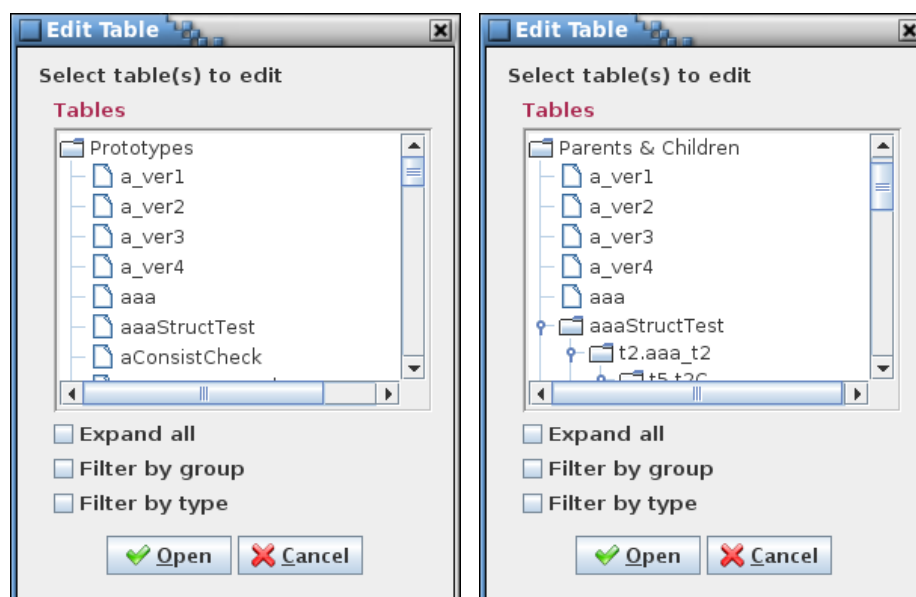


Figure 6. Table tree expansion

Selecting the symbol beside a branch in the tree causes that branch to expand (if collapsed) or to collapse (if expanded). Selection can be made with the mouse pointer, or by using the tab key and up/down arrows to highlight the branch's name, then pressing the right arrow to expand or left arrow to collapse the branch. Positioning the mouse pointer over a branch name and double left-clicking toggles between expanded and collapsed view for that branch. Selecting one or more branches and pressing CTRL-E causes the selected branches and all of their child branches to expand (if collapsed) or collapse (if expanded). The first branch selected determines if any other selected branches are expanded or collapsed. Hovering the mouse over an item in the tree displays a pop-up tool tip showing the description of the item (if it has one).

Below the tree are one or more check boxes. The *Expand all* checkbox is available for every tree; selecting this check box causes all of the tree branches to be displayed. Clearing the check box collapses all of the branches down to the initial level.

The remaining check boxes are used to filter the tree contents. There are two filter methods, *by groups* and *by types*. Depending on the operation one, both, or neither of these check boxes may be available.

If the *Filter by group* check box is selected then sub-branches are inserted at the level below the Prototypes and Parent & Children branches. These sub-branches are the groups defined by the user (see paragraphs 4.6.2 and 4.10.3.9.2). Tables belonging to the group are displayed as sub-branches of the group branch. If a table doesn't not belong to any group then it does not appear in the table tree while this filter is applied. Deselecting the check box removes the group branches.

If the *Filter by type* check box is selected then sub-branches are inserted at the level below the Prototypes and Parent & Children branches. These sub-branches are the table types: structure, command, and any others defined by the user (see paragraphs 4.1 and 4.10.3.6). Tables of a given type are displayed as sub-branches of the table type branch. In other words, all of the Structure tables appear under a Structure branch, all Command tables under a Command branch, and so on for each defined table type. Deselecting the check box removes the type branches.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 22 of 166 |

Both the group and type filters may be applied simultaneously. The branches are first divided by group. Each group is then sub-divided by table type.

#### 4.6.3.1 Variable tree

Another form of the table tree is the variable tree. The variable tree displays only the project's structure tables. These are displayed the same manner as in the table tree, except that the variables belonging to the structure tables are also shown. Variable trees are used where selection of variables is required; e.g., in the links manager (paragraph 4.10.4.1) and the telemetry scheduler (paragraph 4.10.4.2). Like the table tree, variable trees allow filtering by group.

Variable names are displayed in the tree in the format:

`<data type>.<variable name[[array size][...]]>[:bit length]`

Examples: float.bq[1], uint16.faultBits:12

The node icons used in the variable tree indicate if the variable is a bit-wise variable or not (i.e., has a bit length assigned), if the variable is bit-packed with one or more variables, and if the variable belongs to a link (see Table 4). Paragraph 4.6.4.1 provides details on bit-packed variables.

| Icon | Variable type                       |
|------|-------------------------------------|
|      | Non-bit-wise variable               |
|      | Linked non-bit-wise variable        |
|      | Bit-wise variable                   |
|      | Linked bit-wise variable            |
|      | Packed bit-wise variable            |
|      | Linked and packed bit-wise variable |

Table 4. Variable tree icons

#### 4.6.4 Data types

The structure and command tables, and possibly and user-defined table types, contain data type columns. This column is used to set the data type for the referenced parameter (e.g., structure variable or command argument). The data type is either a primitive type or a reference to a structure.

Each primitive data type is derived from one of five base data types: signed integer, unsigned integer, floating point, character, and pointer. The base type, along with the size in bytes, determines the characteristics and usage of the data type. For example, a bit length can be assigned to a variable only if its data type has an integer base type (signed or unsigned), and the bit length is less than or equal to the data type's size (in bits).

The default primitive types supported are shown in Table 5.

| Data Type Name | C-Language Data Type | Number of Bytes | Base Type      |
|----------------|----------------------|-----------------|----------------|
| int8_t         | signed char          | 1               | signed integer |
| int16_t        | signed short int     | 2               | signed integer |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. <i>JSC-#####</i>  | <i>Baseline</i> |
|   | Date: <i>January 2017</i>  | Page 23 of 166  |

|          |                    |    |                  |
|----------|--------------------|----|------------------|
| int32_t  | signed int         | 4  | signed integer   |
| int64_t  | signed long int    | 8  | signed integer   |
| uint8_t  | unsigned char      | 1  | unsigned integer |
| uint16_t | unsigned short int | 2  | unsigned integer |
| uint32_t | unsigned int       | 4  | unsigned integer |
| uint64_t | unsigned long int  | 8  | unsigned integer |
| float    | float              | 4  | floating point   |
| double   | double             | 8  | floating point   |
| char     | char               | 1  | character        |
| string   | char (array)       | >1 | character        |
| address  | void *             | 4  | pointer          |

Table 5. Default primitive data types

A data type with a base type of 'character' is considered a string if the byte size is set to greater than 1. The byte size value in this case is otherwise unused by the application. An array of data type 'string' is treated specially by the application. See paragraph 4.6.1.1 for details.

To the application a pointer represents an address and the actual data type and C type names are irrelevant. However, the application does allow creation of pointers with distinctive names. This is useful, such as in a generated header file to create `typedef` statements for subsequent use in assigning data types to variables (versus using the `void *` data type and type casting each variable appropriately). The application allows creation of any number of pointer data types.

In a data table, when a data type column cell is selected it displays a drop down menu showing the data types. The data types available depend on the usage. In general, in a structure table the data types include primitive types and the names of structures that are not referenced in the hierarchy of the structure being edited (this prevents creating a circular reference). If the structure variable has a bit length or enumeration value then the data types available are limited to primitive types with an integer base type (signed or unsigned). For a command argument only primitive types are displayed, and if the argument has an enumeration value then the data types are limited to primitive types that have an integer base type (signed or unsigned).

The data type manager (paragraph 4.10.3.10) is used to create, modify, and delete the primitive data types.

#### 4.6.4.1 Bit fields

Variables with an integer (signed or unsigned) data type may be assigned as bit fields. A bit field is identified by having a value entered in the structure data table's **Bit Length** column. Variables that are co-located in the table and have the same data type are assumed to be packed together; i.e., these variables occupy the same byte or bytes. The number of variables and bits that are packed is based on the data type's byte size and the bit length of each variable. The bits representing a variable must be contained within a single data type's set of bits. For example, a *uint16* is two bytes, or 16 bits, so bit

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 24 of 166 |

field variables totaling 16 or fewer bits are packed. If three variables of type *uint16* are co-located, with bit lengths of 2, 12, and 5, then the first two variables are packed together ( $2 + 12 < 16$ ), and the third variable occupies its own pair of bytes since its 5 bits won't fit within the first packed pair's 2 unused bits.

Bit-packed variables must have the same telemetry downlink rate. Since the variables are packed together they are downlinked together, even if only a subset of the variables is desired. The table editor accounts for bit-packing by enforcing a common rate among variables that are packed together. In other words, it changes the rates, if needed, of packed variables so that they match. The check for, and update to, a common rate takes place each time an edit is made to the table. If two variables should not be packed then a padding variable must be added between them with the appropriate bit length to ensure the two variables no longer fit within the bit size of the variables' data type.

When transferring variables, such as between trees in the link manager or between the variable tree and messages in the telemetry manager, those that are packed together are automatically moved as a unit, even if not explicitly selected.

#### 4.6.5 Enumerations

Enumerations allow associating a text label with an integer value, and optionally other attributes. Enumerations are useful, for example in displays, since descriptive label text can be substituted for an ambiguous numeric value.

The structure and command tables contain an enumeration column by default. The enumeration's integer value is the value of the parameter described in the same row of the table (the variable name for a structure table and the argument name for a command table). The enumeration's parameter must be an integer data type (signed or unsigned). The structure and command table editors enforce this constraint by not allowing text to be entered into an enumeration cell for which the associated data type is not an integer, and by only displaying integer types in the data type cell if the associated enumeration cell is not blank. The columns named "Data Type" and "Enumeration" are associated by default. For other columns (e.g., the command argument data types and enumerations) the data type and enumeration columns are determined by their respective input type designation (see paragraph 4.8) and are paired based on their order in the table's type definition; the first column designated as containing a primitive input type is associated with the first column designated as containing an enumeration input type, the second primitive input type with the second enumeration type, and so on. The XTCE XML conversion (see paragraph **Error! Reference source not found.**) also checks that the data type is valid for an enumeration, generating an error message if an enumeration is associated with a non-integer data type.

The format of the enumerated values is at the discretion of the user – values can be placed in a single column, separated by a character string chosen by the user, or spread amongst several columns. When the enumeration is used in a script then the script code must use the same separator character(s). However, if enumerated values are to be exported to XTCE XML format then the value must be formatted as follows:

```
<1st enum label> | <1st enum value> [| <other 1st enum attributes>]
[,<2nd enum label> | <2nd enum value> [| <other 2nd enum attributes>]]
[[,...],<nth enum label> | <nth enum value> [| <other nth enum attributes>]]]
```



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 25 of 166 |

The XTCE XML conversion uses only the enumeration label and associated value; any extra attributes are ignored since the XTCE XML format lacks a data field in which to store the extra attribute(s). Scripts, however, can parse and use these extra attributes (e.g., color, limit, etc.). Below is an example of an enumeration with three possible values (0, 1, and 2):

Off | 0 | red, On | 1 | green, Standby | 2 | yellow

In this example the enumerated values 0, 1, and 2 correspond to the labels “Off”, “On”, and “Standby” and the colors “red”, “green”, and “yellow” respectively.

#### 4.6.6 Macros

A macro is a text string used to represent a number or text. Once defined, a macro can be used to replace part or all of the contents of a data table cell. This allows a commonly used string of text to be defined once, then used in as many tables and table cells as desired. If the text subsequently needs to be altered then only the macro's definition need be changed, instead of having to locate and change each table cell where the text is found. An example for such usage would be an enumeration used in multiple cells and/or tables.

Macros are created and their values set or altered using the macro editor, described in paragraph 4.10.3.10. To insert a macro into a table cell first initiate editing in the cell. Position the text cursor within the cell at the point where the macro is to be inserted, or select any existing text the macro is to replace, and press Ctrl-M. A pop-up dialog appears displaying all macros with values that match the input type of the cell being edited (see paragraph 4.8 for information on input types). Use the mouse or keyboard to highlight the macro to insert. If the mouse pointer is hovered over a macro name in the pop up a tool tip pop up appears displaying the macro's value. Once the desired macro is highlighted either press the left mouse button or the Enter key. The macro name is inserted into the table cell, replacing any selected text, bounded by the macro delimiter characters (##), and highlighted to aid in distinguishing it from the non-macro text (see Figure 7). Press the Escape key to remove the macro pop up dialog without inserting a macro.

| Description            | Units   | Data Type | Array Size  |
|------------------------|---------|-----------|-------------|
| ^\$ +?{##MACRO_2##}()z |         | uint8     | ##MACRO_3## |
|                        | MACRO_1 | 16        | ##MACRO_1## |
|                        | MACRO_1 |           |             |
|                        | MACRO_2 |           |             |
|                        | MACRO_3 |           |             |

Figure 7. Example of macro name display and pop-up dialog in a data table

A macro name can also be typed manually into a cell; the delimiter characters (##) must enclose the macro name for the macro to be recognized. A macro that isn't defined may be manually entered into a cell. Until the macro is defined the cell treats the macro name simply as any other text string. Once the macro is defined the cell treats the macro name as a macro.

Multiple macros can be inserted into a cell. However, a macro can't be inserted within another macro (the macro into which the second macro is inserted is no longer recognized as a macro in this case).

If the mouse pointer is hovered over a cell containing a macro a tool tip pop up appears displaying the contents of the cell with each macro name replaced by its value. Pressing Ctrl+Shift-M causes the

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 26 of 166 |

currently selected table to display all cells containing a macro to replace the macro with its value; releasing the Ctrl+Shift-M key restores the macro names in the cells.

When a table's data is retrieved for use in a script or via the web server the option exists to retain the macro names in place of the macro values. See paragraphs 4.11 and 4.10.1.6 for details. An example of use for this is when creating C header files, where a #define statement is used to set a constant that determines array variable size(s). The macro name can be used to set the #define constant's name and value. In the array definition(s) the macro name is retained instead of the using the value so that the #define constant determines the array size (note that the macro delimiter characters must be removed in this example).

## 4.7 Data Fields

Data fields are input fields created by the user for entering information associated with the component to which the field belongs. The fields are assigned names and an input type that constrains the values that can be entered into the field. Data fields can be associated with data tables and groups.

A data field can be used to enter a piece of information for a data table that doesn't fit with a table's row and column format. An example is a message identification (ID) number for a top-level structure table – the message ID applies to the entire table, not a specific row within it. A column could be added for information such as the message ID, but having the same value repeated for each row is both wasteful in storage as well as prone to errors (if every value doesn't match). Any number of data fields (including none) can be associated with each table. Default data fields may be assigned to a table type, so that every table created of that type automatically has the default fields in place. Fields can also be assigned to individual tables – there is no requirement for the tables to have the same fields.

Similarly, table groups can be assigned data fields. For example, a group can be created that consists of all of the tables for a specific CFS application, so that the group represents the application. Applications have data associated with them that isn't appropriate for storage in a table, such as the application schedule rate or execution time. In this case a data field can be assigned to the group to hold the information. Groups designated as CFS applications are automatically assigned certain data fields; see paragraph 4.10.3.8.

### 4.7.1 Data field editor

This section provides details on use of the data field editor (Figure 8). See paragraph 4.10.3.9.4.1 for information specific to adding default data fields, and paragraph 4.10.3.2.5.1 to information specific to adding fields to a particular table. In either case the editor operation is the same.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 27 of 166 |

| Field Name      | Description | Size | Input Type  | Required                 | Applicability |
|-----------------|-------------|------|-------------|--------------------------|---------------|
| Owner           |             | 4    | Text        | <input type="checkbox"/> | All tables    |
| Message ID Name |             | 20   | Text        | <input type="checkbox"/> | Parents only  |
| Message ID      |             | 7    | Hexadecimal | <input type="checkbox"/> | Parents only  |

Buttons: + Ins Row, ↑ Up, + Separator, ↶ Undo, ↷ Update, - Del Row, ↓ Down, + Break, ↷ Redo, ✖ Close

Figure 8. Data field editor

There are five or six columns in the field editor:

- Field Name** This is the name of the data field. The name can be of any length and can contain letters, numerals, and punctuation characters. When the field is displayed below a table or table type the field name is the label shown immediately to the left of the input text field that is used to contain the field's value. The field name is also used if the data field is referenced from a script (see paragraph 4.11). The field name is required.
- Description** The field description is use to describe the content of the data field. The description appears as tool tip text whenever the mouse pointer hovers over the field in a table or table type editor. HTML tags may be entered to provide formatting for the displayed tool tip text. This column may remain blank.
- Size** The field size defines the width, in characters, of the data field's input text field. Due to padding and font differences, the actual field width may appear slightly larger. The size must be a positive integer, and is required.
- Input Type** The field input type constrains the type of value entered into the data field's input text field. If the value entered into the data field doesn't conform to field's specified input type then a warning message dialog is displayed and the field reverts to its previous value. The input types are selectable from the combo box pull-down menu that appears when a cell in the Input Type column is selected. See paragraph 4.8 for information on the available input types.
- Required** The Required column displays a check box, initially unselected. Selecting the check box indicates that the field is required. When the field is displayed in the table or table editor the input text field is highlighted in yellow as long as the field's input text field value is empty. The application does not enforce the user to input data into the fields marked "required"; the highlighting is used merely as a reminder that the field value is considered necessary and should be filled.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 28 of 166 |

**Applicability** This column only appears when assigning default data fields in the table type editor for structure tables. It allows the propagation of the specified field to all tables, parent tables only, or child tables only. Select the applicability for a field from the combo box pull down menu that appears when the applicability cell is selected. The default is *All tables*.

The order that the rows appear in the field editor determines the order of appearance in the table editor, group manager, or table type editor. Field definition rows may be rearranged as desired by first selecting a cell in one or more rows, then pressing the **Up** or **Down** buttons to move the selected row relative to the ones not selected. The editor columns can be sorted by selecting the column header with the mouse pointer and pressing the left mouse button. The rows are first sorted in ascending order, depending on the selected column's contents. Selecting the column again sorts in descending order, and a third selection restores the rows to their original order.

Line separators and line breaks may be inserted as rows using the **Separator** and **Break** buttons respectively. Without these breaks the data fields, when displayed in a table or table type editor, are arranged end to end, wrapping to the next line when the width of the editor is reached. The line break forces the next data field to the next row regardless of the editor width constraint. The line separator does the same, except that a dividing line is drawn between the rows where the separator is inserted. These breaks can be used to aid in grouping related data fields.

The field editor button commands are described below:

|                  |  |
|------------------|--|
| <b>Ins Row</b>   | The editor is initially empty unless the table or table type editor from which it's invoked has any previously defined fields. To add a field first select the <b>Ins Row</b> button; a new field definition row is inserted into the editor. Additional rows can be added in the same manner. The insertion point is dependent on the currently selected row in the editor; if no row is selected then the new row is added at the bottom. The empty row has the Field Name and Size columns highlighted in yellow. The highlighting indicates that these columns are required and must have values assigned. |
| <b>Del Row</b>   | One or more field definition rows may be deleted by first selecting a cell in the target row(s), then pressing the <b>Del Row</b> button. The selection of multiple rows is constrained to contiguous rows; i.e., rows cannot be skipped.  |
| <b>Up</b>        | The order that the rows appear in the field editor determines the order of appearance in the table or table editor. Field definition rows may be rearranged as desired by first selecting a cell in one or more rows, then pressing the <b>Up</b> button to move the selected row(s) up a row relative to the ones not selected.   |
| <b>Down</b>      | Similar to the <b>Up</b> button action, except that selected row(s) are moved down a row relative to the ones not selected.  |
| <b>Separator</b> | Inserts a line separator below the currently selected field's row.   |
| <b>Break</b>     | Inserts a line break below the currently selected field's row.   |
| <b>Undo</b>      | Undoes the last action performed (typing, paste, insert, delete, redo, etc.).  |
| <b>Redo</b>      | Reverses the last action undone (typing, paste, insert, delete, redo, etc.).   |
| <b>Update</b>    | Selecting the Update button applies the data field definitions currently displayed in the field editor to the table or table type editor from which the field editor was invoked.  |

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 29 of 166 |

Any existing fields that are in the table or table type editor are deleted and replaced by the new definitions. However, these changes are not stored in the database – this is only accomplished when the **Store** button in the table or table type editor is selected.

**Close** Closes the field editor window. If any changes made have not been applied using the **Update** button then a dialog appears allowing the user to confirm discarding the updates or to cancel closing the editor.

## 4.8 Input Types

Data table columns and data fields are assigned an input type in the table type editor (see paragraph 4.10.3.6) and the data field editor (see paragraph 4.7) respectively. The input type constrains the type of value entered into a data table cell or data field. Excess whitespace characters (spaces, tabs, etc.) are ignored and eliminated when the text is formatted (exception: spaces between characters in a text string are retained). Leading plus (+) signs and zeroes are allowed for non-negative integer and floating point values, but are ignored and eliminated. In the editor select the row in the Input Type column corresponding to the data table column or data field. A combo box pull down menu appears with the following selections:

|                             |   |
|-----------------------------|---|
| <b>Text</b>                 | This type allows letters, numerals, and punctuation characters. Text is the default data type   |
| <b>Alphanumeric</b>         | This type allows letters, numerals, and underscore characters. A numeral may not begin the text string. Alphanumeric text is appropriate for variable names   |
| <b>Boolean</b>              | Only the integers 0 or 1 are allowed for the boolean type   |
| <b>Integer</b>              | The integer data type allows input of any integer value, positive, negative, or zero  |
| <b>Positive integer</b>     | Only positive integers are allowed for this type, but not zero or negative values   |
| <b>Integer &gt; 1</b>       | Only integer values greater than 1 are allowed for this type  |
| <b>Non-negative integer</b> | Similar to the positive integer type, except that zero is allowed   |
| <b>Negative integer</b>     | Similar to the positive integer type, except that only negative integer values are allowed  |
| <b>Floating point</b>       | The floating point type allows floating point values to be entered; i.e., values with decimal components in the form #.### (e.g., 3.14). Leading plus (+) signs and zeroes are allowed, but are ignored |
| <b>Positive float</b>       | Only positive floating point values are allowed for this type, but not zero or negative values  |
| <b>Non-negative float</b>   | Similar to the positive float type, except that zero is allowed   |
| <b>Negative float</b>       | Similar to the positive float type, except that only negative values are allowed  |
| <b>Hexadecimal</b>          | This type allows only hexadecimal digits to be entered (0 – 9, A – F, and a – f). The hexadecimal digits may optionally be preceded by “0x”   |
| <b>Rate</b>                 | Special format used to designate telemetry sample rate columns. Allows positive integer values and values in the form “1 / #” where # is a positive integer value                                       |

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 30 of 166 |

|                                  |   |
|----------------------------------|---|
| <b>Array index</b>               | Special format used to designate the array size column. Allows 1 or more integer values greater than 1, separated by commas. For the array size column each value represents an array dimension size (e.g., if the array size is 2, 3, 4 then the associated array size is defined by <i>arrayName</i> [2][3][4])   |
| <b>Argument name</b>             | Special format used to designate a command table argument name column. This type allows letters, numerals, and underscore characters. A numeral may not begin the text string   |
| <b>Enumeration</b>               | Special format used to designate a column containing enumerated values. This type allows letters, numerals, and punctuation characters  |
| <b>Minimum</b>                   | Special format used to designate a column containing minimum values. This type allows boolean, integer, floating point, and hexadecimal values depending on the data type associated with it. If the associated data type is missing or blank then the minimum value cell is blanked and cannot be edited. The minimum column is automatically paired with a maximum column (if present); if multiple maximum columns are present then pairing is done in order of column appearance in the table type definition. When paired the minimum value is constrained to be less than or equal to the maximum value |
| <b>Maximum</b>                   | Special format used to designate a column containing maximum values. This type allows boolean, integer, floating point, and hexadecimal values depending on the data type associated with it. If the associated data type is missing or blank then the maximum value cell is blanked and cannot be edited. The maximum column is automatically paired with a minimum column (if present); if multiple minimum columns are present then pairing is done in order of column appearance in the table type definition. When paired the minimum value is constrained to be less than or equal to the maximum value |
| <b>Primitive</b>                 | This type causes a combo box pull down menu to appear when the cell is selected. The menu contains all of the primitive data types. This selection is not available in the data field editor  |
| <b>Primitive &amp; Structure</b> | This type causes a combo box pull down menu to appear when the cell is selected. The menu contains all of the primitive data types along with the names of all the prototype structure tables. This is primarily for use in defining the Data Type column in structure tables but can be used elsewhere. This selection is not available in the data field editor   |

## 4.9 Data Streams

CCDD supports the definition and use of multiple data streams. In this context a data stream refers to an uplink/downlink path; for example, serial, Ethernet, or radio. Each data stream has its own set of rate parameters (see paragraph 4.10.4.4). Addition of a rate stream is accomplished by adding a new sample rate column to a structure table definition (see paragraph 4.10.3.6 for information on altering a table type). A rate column is designated by assigning the column an input type of 'Rate' (see paragraph 4.8 for information on input types). A telemetry parameter can be assigned a rate in each of the defined data streams. The link manager (paragraph 4.10.4.1) allows linking telemetry parameters for allocation in the downlink messages. These linkages are specific to a data stream. The data stream can be

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 31 of 166 |

assigned a name different from its associated rate column name in the rate parameter dialog (paragraph 4.10.4.4).

## 4.10 Command Menu

### 4.10.1 File menu

The **File** menu provides selections for connecting to the database, altering the database connection properties, reading and printing application logs, updating the application's overall appearance, and exiting the program.

#### 4.10.1.1 Select user

When the **Select user** command is issued, if any table editor or the table type editor is open and has unsaved changes then a confirmation dialog first appears, allowing the user to choose whether or not to continue with the user change, discarding the unsaved changes, or to cancel the user change. If there are no unsaved changes or if the user confirms discarding the changes then the editors are closed and the Select User dialog is displayed. The dialog allows entering the user name and/or user password. The appearance of the dialog is dependent on whether or not a connection is currently established with the PostgreSQL server. If no connection exists then the dialog appears as in Figure 9, and both the user and user password must be entered. If a connection to the server does exist (i.e., if changing to another user from one already connected to the server) then the dialog appears as in Figure 10. For this case the user text field is replaced by radio buttons providing an alphabetized list of the users registered in the PostgreSQL server.

Select or type in a user name and, if required by the server, provide the password in the Password field, then select the **Okay** button. An attempt is then made to establish a connection as the indicated user with the most recently selected or open project's database. If a project's database is open when the user is changed and the newly selected user does not have access privileges to this project then the project's database is closed. Select the **Cancel** button to exit the dialog without changing the user.



Figure 9. Select User dialog (no server connection)

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 32 of 166 |

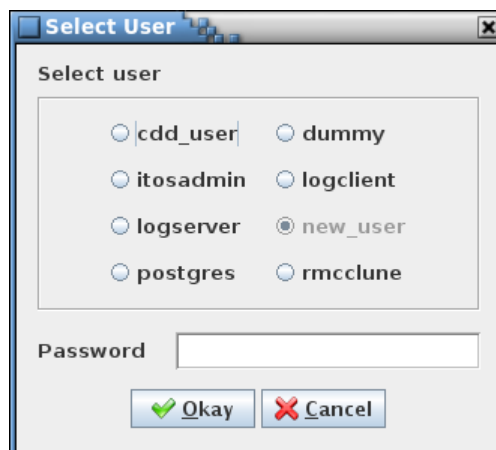


Figure 10. Select User dialog (server connected)

#### 4.10.1.2 Database server

When the **Database server** command is issued, if any table editor or the table type editor is open and has unsaved changes then a confirmation dialog first appears, allowing the user to choose whether or not to continue with the property change, discarding the unsaved changes, or to cancel the property change. If there are no unsaved changes or if the user confirms discarding the changes then the editors are closed and the Database Server dialog is displayed (see Figure 11). The dialog allows entering the server host name and/or server port number.

The dialog allows selection of a PostgreSQL server on the local or a remote host. Enter the server host name and, if needed, the server port number, then select the **Okay** button. If the host field is empty then the default host name, localhost, is used. The default port number for the PostgreSQL server is 5432. An attempt is then made to establish a connection as the current user with the most recently selected or open project's database. If a project's database is open when the server properties are changed and the user does not have access privileges to this project on the newly selected server then the project's database is closed. Select the **Cancel** button to exit the dialog without changing the database server properties.



Figure 11. Database server dialog

#### 4.10.1.3 Read log

The **Read log** command causes the Open Event Log file selection dialog to be displayed. Navigate to the location of the CCDD event log file, highlight it using the mouse or keyboard controls, and select **Open** to open the log in a window similar to the main program window. The log file names are in the format CCDD-YYYYMMDD-hhmmss.log, where YYYYMMDD is the year, month, and day, and hhmmss is the hour, minute, and second when the log was created. Select **Cancel** to close the file selection dialog



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 33 of 166 |

without opening a log file. See paragraph 4.5 for details on the event log columns and filter selections. The log window may be resized. The **Search** button displays the event log search dialog (see paragraph 4.10.1.5); the log can be searched for a user-specified text string. Select the **Print** button to open a printer selection dialog in order to print a copy of the log to the selected printer. Select **Close** to close the log window.

#### 4.10.1.4 Print log

The **Print log** command causes a dialog to appear allowing selection of a printer in order to print a copy of the current session's event log; i.e., the log displayed in the main application window.

#### 4.10.1.5 Search log

The **Search log** dialog provides a means of searching the current session's event log for a specified text string (see Figure 12). Case sensitivity for the search is governed by the **Ignore text case** check box. Enter the search text in the input field and select the **Search** button. The search results are displayed in the table at the bottom of the search dialog. The first column, **Log Index**, shows the log entry's index number where a match is found. The second column, **Column Name**, provides the column where the match occurs in the event log. The last column, **Context**, displays the string from log entry containing the search text, with the search text highlighted. The full length of the log message text is searched (and displayed, if a match is found), even if the message is truncated in the event log due to length restrictions. Pressing the Ctrl-S key sequence while the main application window has the focus also displays the event log search dialog.

The search results can be output to a file or printer by selecting the **Print** button. To exit the search dialog select the **Close** button.

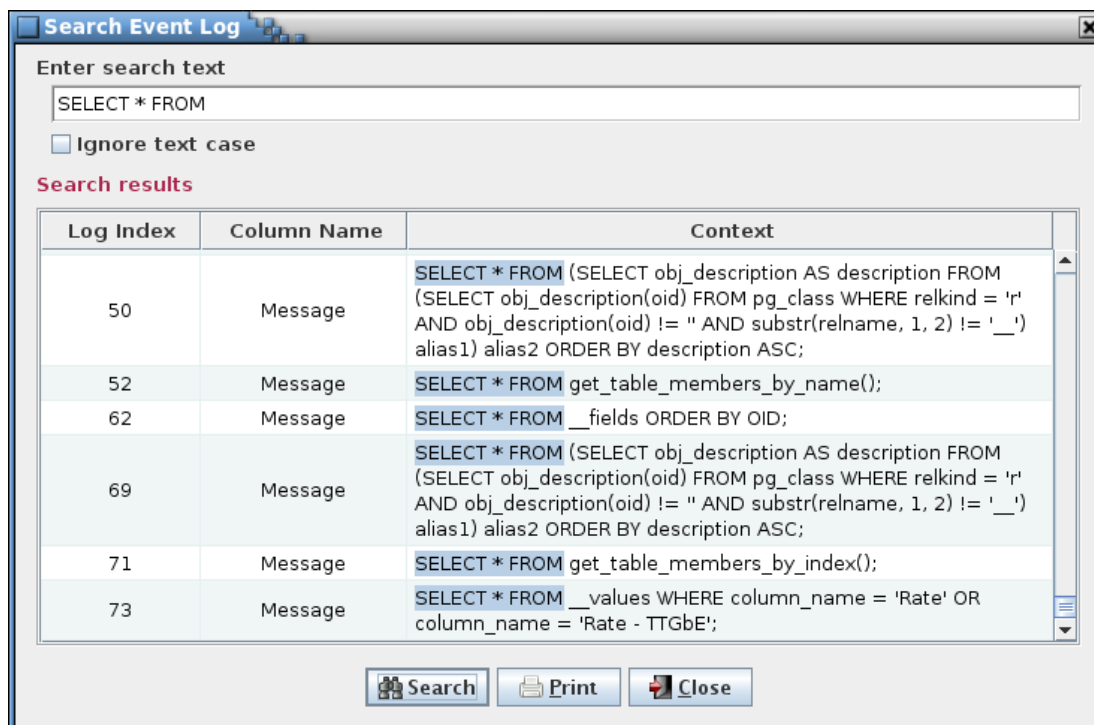


Figure 12. Search event log dialog

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 34 of 166 |

#### 4.10.1.6 Web server

The embedded Jetty web server allows web-based applications access to a project's data. The web server must be started before any requests are made. If the application is running the **Enable server** command (paragraph 4.10.1.6.1) is used to start and stop the server. The default is for the server to be disabled. Command line options are available to allow the server to be started at program start-up, with or without the GUI enabled. See Table 1 for the web server command line arguments.

All requests are directed to the currently open project database. The query format is:

*host:port/component<?attribute<=name>>*

The *host* name is the network name or IP address on which the CCDD application, with the web server active, is operating (localhost if active on the same machine as the requesting application). The *port* number is the port to which the server is assigned to listen (the default is 7070; this can be changed via command line command or menu option). The *component*, *attribute*, and *name* portions of the request determine the data returned. Data for tables, groups, applications, and the telemetry and application scheduler is available. Data may be requested for a single table, group, or application, or for all of the given component. List containing the names of all tables, groups, or applications can be requested. Table 6 contains the recognized *component*, *attribute*, and *name* combinations.

If a table contains macro references then the table values default to replacing the macro names with the corresponding macro values, as defined in the macro editor (see paragraphs 4.6.6 and 4.10.3.10 for more information relating to macros). The server "all" and "data" requests can be made to return the table data with the macro names displayed in place of the macro values by appending **;macro** (or **;macros**) at the end of the *name* portion of the request (example: the request *table?=<macro>* returns the table data for all tables with the macro names displayed).

The data is returned to the requesting application in JavaScript Object Notation (JSON) format per the Output column in Table 6. For the initial request the user is prompted for a valid PostgreSQL server user name and password. Additionally, this user must have access to the project open in the CCDD application hosting the web server.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-####  | Baseline       |
|   | Date: January 2017   | Page 35 of 166 |

| Request        |                |            | Returned Information   | Output  |
|----------------|----------------|------------|--|---|
| Component      | Attribute      | Name       |  |   |
| table<br>table | all (or blank) |            | Type, description, size (if a structure), data, and data fields for all data tables  | [<first table's name, type, description, data, and data fields; see below for format>,<next table's information<,...>>]   |
|                | all (or blank) | table name | Type, description, size (if a structure) , data, and data fields for the specified data table. <i>table name</i> is case insensitive | {"table name":"table name","table type":"type","table description":"description","table size":size,"table data":[{"columns":["column name"<,"next column name<,...>>]},{"row 0":["column value"<,"next column value<,...>>]},{"next column<,...>>]},{"next row<,...>>"],"table data fields":[{"field columns":["Field Name", "Description", "Input Type", "Value"]},{"field 0":["field name","description","input type","value"]}]<,"next field's data<,...>>}} |
|                | data           |            | Data for all data tables   | [<first table's name and data information; see below for format>,<next table's name and data information<,...>>]  |
|                | data           | table name | Data for the specified data table. <i>table name</i> is case insensitive   | {"table name":[{"columns":["column name"<,"next column name"<,...>>]},{"row 0":["column value"<,"next column value"<,...>>]},{"next row<,...>>}]}   |
|                | description    |            | Descriptions for all tables  | [<first table's name and description; see below for format>,<next table's name and description<,...>>]  |
|                | description    | table name | Description for the specified table. <i>table name</i> is case insensitive   | {"table name":"description"}  |
|                | fields         |            | Data field information all tables  | [<first table's data fields; see below for format>,<next table's data fields<,...>>]  |

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 36 of 166 |

|                |  |                   |  |   |
|----------------|--|-------------------|--|---|
|                | fields   | <i>table name</i> | Data field information for the specified table. <i>table name</i> is case insensitive    | { <i>"table name"</i> : [{ <i>"field columns"</i> : [ <i>"Field Name"</i> , <i>"Description"</i> , <i>"Input Type"</i> , <i>"Value"</i> ]}, { <i>"field 0"</i> : [ <i>"field name"</i> , <i>"description"</i> , <i>"input type"</i> , <i>"value"</i> ]}}<,next field's data<,...>>}   |
|                | names  |                   | Names of all tables of all table types   | [< <i>table type and names</i> >,<next table type and names<,...>>]   |
|                | names  | <i>table type</i> | Names of all tables of the specified table type. <i>table type</i> is case insensitive   | { <i>"table type"</i> , [ <i>"table name"</i> <,"next table name"<,...>>]}  |
|                | size   |                   | Size (in bytes) for all structure tables   | [< <i>first table's name and size; see below for format</i> >,<next table's name and size<,...>>]   |
|                | size   | <i>table name</i> | Size (in bytes) for the specified structure table. <i>table name</i> is case insensitive | { <i>"table name"</i> :size}  |
| proto_table    | Same requests as for table above, except only table information for prototype tables is returned |                   |  |   |
| instance_table | Same requests as for table above, except only table information for instance tables is returned  |                   |  |   |
| group          | all (or blank)   |                   | Description, associated table(s), and data fields for all groups                         | [< <i>first group's name, description, tables, and data fields; see below for format</i> >,<next group's information<,...>>]  |
|                | all (or blank)   | <i>group name</i> | Description, associated table(s), and data fields for the specified group                | { <i>"group name"</i> : <i>"group name"</i> , <i>"group description"</i> : <i>"description"</i> , <i>"group tables"</i> : [ <i>"table name"</i> <,"next table name"<,...>>], <i>"group data fields"</i> : [{ <i>"field columns"</i> : [ <i>"Field Name"</i> , <i>"Description"</i> , <i>"Input Type"</i> , <i>"Value"</i> ]}, { <i>"field 0"</i> : [ <i>"field name"</i> , <i>"description"</i> , <i>"input type"</i> , <i>"value"</i> ]}}<,next field's data<,...>>} |
|                | tables   |                   | Tables associated all groups   | [< <i>first group's name and tables; see below for format</i> >,<next group's name and tables<,...>>]   |

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 37 of 166 |

|             |   |  |   |   |
|-------------|---|--|---|---|
|             | tables  | <i>group name</i>  | Table(s) associated with the specified group. <i>group name</i> is case sensitive   | {"group name":["table name"<,"next table name"<,...>>"]}  |
|             | description   |  | Data field descriptions for all groups  | [<first group's name and description; see below for format>,<next group's name and description<,...>>]  |
|             | description   | <i>group name</i>  | Description for the specified group. <i>group name</i> is case sensitive            | {"group name":"description"}  |
|             | fields  |  | Data field information for all groups   | [<first group's data fields; see below for format>,<next group's data fields<,...>>]  |
|             | fields  | <i>group name</i>  | Data field information for the specified group. <i>group name</i> is case sensitive | {"group name":{"field columns":["Field Name", "Description", "Input Type", "Value"]},"field 0":["field name","description","input type","value"]<,<next field's data<,...>>}  |
|             | names   |  | Names of all groups   | {"group names":["group name"<,<next group name<,...>>]}   |
| application | Same requests as for group above, except only group information for groups representing a CFS application is returned |  |   |   |
| scheduler   | telemetry   | <data stream name>,<br><header size (bytes)>,<br><message ID name data field name>,<br><optimize flag (true or false)> | Telemetry scheduler's copy table entries  | {"data stream":"stream name","header size":"size","optimized":"true or false","copy table":{"column names":["Input Message ID","Input Offset","Output Message ID","Output Offset","Number of Bytes","Root Table","Variable Path"]},"row 0 values":["input ID","input offset","output ID","output offset","output size in bytes","root table name","variable path"]<,<next row<,...>>} |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. <i>JSC-####</i>   | <i>Baseline</i> |
|   | Date: <i>January 2017</i>  | Page 38 of 166  |

|          |  |  |  |   |
|----------|--|--|--|---|
|          | application  |  | Application scheduler's schedule table entries |   |
| datatype |  |  | Data type definitions                          | <i>{"user name":"user-defined name","c name":"C-language name","size":"size in bytes","base type":"base data type"}&lt;,next data type definition&lt;,...&gt;&gt;]}</i> |
| macro    |  |  | Macro definitions                              | <i>{"macro name":"macro value"}&lt;,next macro definition&lt;,...&gt;&gt;]}</i>   |
| shutdown | Close the web server and project database, and exit the CCDD application |  |  |   |

Table 6. Web data access commands

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 39 of 166 |

#### 4.10.1.6.1 Enable server

Selecting the **Enable server** command toggles between starting and stopping the web server. When enabled, the web server responds to web-based queries. When disabled, the server ignores web queries.

#### 4.10.1.6.2 Select port

The **Select port** command displays a dialog (Figure 13) that allows selection of a port number for the embedded Jetty web server. Enter the server port number, then select the **Okay** button. If the web server is active then it's automatically restarted using the new port number. Select the **Cancel** button to exit the dialog without changing the server port.



Figure 13. Web Server dialog

#### 4.10.1.7 Appearance

The **Appearance** command displays the Application Appearance dialog (Figure 14), which allows choosing the style, or “look and feel” (L&F), applied to the program’s GUI components. Different L&Fs can change the shape and color scheme of the graphical components (see Figure 15). The default is “Metal”, the standard L&F provided with Java. The actual list of L&F selections displayed in the dialog is dependent on the available L&Fs loaded on the host machine. When the radio button associated with the desired L&F is selected the Application Appearance dialog, main application window, and any other open CCDD windows are immediately redrawn to reflect the L&F chosen. Select the **Close** button to exit the dialog.

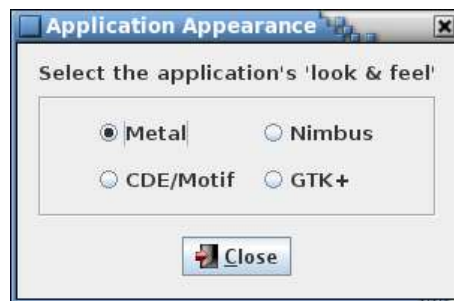


Figure 14. Application Appearance dialog

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 40 of 166 |

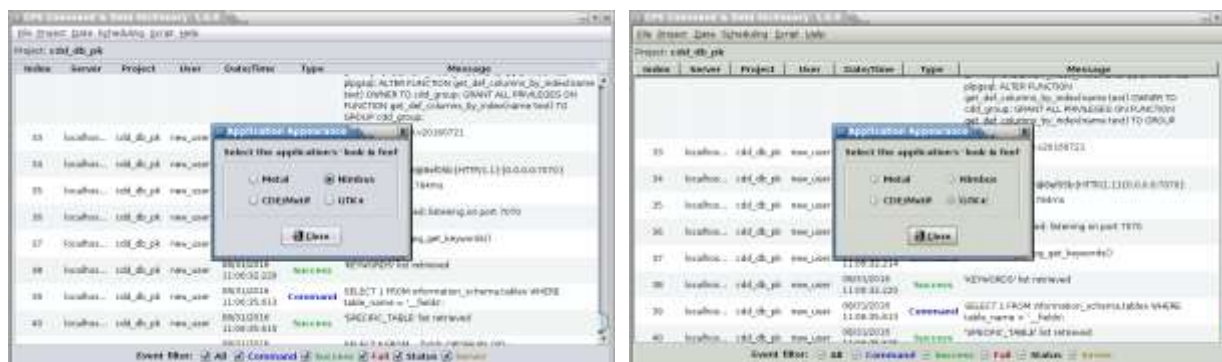


Figure 15. Example look and feel differences

#### 4.10.1.8 Exit

Choosing the **Exit** command displays a dialog so that the user can confirm whether or not to exit the application. Select **Okay** to exit CCDD. If there are unsaved changes to a table editor or the table type editor then the user is queried whether or not to continue. If **Okay** is selected the open editors are closed (any unsaved changes are discarded), the main application window is closed, and the application exits. Select **Cancel** to close the dialog without exiting the application.

### 4.10.2 Project menu

The **Project** menu contains commands for interacting with a project's database.

Each project has a locked/unlocked status flag. This flag is checked by the application when attempting to access a project. Project access is required for the **Open**, **New**, **Rename**, **Copy**, and **Delete** commands described in this section. If the flag indicates the project is unlocked the command proceeds. If the flag indicates the project is locked, the project access is denied and the specified operation is terminated. Access failure results in display of a database error dialog and the failure is written to the event log. The lock status is set to "locked" for an open project database. When the project database is closed (e.g., when exiting the CCDD application) the flag is set to "unlocked". Abnormal termination of the CCDD application can result in a project database retaining a locked status. The **Unlock** command (paragraph 4.10.2.9) allows clearing a project's lock status.

*Note: In the project dialogs below, only those project databases for which the current user is allowed access are displayed.*

#### 4.10.2.1 Open

Selecting the **Open** command results in a dialog being displayed that shows the CCDD project databases, along with their descriptions, that are available in the PostgreSQL server (see Figure 16). The currently open project database is shown selected and grayed out. Other projects that are open in another instance of the CCDD application are also grayed out and have their radio button disabled. Select a project and then the **Open** button to open the selected project's database. The currently open project is first closed, along with any open table or table type editors. If the editors have any unsaved changes then a confirmation dialog appears, allowing the user to choose whether or not to continue with the project change, discarding the unsaved changes, or to cancel the project change. Select **Cancel** to leave the open project unchanged.



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 41 of 166 |

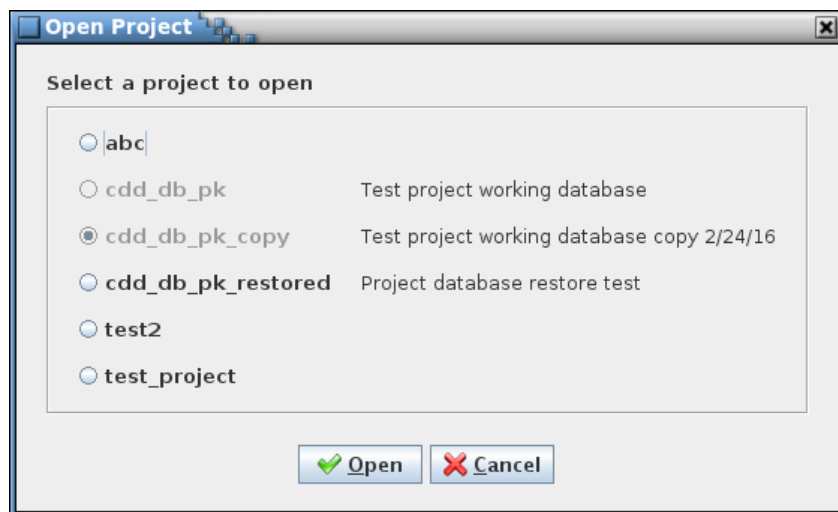


Figure 16. Select Project dialog

#### 4.10.2.2 Close

When the **Close** command is selected the currently open project database is closed, along with any open data table or table type editors. If the editors have any unsaved changes then a confirmation dialog appears, allowing the user to choose whether or not to continue with closing the project, discarding the unsaved changes, or to cancel closing the project.

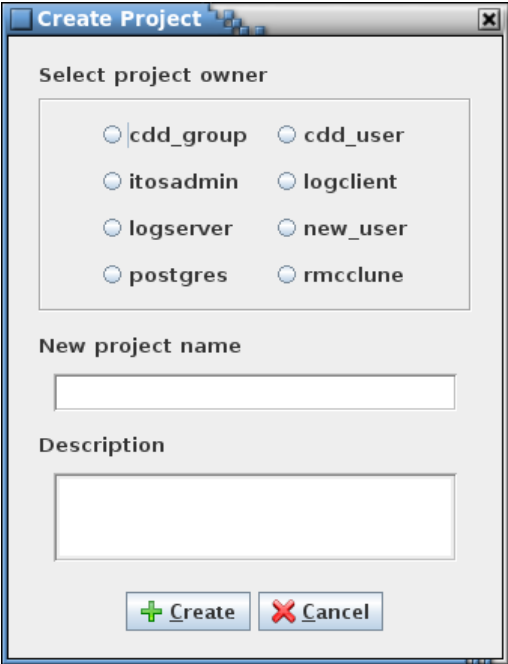
#### 4.10.2.3 New

The Create Project dialog (see Figure 17) appears when the **New** command is chosen, which allows creation of a new CCDD project database. A project owner must be selected from the list of available roles stored in the server, and a name supplied for the new project. Optionally, a description can be entered for the project.

The choice of owner should take into account the number of users that require access to the project's database. If only a single user needs access then the user can be selected as the owner. If multiple users need access then a group role should be created and this role assigned as the owner. All users requiring access would then need to be made members of this group role. Note that any user with superuser status can access the project's database regardless of the owner. See paragraph 4.4 for further information regarding setup of the PostgreSQL server.

The project name can only contain the characters a - z (upper and lower case are accepted, but all characters are changed to lower case), 0 -9, and the underscore (\_), and must begin with an alphabetic character. The name length cannot exceed 63 characters. Also, the project name must be unique; it may not be the same as the name of another project database existing in the server. The description can contain any characters and, optionally, can be formatted using HTML tags.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 42 of 166 |



The image shows a 'Create Project' dialog box. It has a title bar with a close button. Inside, there's a section 'Select project owner' with eight radio buttons arranged in two columns: cdd\_group, cdd\_user, itosadmin, logclient, logserver, new\_user, postgres, and rmclune. Below this is a 'New project name' text field and a 'Description' text area. At the bottom are two buttons: '+ Create' and 'X Cancel'.

Figure 17. Create Project dialog

#### 4.10.2.4 Rename

When the **Rename** command is selected the Rename Project dialog is displayed (see Figure 18). This dialog allows an existing project to be renamed, its description to be altered, or both. When one of the radio buttons representing a project's database is selected the name and description appear in the fields below the radio button panel. Projects that are open in another instance of the CCDD application cannot be renamed and are grayed out with their radio button disabled. See paragraph 4.10.2.3 for constraints on the project name and description. When the **Rename** button is selected the project and description are updated. Note that this dialog can, if desired, be used to alter only the project's description. Selecting the **Cancel** button closes the dialog without making any alterations.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 43 of 166 |

**Rename Project**

Select a project to rename

- ☒ abc
- ☐ cdd\_db\_pk      Test project working database updated
- ☐ cdd\_db\_pk\_copy      Test project working database copy 2/24/16
- ☐ cdd\_db\_pk\_copy2      Test project working database
- ☐ cdd\_db\_pk\_restored      Project database restore test
- ☐ test2
- ☐ test\_project

New project name

Description

**Rename**    **Cancel**

Figure 18. Rename Project dialog

#### 4.10.2.5 Copy

When the **Copy** command selected the Copy Project dialog is displayed (see Figure 19). This dialog allows an existing project's database to be copied. When one of the radio buttons representing a project is selected the name and description appear in the fields below the radio button panel. The project name has the text '\_copy' automatically appended, though the copy's name and description can be altered as desired. See paragraph 4.10.2.3 for constraints on the project name and description. When the **Copy** button is selected the selected project's database is copied, using the copy name and description. Selecting the **Cancel** button closes the dialog without making a copy.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 44 of 166 |

**Copy Project**

Select a project to copy

- ☐ abc
- ☒ cdd\_db\_pk Test project working database updated
- ☐ cdd\_db\_pk\_copy Test project working database copy 2/24/16
- ☐ cdd\_db\_pk\_copy2 Test project working database
- ☐ cdd\_db\_pk\_restored Project database restore test
- ☐ test2
- ☐ test\_project

Project copy name

cdd\_db\_pk\_copy

Description

Test project working database updated

Copy Cancel

Figure 19. Copy Project dialog

#### 4.10.2.6 Delete

The **Delete** command allows one or more project databases to be deleted. The Delete Project(s) dialog (Figure 20) appears when the command is issued. Projects that are open, in this or another instance of the CCDD application, cannot be deleted and are grayed out with their radio button disabled. After selecting a project (or projects) to delete, selecting the **Delete** button removes the project database(s) from the server and selecting the **Cancel** button exits the dialog without deleting any projects. If **Delete** is selected a confirmation dialog is displayed for each selected project; selecting **Okay** continues with the delete operation for that project, and **Cancel** ignores the indicated project and does not delete it.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 45 of 166 |

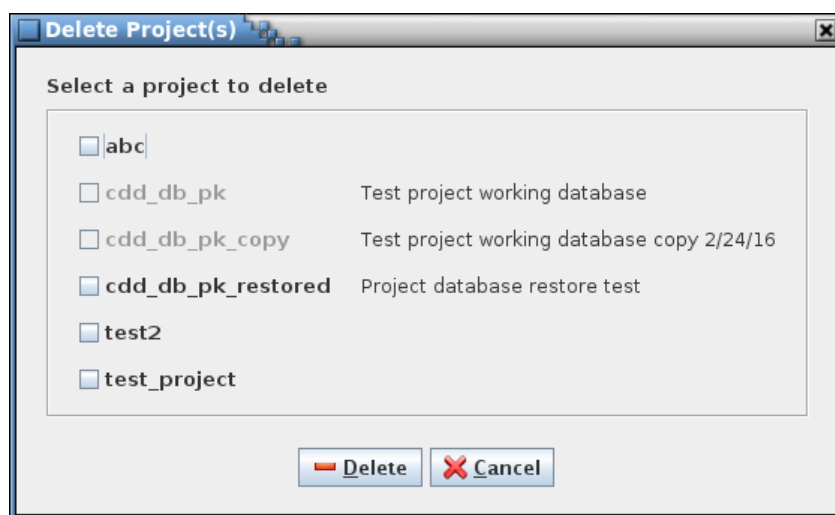


Figure 20. Delete Project dialog

#### 4.10.2.7 Backup

The **Backup** command allows the user to create a backup of the currently open project's database. A file selection dialog is displayed for choosing the location and name of the backup file (Figure 21). The backup file extension is '.dbu'. Select the **Backup** button to proceed; if the file selected already exists an overwrite confirmation dialog appears. The backup file is created using the PostgreSQL *pg\_dump* command. This produces a PostgreSQL script file, in plain ASCII text, that has all of the commands necessary to create the project's database as it currently exists. The backup file makes it easy to transfer the database between servers and platforms. The **Restore** command, detailed in paragraph 4.10.2.8, uses the file generated by the **Backup** command to recreate a project's database. Selecting the **Cancel** button exits the dialog without creating a backup.

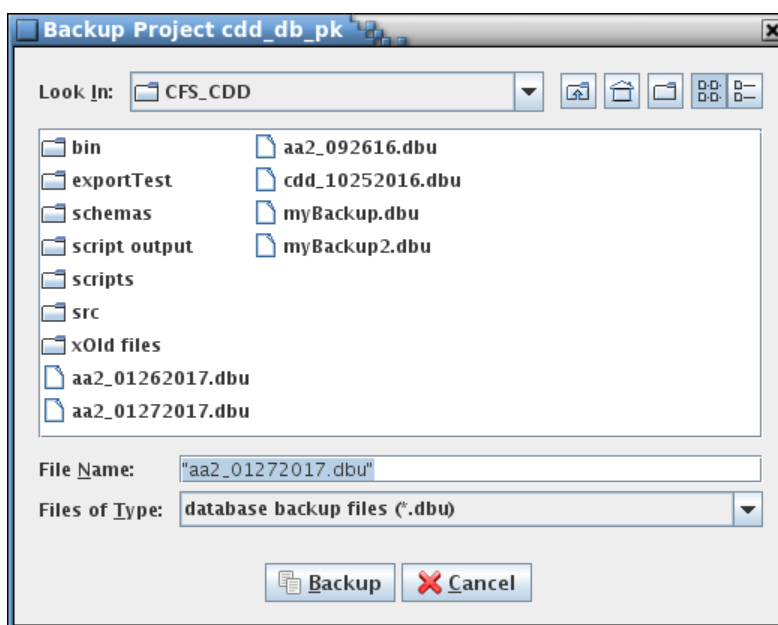


Figure 21. Backup Project dialog

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 46 of 166 |

#### 4.10.2.8 Restore

The **Restore** command allows (re)creating a project's database on a server using a PostgreSQL script file created using the **Backup** command (see paragraph 4.10.2.6). Using the dialog that appears, navigate to the desired backup file, select it, and press the **Restore** button; the project database recorded in the script file is restored to the server. The name of the restored database is its original name with “\_restored” appended, and the owner is set to its original owner.

If the name of the restored project's database would match that of an existing database then a sequence number is appended to the restored database's name. For example, if the database *abc* is restored and the database *abc\_restore* already exists then the database is restored as *abc\_restore1*; if *abc\_restore1* already exists then *abc\_restore2* is used, and so on until an unused name is found.

#### 4.10.2.9 Unlock

The **Unlock** command allows the locked status to be changed to “unlock” for a project database. This command is intended to be used to remove a lock from a project that remains locked after abnormal termination of the CCDD application. The Unlock Project(s) dialog (Figure 22) appears when the command is issued. Though all projects are displayed, only those that are locked have the check box enabled.

The locked/unlocked status is displayed beside the project database name along with the names of the users that have active connections to the project. A project is shown as “Current” if opened by the current instance of the CCDD application and is shown as “in use by” the current user. “Locked” is displayed if a project is in use by another instance of the CCDD application or was open when the application terminated abnormally. “Unlocked” indicates that no other instance of the CCDD application has the project open. Other applications may have active connections to the project (e.g., the PostgreSQL command line interface application, *psql*). The users for these non-CCDD connections are also shown in the “in use by” list. Referring to Figure 22, project “abc” is unlocked with no active connections, project “cdd\_db\_pk” is open by user “new\_user” in this instance of the CCDD application, project “test2” is not open by CCDD but does have an active connection from another application by user “rmcclune”, and project “test\_project” is open by user “new\_user” in another instance of CCDD.

After selecting a project (or projects) to unlock, selecting the **Unlock** button unlocks the project database(s). Selecting the **Cancel** button exits the dialog without altering the project lock statuses.

**Warning:** Removing a project's lock allows concurrent access to the project from more than one instance of the CCDD application. This may produce unexpected results or corruption of the project database if multiple instances update the database.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 47 of 166 |

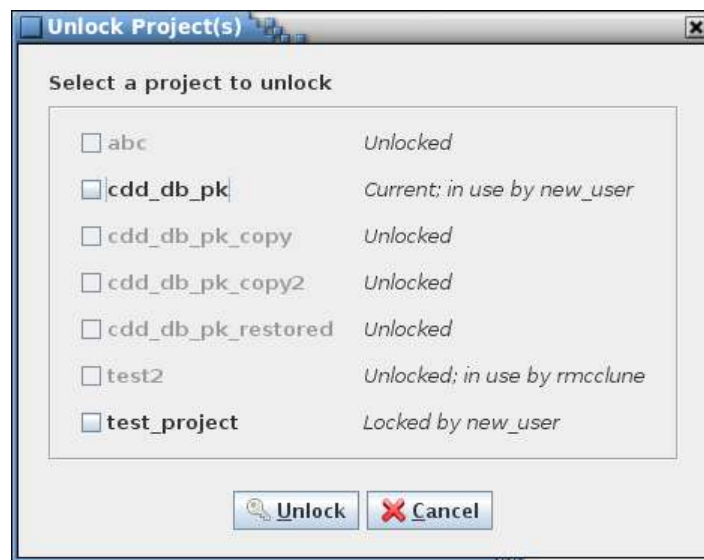


Figure 22. Unlock Project(s) dialog

#### 4.10.2.10 Verify

The **Verify** command performs a consistency check on the currently open project database. This check ensures that the project's data tables are consistent with the table type definitions and that the information within a table is valid. Errors in the tables should not arise from interactions with the CCDD application. However, changes to the project's database from another application (e.g., psql) or using a version of the CCDD application that differs from the one used to create the project could result in the introduction and flagging of errors. The user is alerted to any potential problems and, where possible, is given the option to make corrections to the project's tables, ignore the problem and continue the check, or to cancel the check. There are three areas of verification performed, described in the following paragraphs. No changes are made to the project database until the user selects and confirms applying the updates at the end of the check. Since the project database can be altered by the verification it is recommended that the project be backed up or copied prior to allowing any updates to the database.

The *internal table check* verifies the project database's internal tables. These tables are for use by the CCDD application and are not directly viewable or editable by the user from within the application. The verification checks that the tables contain the expected number of columns and that the columns have the expected names and data types. Extraneous internal tables – tables with names conforming to the internal table naming scheme – are also detected.

The *table type check* compares each data table to its table type definition, verifying that the number of columns, column names, and column order match.

Each data table is checked to see if its type matches one of the defined table types, and if so, that it contains only those columns defined for that type. If updating is confirmed then missing columns are added (devoid of data) and extra columns are eliminated (including the data in them). If the table's type is not defined then the entire table is deleted, along with its contents, when the update is applied. If the internal data type of the column doesn't match the expected one then the column data type is corrected with no loss of the column's data.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 48 of 166 |

The *table data check* performs a check of the data within each table. In doing so it opens and inspects each data table and can generate a considerable number of database queries as is evidenced in the event log. Depending on the number of tables and the amount of data within them this operation can take a while. Each data value is checked to ensure it isn't null (empty cells in a data table contain blanks instead of nulls) and that it is compatible with the input type as defined in the table's type definition (e.g., no alphabetical characters in an integer cell). A check is made that each row in the table contains a row index (which are hidden from display in the table editor), that the row indices begin at 1, and there are no gaps in the index values. For structure tables containing array variables the check looks for missing array definitions (i.e., and array member without a corresponding array definition) and members (e.g., an array with an array size of 3 having only two members). Any columns in the table marked as unique (via the table type manager) are checked for duplicate values. If a duplicate is found and updating is confirmed then the value is replaced with a blank.

Note that certain inconsistencies may prevent a complete check of a project. For example, if a column is missing from a data table then the table's data can't be loaded and checked until the missing column is added (an error dialog is displayed indicating the table's data can't be loaded) at the end of the verification check. For this case the column should be allowed to be added during the first verification check, then a second verification performed so that the data within the affected table is checked. Inconsistencies ignored during the table data verification section may lead to subsequent inconsistency detection that otherwise wouldn't exist. An example would be ignoring a missing array definition when multiple array members are present – an issue is raised for each array member if the missing definition is ignored, whereas if update is selected the subsequent missing definition warnings won't occur.

When the verification steps are complete, if any issues are detected then a dialog appears detailing the issues and the corrective action to be taken (Figure 23). After selecting the check box(es) in the **Action** column (or using the **Select all** check box to toggle selection of all of the issues), selecting **Okay** applies the corrective action(s) to the project database to address the issues flagged to be updated. Selecting **Print** allows outputting the list to a printer. Selecting **Cancel** exits the verification check without making any changes.



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 49 of 166 |

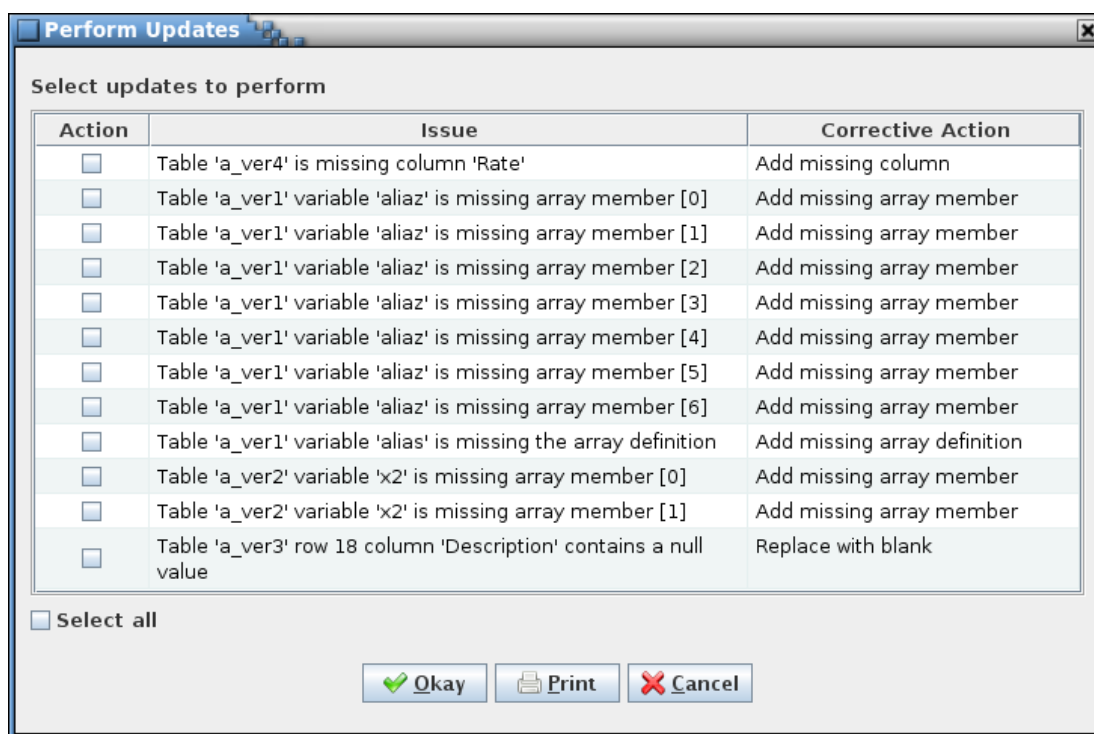


Figure 23. Example Perform Updates dialog

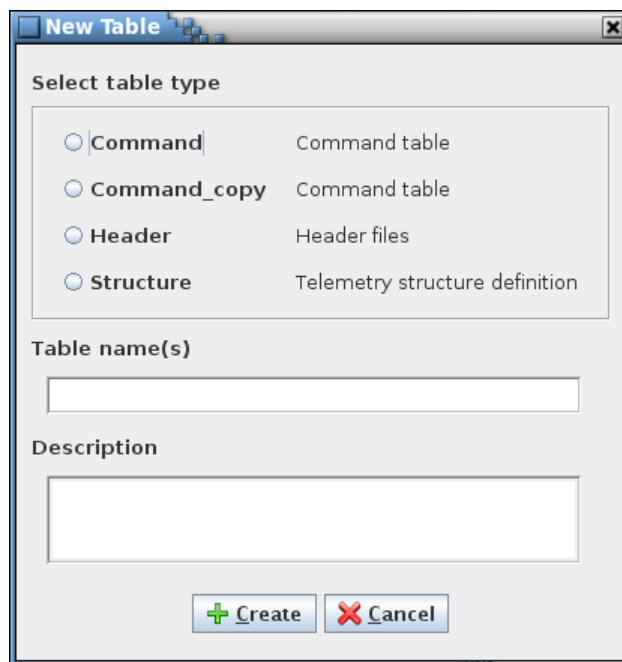
### 4.10.3 Data menu

The **Data** menu has the commands for manipulating the data tables that contain a project's data.

#### 4.10.3.1 New table(s)

The **New Table** command allows creation of a new data table. The New Table dialog (Figure 24) displays the available defined table types and an input field for the table name and its description.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 50 of 166 |



The image shows a 'New Table' dialog box with a title bar containing a blue icon and the text 'New Table'. The dialog has a 'Select table type' section with four radio buttons: 'Command' (selected), 'Command\_copy', 'Header', and 'Structure'. To the right of each radio button is a label: 'Command table', 'Command table', 'Header files', and 'Telemetry structure definition'. Below this is a 'Table name(s)' text field and a 'Description' text area. At the bottom are two buttons: '+ Create' and 'X Cancel'.

Figure 24. New Table dialog

A table type must be selected along with a valid table name. The description is optional and can be added or altered later using the table editor. Table names must be unique within a project. Though upper and lower case characters may be used, the name must still be unique if all of the characters are forced to lower case. The name must begin with a character or underscore ( `_` ) and can only contain characters, numerals, and underscores. Name length is constrained by PostgreSQL to a maximum of 63 characters. Also, the name may not match a primitive data type (e.g., double, or int8), a PostgreSQL reserved word, or one of the internal table names created by the CCDD application (these begin with double underscores). A warning dialog appears if any constraint is violated.

Multiple tables of the same type may be created at the same time by entering more than one name in the table name field with each name separated by a comma. The new tables created in this manner share the description entered in the description field (if any). The descriptions can be added or altered later using the table editor.

Selecting **Create** causes the table(s) to be created and stored in the database. Each table created has the columns defined by the selected table type and initially has no rows. If the type chosen has default data fields, then the new table inherits these fields and their default values. The new table(s) can then be opened using the **Edit** command (see paragraph 4.10.3.2).

#### 4.10.3.2 Edit table(s)

The Edit command displays the data table selection dialog (Figure 25).

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 51 of 166 |

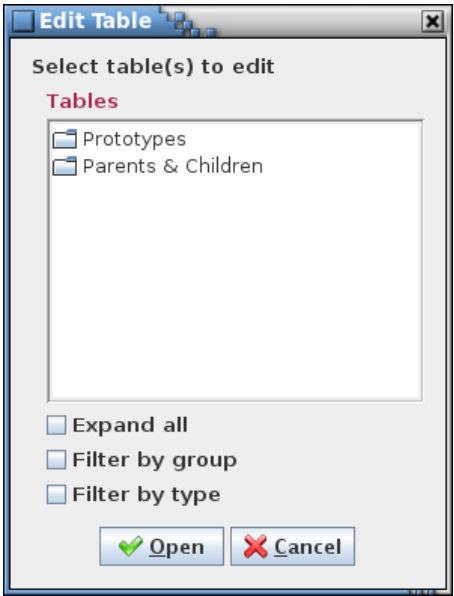


Figure 25. Select Table dialog

The selection dialog has a table tree (see paragraph 4.6.3) from which one or more tables are selected for editing. Pressing the **Open** button opens the selected table(s) in a table editor (see Figure 26 for an example). Positioning the mouse pointer over a table name in the tree and double right-clicking can also be used to open the selected table. The **Cancel** button closes the table selection dialog without opening a table.

Menu bar  
Table tabs

Table data

Description  
Data fields

Buttons

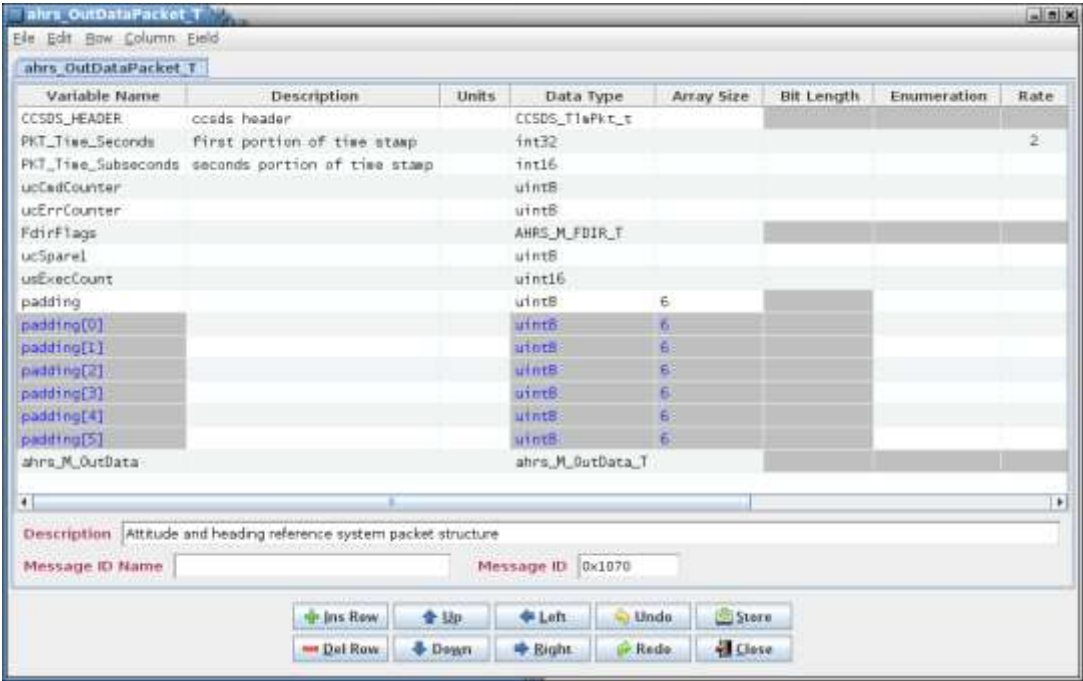


Figure 26. Example table editor

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 52 of 166 |

The table editor provides the means by which data is added to, altered, or removed from a data table. The editor is divided into four main sections.

**Menu bar** The first is the menu bar, which contains the commands, described in the following paragraphs, for manipulating the table contents.

**Table tabs** The second section has one or more tabbed panes, each representing a data table's contents. The tab names indicate the table to which the tab applies. A prototype or parent table shows only the prototype/parent name. For a structure table's child table tab displays a name in the format *parent : structure.variable* where *parent* is this table's parent table name, *structure* is the name of the prototype structure represented by this table, and *variable* is the variable name that references this child table in the child's immediate parent structure. An asterisk beside the table name in the tab indicates that a change has been made to the table that hasn't been stored in the project database. Hovering the mouse pointer over the tab name produces a tool tip showing the table's type, top-level parent, and complete structure and variable path.

**Table data** The columns displayed in the tabbed pane's table are determined by the table type chosen for the table being edited. The table columns can be sorted by selecting the column header with the mouse pointer and pressing the left mouse button. The rows are first sorted in ascending order, depending on the selected column's contents. Selecting the column again sorts in descending order, and a third selection restores the rows to their original order. The column order may be changed by positioning the mouse pointer over a column header, pressing and holding the left mouse button, then dragging the column to the new location (see paragraph 4.10.3.2.4 for the menu commands for repositioning the columns). If the column order change is stored in the database then it is restored when the table is reopened. Column ordering is preserved separately for each user.

**Description** The third section contains the table description. The description is initially empty. The text entered here is used as a tool tip when the mouse pointer hovers over the table name in the table tree. Letter, numeral, and punctuation characters may be entered. Additionally, HTML tags can be inserted to provide additional formatting to the tool tip text.

**Data fields** The next section displays any data fields assigned to the table. See paragraph 4.7 for details concerning data field creation.

**Buttons** The remaining section has a series of buttons that perform some of the more commonly used commands. Certain buttons may be disabled depending on the table displayed in the editor. The buttons are as follows:

**Ins Row** Inserts a new row in the table. See paragraph 4.10.3.2.3.1.

**Del Row** Deletes the selected row(s) from the table. See paragraph 4.10.3.2.3.1.

**Up** Moves the selected row(s) up one row. See paragraph 4.10.3.2.3.2.

**Down** Moves the selected row(s) down one row. See paragraph 4.10.3.2.3.3.

**Left** Moves the selected column(s) left one column. See paragraph 4.10.3.2.4.1

**Right** Moves the selected column(s) right one column. See paragraph 4.10.3.2.4.1.

**Undo** Undoes the last action performed (typing, paste, insert, delete, redo, etc.).

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 53 of 166 |

- Redo** Reverses the last action undone (typing, paste, insert, delete, redo, etc.).
- Store** Stores the currently displayed table's contents (cell data, description, and data fields) in the database. See paragraph 4.10.3.2.1.2.
- Close** Closes the currently displayed table's editor. See paragraph 4.10.3.2.1.7.

The following paragraphs provide details on the commands available in the table editor menu bar.

#### 4.10.3.2.1 File menu

##### 4.10.3.2.1.1 Edit table(s)

The **Edit table(s)** command displays the Edit Table dialog (Figure 25). The table(s) opened from this dialog appear in the current table editor under their own tabs.

##### 4.10.3.2.1.2 Edit prototype

If the currently displayed table in the editor is a child table then issuing the **Edit prototype** command opens the child's prototype table in the editor under its own tab.

##### 4.10.3.2.1.3 Store current

The **Store current** command stores the currently displayed table's contents, including the table's cell data, description, data fields, row order, or column order, into the database if changes have been made since the table was opened or since the last store operation. If no changes have been made then no action is taken; otherwise a confirmation dialog appears allowing the user to choose between continuing with the store operation or canceling it.

##### 4.10.3.2.1.4 Store all

Selecting the **Store all** command is similar to the **Store current** command (paragraph 4.10.3.2.1.3) except that all tables in the table editor are stored to the database if changes have been made. A confirmation dialog appears allowing the user to choose between continuing with the store operation or canceling it.

##### 4.10.3.2.1.5 Import data

The **Import data** command provides a means of inserting data from a CSV, XTCE XML, or EDS XML formatted file into an existing table. The file may contain the table name and type, table data, table description, and/or data field values for one or more tables. A dialog appears allowing the user to choose the import file. Only files with the extension .csv are displayed by default. The **Export .csv** command produces a file compatible with the import command; see paragraph 4.10.3.2.1.6 for details on the file format. Rows are inserted into the table at the currently selected row (or at the end of the table if no cell is selected) to contain the imported data.

Empty rows in the import file - including those containing only commas - are ignored. If the file contains table cell data, then the first non-empty row in the file must contain the column names. The names are compared to those in the table in order to insert the values into the proper column. If the number of columns in the file differs from the number in the table, or if a column in the file doesn't exist in the table then a dialog appears alerting the user to the discrepancy. The user can elect to continue with the import or cancel it at this point. Only data field names matching those in the table have their values imported; any non-matching fields in the file are ignored. Values in the import file may be contained within double quotes to allow for quotes and commas within the values.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 54 of 166 |

#### 4.10.3.2.1.6 Export table

The **Export table** command provides a means of outputting the current table's name, type, data, description, and data field values to a file in CSV, XTCE XML, or EDS XML format. A dialog appears allowing the user to choose the location and name of the export file. If the file already exists then another dialog asks for confirmation to overwrite the file. The exported file may be imported into an existing table using the **Import data** command (paragraph 4.10.3.2.1.5).

The export file is divided into four sections: table name & type, column data, table description, and data field values. Each section is designated by the following text strings, respectively, each on its own row: `_name_type_`, `_column_data_`, `_description_`, and `_data_fields_`. On subsequent rows following a section designator are the data pertinent to that section.

The data for the name & type section is in the form `"table name", "table type"`. This information is written to the export file only to provide information on where the data originated; it is not used when importing the file. The first row in the column data section contains the column names; the remaining rows in this section are the values in each row of the table for these columns. All cells, including empty ones, are output to the file; however, entirely empty rows are ignored. The description section contains the table description. The data field section has a separate row for each of the table's data fields. Each row shows the field in the format `"field name", "field value"`.

The table name, type, cell values, description, and field names and values are enclosed in double quotes to preserve quotes and commas in the values.

#### 4.10.3.2.1.7 Print current

The **Print current** command outputs the currently displayed table to a printer or file selected by the user from the printer dialog that appears. The table's data fields are also output on a separate page.

#### 4.10.3.2.1.8 Close current

The **Close current** command closes the currently displayed table's editor tab. If this is the last table in the editor then the editor window is also closed. If any changes to the table's data, description, data fields, row order, or column order have been made, but not stored in the database, then a confirmation dialog appears allowing the user to choose between continuing with the close operation, discarding the changes, or canceling it, keeping the table editor open.

#### 4.10.3.2.1.9 Close all

The **Close all** command performs a similar operation to the **Close current** command (paragraph 4.10.3.2.1.8) except all tables in the editor are closed as well as the editor. For unstored changes in any of the tables, only a single confirmation dialog appears; if confirmed, the changes in all tables in the editor are discarded.

### 4.10.3.2.2 Edit menu

#### 4.10.3.2.2.1 Copy

The **Copy** command places the contents of the highlighted cell(s) into the clipboard. This information can then be pasted into another row in this table's editor, another table's editor, into a table type editor, or into applications other than CCDD. The Ctrl+C keys perform the same operation.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 55 of 166 |

#### 4.10.3.2.2.2 Paste

The **Paste** command places the contents of the clipboard into the editor. The paste location is determined by the leftmost and uppermost highlighted cell. If the table contains collapsed arrays then the arrays are expanded prior to pasting the data. The rows and columns of the copied cells are the placed into the editor beginning at this location and extending down and to the right and overwrites the existing data in the cells. If insufficient columns exist for the pasted data then the excess column(s) is truncated. Extra rows are inserted at the bottom of the table to provide room for data that would be placed below the editor's last row. See paragraph 4.10.3.2.2.3 on inserting copied data without overwriting the existing cell contents. The Ctrl+V keys produce the same operation.

#### 4.10.3.2.2.3 Insert

The **Insert** command behaves similarly to the **Paste** command (paragraph 4.10.3.2.2.2) except that no editor data is overwritten. Instead, rows are inserted, beginning at the row below the upper- and leftmost highlighted cell, to accommodate the pasted values. The Ctrl+I keys produce the same operation.

#### 4.10.3.2.2.4 Undo

The **Undo** command performs the same action as the **Undo** button. The command undoes the last action performed (typing, paste, insert, delete, redo, etc.) in the currently displayed type editor. The Ctrl-Z keys produce the same operation.

#### 4.10.3.2.2.5 Redo

The **Redo** command performs the same action as the **Redo** button. The command reverses the last action undone (typing, paste, insert, delete, redo, etc.) in the currently displayed type editor. The Ctrl+Y keys produce the same operation.

#### 4.10.3.2.2.6 Clear data

The **Clear data** command erases the contents of the selected cells in the currently visible table.

### 4.10.3.2.3 Row menu

#### 4.10.3.2.3.1 Insert row

The **Insert row** command performs the same action as the **Ins Row** button. The command causes an empty row to be inserted below the currently selected cell's row. If no cell is selected then the new row is inserted at the end of the table. The Insert key produces the same operation.

#### 4.10.3.2.3.2 Delete row

The **Delete row** command performs the same action as the **Del Row** button. This command deletes the row associated with each currently selected cell. If no row is selected then this command has no effect. The Delete key produces the same operation.

#### 4.10.3.2.3.3 Move up

Issuing the **Move up** command moves the row(s) of the selected cell(s) up one row relative to the remaining rows. Only prototype tables may have their rows reordered; reordering the prototype's rows affects the row ordering for all tables based on the prototype. Reordering the rows is recognized as a table change and is preserved in the database via use of the store command (menu or button).

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 56 of 166 |

#### 4.10.3.2.3.4 Move down

Issuing the **Move down** command moves the row(s) of the selected cell(s) down one row relative to the remaining rows. Only prototype tables may have their rows reordered; reordering the prototype's rows affects the row ordering for all tables based on the prototype. Reordering the rows is recognized as a table change and is preserved in the database via use of the store command (menu or button).

#### 4.10.3.2.3.5 Expand arrays

Selecting the **Expand arrays** command toggles display of array members in those tables containing an "Array Size" column. When enabled, each array member is displayed in its own row in the table beneath the array's definition row. When disabled, the array members are hidden, though the array's definition row continues to be displayed. Array member visibility can also be toggled by positioning the mouse pointer over any cell in the "Array Size" column (except the column name row) and double right-clicking.

#### 4.10.3.2.3.6 Array overwrite

The **Array overwrite** command is a submenu with three mutually exclusive selections: Overwrite all, Overwrite empty, and Overwrite none. The selection governs pasting of data into array member cells already containing values. Overwrite all, the default, overwrites any existing values with the pasted values. Overwrite empty only pastes values into cells that are currently empty; paste values are discarded if the target cell is occupied. Overwrite none prevents pasting values into array member cells.

### 4.10.3.2.4 Column menu

#### 4.10.3.2.4.1 Move left

Issuing the **Move left** command moves the currently column(s) of the selected cell(s) to the left one column relative to the remaining columns. Reordering the columns is recognized as a table change and is preserved in the database separately for each user via use of the store command (menu or button).

#### 4.10.3.2.4.2 Move right

Issuing the **Move right** command moves the currently column(s) of the selected cell(s) to the left one column relative to the remaining columns. Reordering the columns is recognized as a table change and is preserved in the database separately for each user via use of the store command (menu or button).

### 4.10.3.2.5 Field menu

#### 4.10.3.2.5.1 Manage fields

The **Manage fields** command allows the user to create, alter, and delete default data fields for the table represented by the active table editor tab. See paragraph 4.7 for information regarding data fields and use of the data field editor.

The fields manipulated by the field editor are displayed below the table editor table and description when the **Update** button is pressed. After the field editor is closed values can be entered into the data fields. The table editor's **Store** button or command must be used to store the changes in the project database and apply them to the tables.

#### 4.10.3.2.5.2 Clear values

The **Clear values** command clears the contents of all of the currently displayed editor's data fields. A confirmation dialog is first displayed. Selecting **Okay** causes all of the data field values to be blanked. Selecting **Cancel** exits the dialog without affected the data field values.



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 57 of 166 |

#### 4.10.3.3 Rename table

The **Rename Table** command allows a prototype data table to be renamed. Child tables cannot be renamed using this dialog. Child tables are instances of a prototype table assigned as a variable, so a child table's name is a combination of its prototype table name and the variable name in its parent table's prototype. Therefore, child table names can only be altered by changing the name of the child table's prototype table, or changing the name of the variable representing the child table in its parent table's prototype table.

The Rename Table dialog (Figure 27) appears, which displays a table tree showing the prototype tables, and input fields for providing the selected table's new name and description. See paragraph 4.6.3 for details on the table tree. A table is first selected from the tree; the table name and description (if any) appear in the input fields. After altering the name and description fields as desired the **Rename** button is selected to change the table's name and description. See paragraph 4.10.3.1 for details on table name constraints. The new name and description is immediately reflected in all parent and child tables, including those appearing in open table editors. The description is optional and can be added or altered later using the table editor. Select the **Cancel** button to exit the dialog without making changes to the table names.

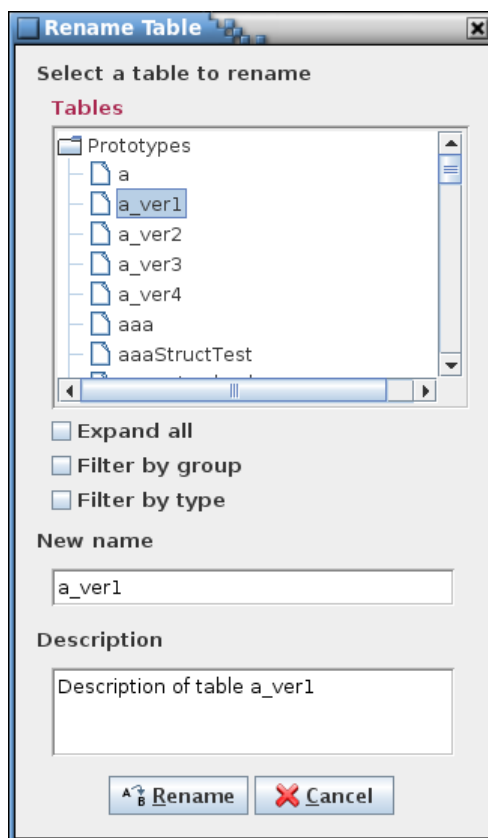


Figure 27. Rename Table dialog

#### 4.10.3.4 Copy table

The **Copy Table** command allows a prototype data table to be copied. The Copy Table dialog (Figure 28) appears, which displays a table tree showing the prototype tables, and input fields for providing the

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-####  | Baseline       |
|   | Date: January 2017   | Page 58 of 166 |

name and description of the selected table's copy. See paragraph 4.6.3 for details on the table tree. A table is first selected from the tree; the table name appears in the input field with “\_copy” appended and its description, if any, appears in the description field. After altering the name and description fields as desired the **Copy** button is selected to create the table's copy. See paragraph 4.10.3.1 for details on table name constraints. The description is optional and can be added or altered later using the table editor. Select the **Cancel** button to exit the dialog without making changes to the table names.

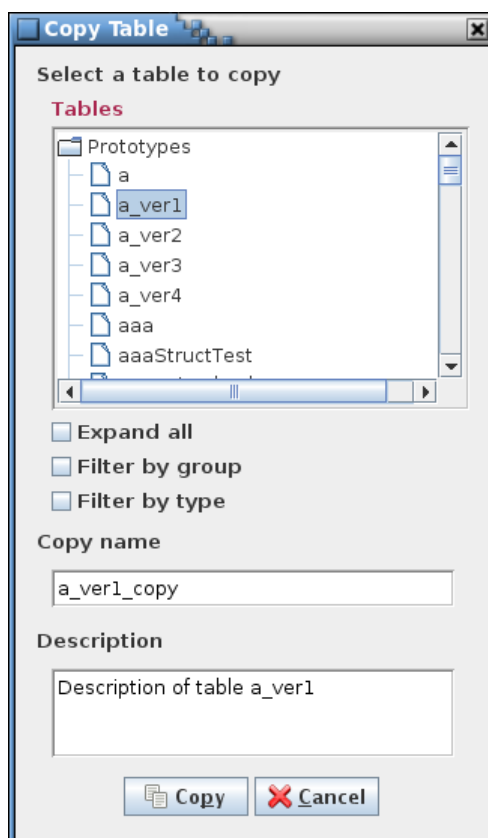


Figure 28. Copy Table dialog

#### 4.10.3.5 Delete table(s)

The **Delete Table** command allows one or more prototype tables to be deleted. The Delete Table dialog (Figure 29) appears, which displays a table tree showing the prototype tables. See paragraph 4.6.3 for details on the table tree. After a table is selected from the tree the **Delete** button is selected to delete the table. *All instances of the deleted table, both parent and child tables, are immediately deleted, including those appearing in open table editors.* Select the **Cancel** button to exit the dialog without deleting a table.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 59 of 166 |

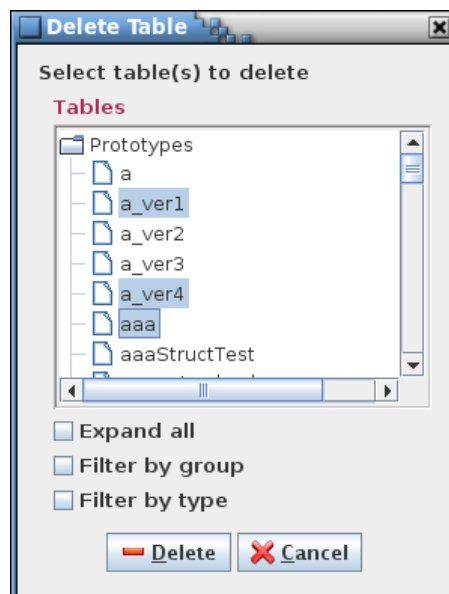


Figure 29. Delete Table dialog

#### 4.10.3.6 Import table(s)

The **Import table(s)** command allows loading the table name, type, data, description, and data field values of one or more tables, from one or more CSV or XML (EDS or XTCE) formatted files. A dialog appears allowing the user to choose the location of the import file(s) (see Figure 30).

A check box allows the user to choose whether to replace existing tables with the same name with the import file data. If selected the check box is enabled that indicates if replaced tables that have data fields have the existing fields appended to those imported. If this check box is selected then the check box is enabled that determines whether to use the existing data field or the imported one in the event that the data fields have the same name.

The remaining check box determines if a backup of the project database is created prior to continuing with the import operation.

The format of the import file is described in paragraph 4.10.3.2.1.6.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-####  | Baseline       |
|   | Date: January 2017   | Page 60 of 166 |

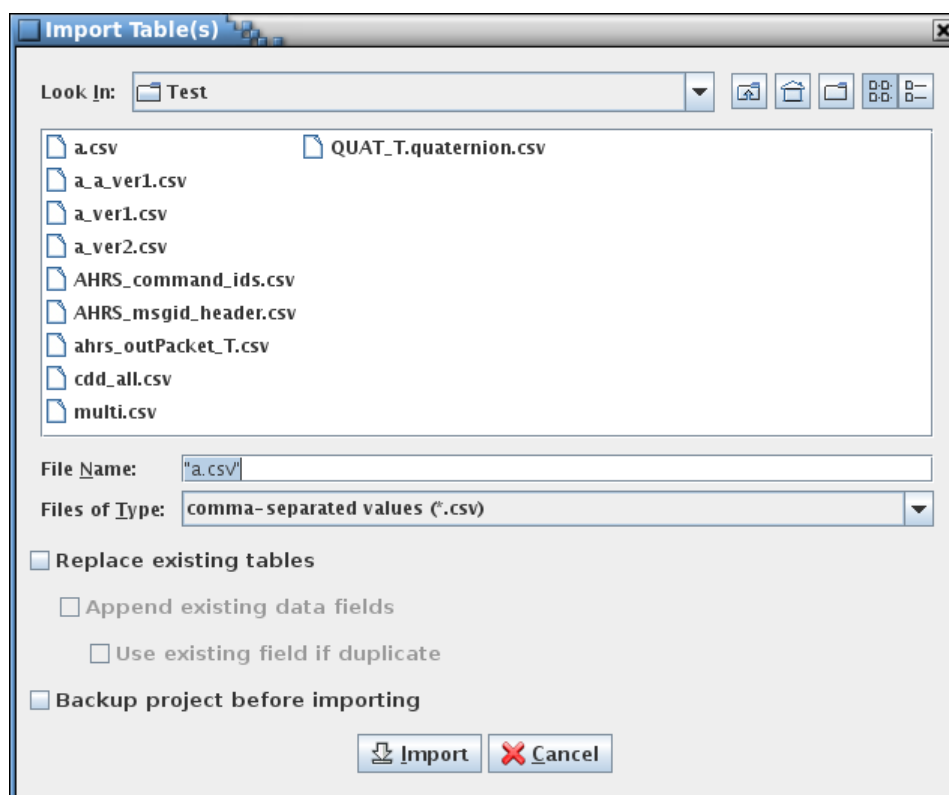


Figure 30. Import table(s) dialog

#### 4.10.3.7 Export table(s)

The **Export table(s)** commands allow outputting the table name, type, data, description, and data field values of one or more tables, either combined into a single file or with each table in its own file, in CSV or XML formats.

##### 4.10.3.7.1 CSV

The **Export table(s) - CSV** command allows outputting one or more selected tables in CSV format. A dialog appear allowing the user to choose the location of the export file(s) and to choose which tables to export (see Figure 31). If the tables are combined into one file then the output file name must be entered in the export file field or selected from the file chooser dialog that appears is the **Select** button is pressed. For multiple tables output to individual files the file names are automatically assigned based on the table names. If an output file already exists then another dialog asks for confirmation to overwrite the file. The exported file may be imported into a table using the **Import .csv** command (paragraph 4.10.3.2.1.5) or into the project using the **Import table(s)** command (paragraph 4.10.3.6). The format of the export file is described in paragraph 4.10.3.2.1.6.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 61 of 166 |

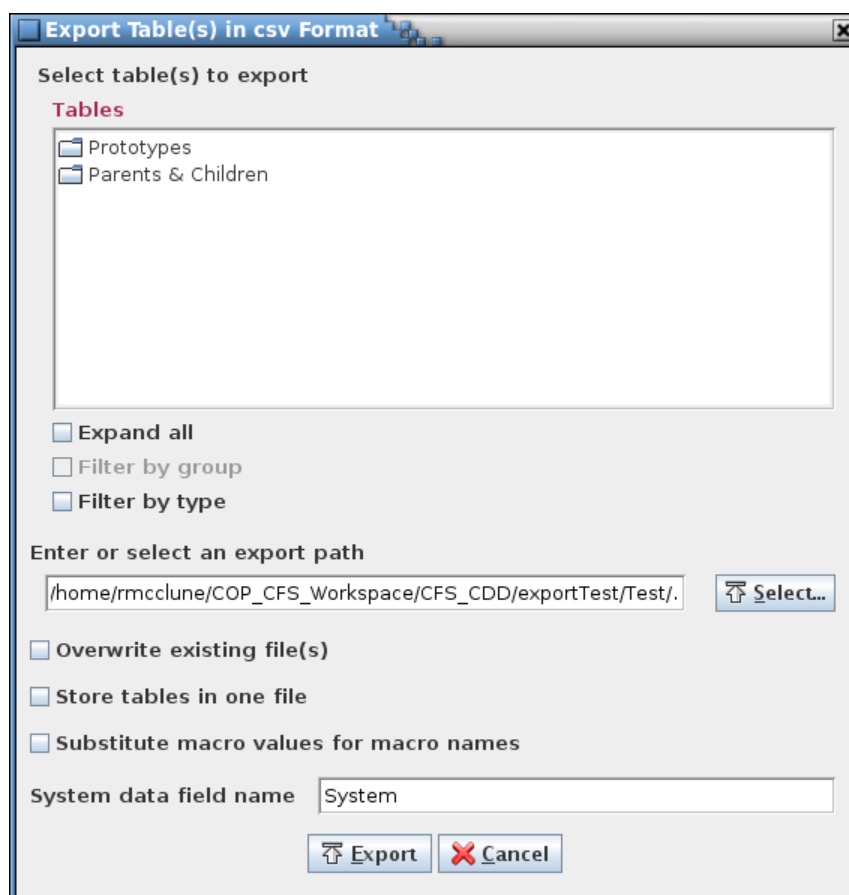


Figure 31. CSV export dialog

#### 4.10.3.7.2 EDS

The **EDS** command allows conversion of the selected table(s) into electronic data sheet (EDS) XML format. When the command is executed the Export EDS dialog appears (Figure 32). The file to which the project is exported is entered in the export file name field; alternately, an existing file may be chosen by pressing the **Select...** button, which produces a file selection dialog. Pressing the **Export** button converts the currently open project into EDS XML format and stores it into the selected file. Pressing the **Cancel** button closes the dialog without exporting the project. See paragraph 4.6.4 for information regarding the conversion of enumerated values into EDS XML format.

EDS conversion assumptions...

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 62 of 166 |

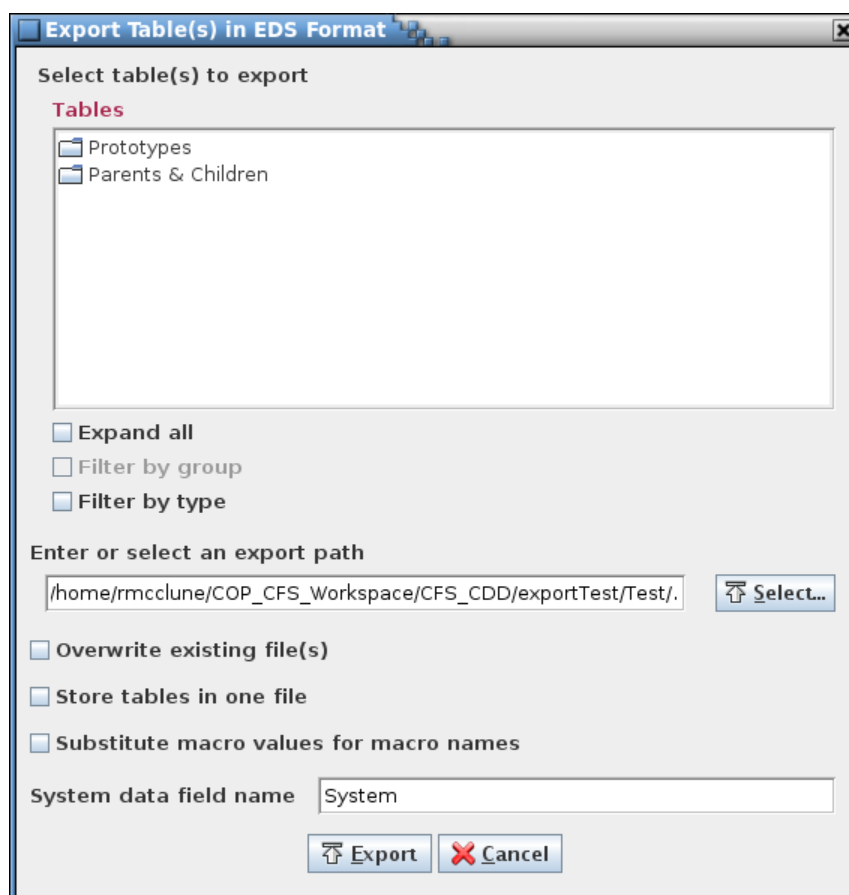


Figure 32. EDS export dialog

#### 4.10.3.7.3 XTCE

The **Export table(s) - XTCE** command allows conversion of the selected table(s) into XML telemetric & command exchange (XTCE) format. When the command is executed the XTCE export dialog appears (Figure 33). The file to which the project is exported is entered in the export file name field; alternately, an existing file may be chosen by pressing the **Select...** button, which produces a file selection dialog. Certain attributes are set in the remaining input fields. The meaning of these attributes is user-defined, but none of the fields may be empty when the project is exported. Pressing the **Export** button converts the currently open project into XTCE format and stores it into the selected file. Pressing the **Cancel** button closes the dialog without exporting the project. See paragraph 4.6.4 for information regarding the conversion of enumerated values into XTCE format.

XTCE conversion assumptions... Need a data field labeled "System" in order to group related tlm and cmds; column names... Also, "units" don't really apply to array definitions or to structure variables – only the end item variable (primitive) units are used

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 63 of 166 |

**Export Table(s) in XTCE Format**

Select table(s) to export

**Tables**

- Prototypes
- Parents & Children

☐ Expand all  
☐ Filter by group  
☐ Filter by type

Enter or select an export path

/home/rmclune/COP\_CFS\_Workspace/CFS\_CDD/exportTest/Test/. Select...

☐ Overwrite existing file(s)  
☐ Store tables in one file  
☐ Substitute macro values for macro names

System data field name

**XTCE Attributes**

Version   
Validation Status

**Classification**

First level   
Second level   
Third level

Export Cancel

Figure 33. XTCE export dialog

#### 4.10.3.8 Manage groups

The **Manage groups** command allows data tables to be assigned to user-defined groups. These groups can be used to filter the tables in the table trees used in other dialogs, making it easier to locate tables that are related (e.g., by a vehicle subsystem or CFS application). Groups can be added, altered, or deleted. When the command is selected a dialog similar to that in Figure 34 appears.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 64 of 166 |

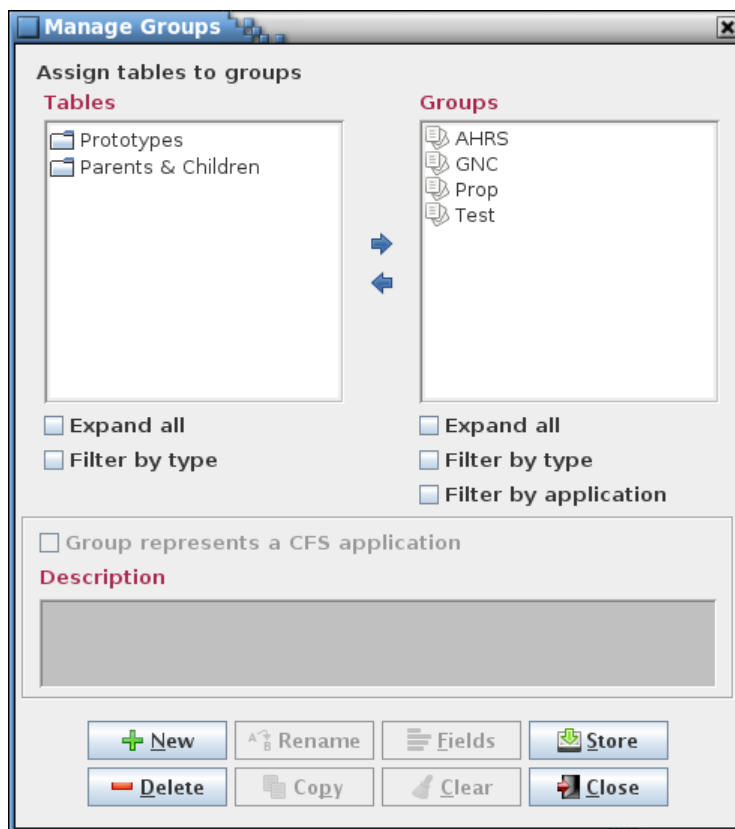


Figure 34. Manage Groups dialog

The upper left of the dialog contains a table tree (under the heading “Tables”). The upper right shows the groups and their trees (under the heading “Groups”). Both trees have the “Expand all” and “Filter by types” check boxes (see paragraph 4.6.3 for further details). In between these are arrows for moving tables in and out of the group(s). Below the trees is a check box for indicating that the group represents a CFS application and an input field for adding a description for the group. This description is used as text for a tool tip that appears is the table trees whenever the mouse pointer hovers over a group name.

To create a group select the **New** button and provide a group name and description in the input dialog that appears (Figure 35). The group name may not be blank, nor is the name allowed to match that of an existing group. The group name may contain alphanumeric, spaces, and punctuation characters; there is no length constraint. If the check box, “Group represents a CFS application” is selected then the group is automatically assigned a number of data fields appropriate for an application (the fields may be altered using the **Fields** button after the group is created; a group’s classification as a CFS application can be altered later – see below). These fields are Schedule Rate, Execution Time, Execution Priority, Message Rate, Wake-Up Name, Wake-Up ID, HK Send Rate, HK Wake-Up Name, HK Wake-Up ID, and SCH Group. The contents of these fields is used when population the scheduler table created with the application scheduler (paragraph 4.10.4.3). If **Okay** is selected the new group’s name appears in the group tree.



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 65 of 166 |



Figure 35. New Group dialog

To add tables to a group select the group in the Groups tree using the mouse or keyboard. Then, in the Tables tree, expand the tree as needed and select one or more tables using the mouse or keyboard. Multiple tables can be selected simultaneously by holding the Ctrl or Shift keys down when making a selection. Selecting a table automatically includes its child tables (and their children, etc.). Choosing a child table automatically includes its parent table, and its parent's parent, etc., up to the top-level of the tree, but does not include any of its siblings (i.e., tables having the same parent and at the same tree level as the chosen table). Finally, select the right arrow button in the center of the dialog. The table(s) chosen appear in the selected group, and the group's tree is expanded to show the table(s) added. Note that the table hierarchy is preserved in the group's tree. More tables can be assigned to the group as described above.

To remove tables from a group expand the group's tree and select the table(s) to remove using the mouse or keyboard. Then select the left arrow button in the center of the dialog to delete the tables from the group. A table's children (and their children, etc.) are removed along with the chosen table.

To delete a group, first select it in the Groups tree, then select the **Delete** button. Multiple groups can be removed simultaneously if desired by highlighting them while using the Shift or Ctrl keys.

To rename a group, select a single group from the group tree, then press the **Rename** button. An input dialog appears with the name of the selected group in the input field. Alter the name as desired and select **Okay** to change the group's name. The renamed group name may not be blank, nor is the name allowed to match that of an existing group. Select **Cancel** to exit the input dialog without affecting the group's name.

To copy a group and its member tables, select a single group from the group tree, then press the **Copy** button. An input dialog appears with the name of the selected group in the input field with the text "\_copy" appended. Alter the name as desired and select **Okay** to create a copy of the selected group. The group name of the copy may not be blank, nor is the name allowed to match that of an existing group. Select **Cancel** to exit the input dialog without copying the group.

Data fields (see paragraph 4.7) may be assigned to a group. A group must first be selected in the Groups tree; this enables the **Fields** and **Clear** buttons. Select **Fields** to display the data field editor for the currently selected group. The data field values for the currently selected group can be cleared by selecting the **Clear** button.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 66 of 166 |

A group's description can be added or changed by first selecting the group in the Groups tree. The current description for the group appears in the Description input field. The description can then be changed as desired. When a group is selected a check box appears above the description field allowing the group's classification as a CFS application to be changed. If checked a number of default fields are automatically added below the description field, unless these fields are already present. Deselecting the check box does not remove these fields.

Changes made in the group manager (group additions or deletions, table assignments, data field updates, or changes to descriptions) are stored in the database only when the **Store** button is pressed. If changes have been made a confirmation dialog first appears. Select **Okay** to store the updates; select **Cancel** to exit the confirmation dialog without altering the database.

Select the **Close** button to exit the group manager dialog. If there are any unsaved group changes a dialog appears requesting confirmation to discard the changes. Select **Okay** to exit the group manager, losing any unsaved changes. Select **Cancel** to return to the group manager dialog.

#### 4.10.3.9 Manage table types

The **Manage table types** command opens the table type editor (Figure 36). The editor window can be broken down into the following segments: the menu bar, table type tabs, column definitions, type description, and buttons. An additional segment, showing data fields, is visible below the description if data fields are created.

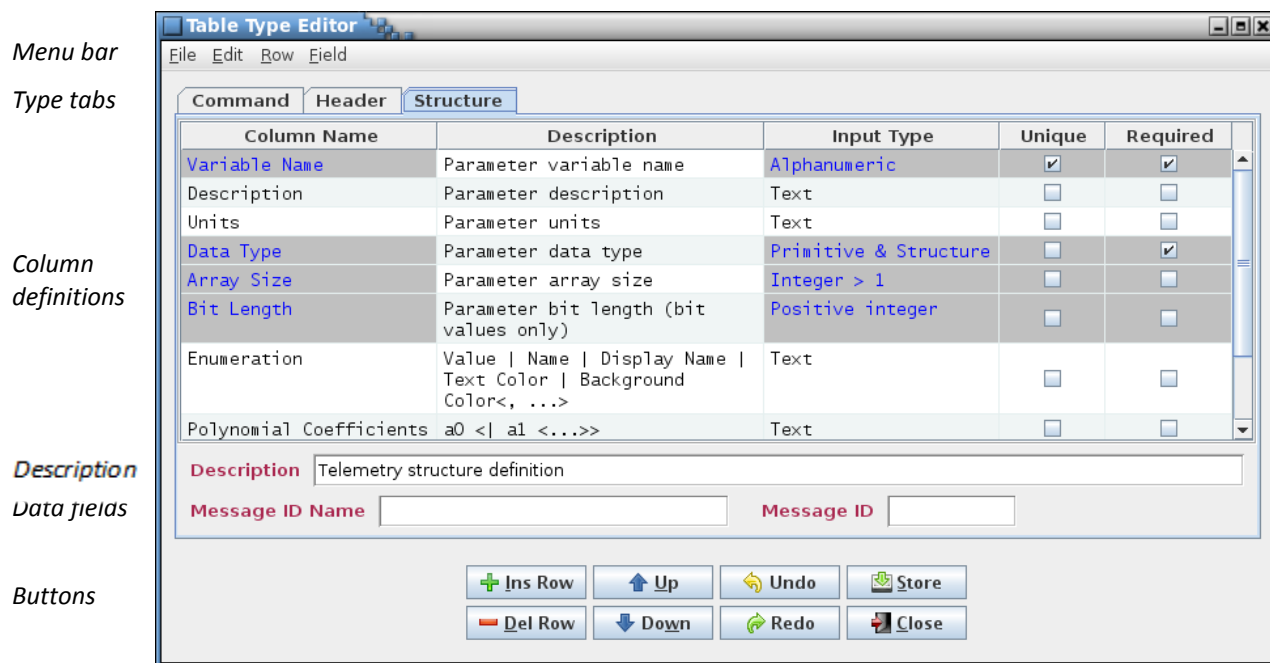


Figure 36. Table type editor

The menu bar has commands associated with the type editor; the commands are described in subsequent paragraphs. The buttons, described in detail below, represent some of the more commonly used commands; each has a counterpart in the menu bar. Each type tab represents one of defined table types. At a minimum this includes the Structure and Command types. Any types created by the user

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 67 of 166 |

also appear. The tabs are arranged in alphabetical order. Selecting the tab causes the editor to display the information for the selected type.

Each row in the editor is a definition of a column that appears in a table of this type when it's created. The order of the column definitions determines the initial column order when the table is first displayed. The editor columns can be sorted by selecting the column header with the mouse pointer and pressing the left mouse button. The rows are first sorted in ascending order, depending on the selected column's contents. Selecting the column again sorts in descending order, and a third selection restores the rows to their original order. Column order can be changed for each individual table in the table editor (see paragraph 4.10.3.2).

Editor cells that are grayed out cannot be changed. These cells represent the information that is necessary to define a table of the type shown; e.g., a Structure table must have at a minimum the variable name, data type, array size, bit length, and enumeration columns. If a copy is made of a table type the cell protection is removed for the copied type.

The editor column descriptions are as follows:

|                    |   |
|--------------------|---|
| <b>Column Name</b> | When a table of this type is displayed, this is the name that's displayed in the table's column header.   |
| <b>Description</b> | This text is displayed as a tool tip whenever the mouse pointer hovers over the table's column header.  |
| <b>Input Type</b>  | The input type constrains the type of value entered into the table cells for the column defined on this row of the editor. If the value entered into the cell doesn't conform to column's specified input type then a warning message dialog is displayed and the cell reverts to its previous value. The input types are selectable from the combo box pull-down menu that appears when a cell in the Input Type column is selected. See paragraph 4.8 for information on the available input types. |
| <b>Unique</b>      | This check box, if selected, indicates that a column's data value must be unique within this column. If a duplicate value is entered into the table's cell then a warning message dialog is displayed and the cell reverts to its previous value.   |
| <b>Required</b>    | This check box, if selected, indicates that a column's data is required. This causes the cell in the table to be highlighted if it is empty. This does not force the user to populate the highlighted cell prior to saving changes to the table, but simply serves as a reminder to the user that the information in this cell is considered important or necessary to one of the scripts.  |

The button commands mirror commands available in the editor menu bar and provide an easy method of accessing the commonly used editor commands. The button commands are described below:

|                |   |
|----------------|---|
| <b>Ins Row</b> | Inserts an empty row below the currently selected cell's row. If no cell is selected then the new row is inserted at the end of the table.  |
| <b>Del Row</b> | Deletes the row associated with each currently selected cell. If no row is selected then this has no effect.  |
| <b>Up</b>      | Move the row(s) of the currently selected cell(s) up one row. This affects the order of the columns of new instances of this table type; it does not affect existing tables of this |

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 68 of 166 |

type. Column order can be changed for each individual table in the table editor (see paragraph 4.10.3.2).

- Down** Move the row(s) of the currently selected cell(s) down one row. This affects the order of the columns of new instances of this table type; it does not affect existing tables of this type. Column order can be changed for each individual table in the table editor (see paragraph 4.10.3.2).
- Undo** Undoes the last action performed (typing, paste, insert, delete, redo, etc.).
- Redo** Reverses the last action undone (typing, paste, insert, delete, undo, etc.).
- Store** Stores the changes made to the currently displayed tab in the type editor (not those in the other tabs) in the database. See paragraph 4.10.3.9.1.5 for further details.
- Close** Closes the type editor window. If any changes for any of the tabs have not been stored then a dialog appears allowing the user to confirm discarding the updates or to cancel closing the editor.

The commands in the editor menu bar are described in the following paragraphs.

#### 4.10.3.9.1 File menu

##### 4.10.3.9.1.1 New type

The **New type** command allows the user to create a new table type. A dialog appears with an input field for entering the new type's name. Select **Create** to create the new type, which is opened in the type editor. The type editor can then be used to populate the type with column definitions, and afterwards new tables of this type may be created and edited. Select **Cancel** to exit the dialog without creating a new table type.

##### 4.10.3.9.1.2 Copy type

The **Copy type** command is used to create a new table type from an existing one, including all of its column definitions, default cell values, and data fields. The active tab in the type editor determines which type is to be copied, so the intended tab must be selected prior to executing the copy command. A dialog appears with an input field for entering the name of the type's copy. The name of the selected type is displayed with "\_copy" appended. After altering the name as desired, select **Copy** to create a copy of the type. Select **Cancel** to exit the dialog without creating a copy.

##### 4.10.3.9.1.3 Rename type

The **Rename type** command is used to rename an existing table type. The active tab in the type editor determines which type is to be renamed, so the intended tab must be selected prior to executing the rename command. A dialog appears with an input field for entering the new name for the table type. The name of the selected type is automatically displayed. After altering the name as desired, select **Rename** to rename the table type. All tables of the renamed type are changed to the new type name. Select **Cancel** to exit the dialog without renaming the table type.

##### 4.10.3.9.1.4 Delete type

The **Delete type** command deletes an existing table type. The active tab in the type editor determines which type is to be deleted, so the intended tab must be selected prior to executing the delete command. A confirmation dialog appears. Selecting **Delete** removes the table type *and all tables of the*

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 69 of 166 |

*deleted type from the database.* Select **Cancel** to exit the dialog without deleting the table type or any tables.

#### 4.10.3.9.1.5 *Store current*

The **Store current** command performs the identical action to the **Store** button. The command stores the changes made to the currently displayed tab in the type editor (not those in the other tabs) in the database. Afterwards, any table created using this table type inherits the columns and data fields of the type.

All existing tables of this type, including those in any open table editors, are updated immediately with the changes. New data fields are added to existing tables; however, deleted data fields are not removed from existing tables. Changes to data field values are applied based on the **Overwrite values** check box described in paragraph 4.10.3.9.4.3.

A confirmation dialog appears allowing the user to choose between continuing with the store operation or canceling it.

#### 4.10.3.9.1.6 *Store all*

The **Store all** command stores the changes made to all of tabs in the type editor in the database. All existing tables of the affected type(s), including those in any open table editors, are updated immediately with the changes. A confirmation dialog appears allowing the user to choose between continuing with the store operation or canceling it.

#### 4.10.3.9.1.7 *Print current*

The **Print current** command prints the contents of the currently displayed tab to be sent to a printer. A dialog first appears allowing the user to select the printer (or file) and adjust the page setup. Selecting **Print** causes the editor contents, including the data fields (if any) are output to the selected printer (or file). Selecting **Cancel** removes the print dialog without printing the table type editor contents.

#### 4.10.3.9.1.8 *Close*

The **Close** command performs the identical action to the **Close** button. The command closes the type editor window. If any changes for any of the tabs have not been stored then a dialog appears allowing the user to confirm discarding the updates or to cancel closing the editor.

### 4.10.3.9.2 **Edit menu**

#### 4.10.3.9.2.1 *Copy*

The **Copy** command places the contents of the highlighted cell(s) into the clipboard. This information can then be pasted into another row in this type's editor, another type's editor, into a data table, or into applications other than CCDD. The Ctrl+C keys perform the same operation.

#### 4.10.3.9.2.2 *Paste*

The **Paste** command places the contents of the clipboard into the editor. The paste location is determined by the leftmost and uppermost highlighted cell. The rows and columns of the copied cells are placed into the editor beginning at this location and extending down and to the right and overwrites the existing data in the cells. If insufficient columns exist for the pasted data then the excess column(s) is truncated. Extra rows are inserted at the bottom of the table to provide room for data that would be placed below the editor's last row. See paragraph 4.10.3.9.2.3 on inserting copied data without overwriting the existing cell contents. The Ctrl+V keys produce the same operation.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-####  | Baseline       |
|   | Date: January 2017   | Page 70 of 166 |

#### 4.10.3.9.2.3 Insert

The **Insert** command behaves similarly to the Paste command (paragraph 4.10.3.9.2.2) except that no editor data is overwritten. Instead, rows are inserted, beginning at the row below the upper- and leftmost highlighted cell, to accommodate the pasted values. The Ctrl+I keys produce the same operation.

#### 4.10.3.9.2.4 Undo

The **Undo** command performs the same action as the **Undo** button. The command undoes the last action performed (typing, paste, insert, delete, redo, etc.) in the currently displayed type editor. The Ctrl-Z keys produce the same operation.

#### 4.10.3.9.2.5 Redo

The **Redo** command performs the same action as the **Redo** button. The command reverses the last action undone (typing, paste, insert, delete, redo, etc.) in the currently displayed type editor. The Ctrl+Y keys produce the same operation.

#### 4.10.3.9.2.6 Clear data

The **Clear data** command empties all of the currently displayed editor's cells. A confirmation dialog is first displayed. Selecting **Okay** causes all of the cells' contents to be deleted. Selecting **Cancel** exits the dialog without affected the cell data.

### 4.10.3.9.3 Row menu

#### 4.10.3.9.3.1 Insert row

The **Insert row** command performs the same action as the **Ins Row** button. The command causes an empty row to be inserted below the currently selected cell's row. If no cell is selected then the new row is inserted at the end of the table. The Insert key produces the same operation.

#### 4.10.3.9.3.2 Delete row

The **Delete row** command performs the same action as the **Del Row** button. This command deletes the row associated with each currently selected cell. If no row is selected then this command has no effect. The Delete key produces the same operation.

#### 4.10.3.9.3.3 Move row(s) up

The **Move row(s) up** command performs the same action as the **Up** button. This command causes the row(s) of the currently selected cell(s) to move up one row. This affects the order of the columns of new instances of this table type; it does not affect existing tables of this type. Column order can be changed for each individual table in the table editor (see paragraphs 4.10.3.2 and 4.10.3.2.4).

#### 4.10.3.9.3.4 Move row(s) down

The **Move row(s) down** command performs the same action as the **Down** button. This causes the row(s) of the currently selected cell(s) to move down one row. This affects the order of the columns of new instances of this table type; it does not affect existing tables of this type. Column order can be changed for each individual table in the table editor (see paragraphs 4.10.3.2 and 4.10.3.2.4).

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 71 of 166 |

#### 4.10.3.9.4 Field menu

##### 4.10.3.9.4.1 Manage fields

The **Manage fields** command allows the user to create, alter, and delete default data fields for the type represented by the active type editor tab. See paragraph 4.7 for information regarding data fields and use of the data field editor.

The fields manipulated by the field editor are displayed below the type editor table and description when the **Update** button is pressed. The type editor's **Store** button or command must be used to store the changes in the database and apply them to the tables. The structure table data field editor allows the user to designate newly assigned fields to only parent or child structure tables, or to all structure tables.

After the field editor is closed values can be entered into the data fields; these become default values for the fields in the tables to which the fields are applied. When the field updates are stored all tables of the affected table type are updated, including those in any open table editors, and tables of this type that are subsequently created have the default fields. If an existing table already has a field of the same name then it is not added; however, the field size, data type, required status, description, or value updated in the table to match the default.

Fields can only be added to tables using this method. If a default field's name is changed then this is considered a new field and is added to the tables of the affected type if the type is stored. If a default field is deleted then there is no effect on the tables when the type is stored.

##### 4.10.3.9.4.2 Clear values

The **Clear values** command clears the contents of all of the currently displayed editor's data fields. A confirmation dialog is first displayed. Selecting **Okay** causes all of the data field values to be blanked. Selecting **Cancel** exits the dialog without affected the data field values.

##### 4.10.3.9.4.3 Overwrite values

The **Overwrite values** command determines how the default field values are applied when the Table Type Editor **Store** button is selected. The default setting is unchecked.

If the **Overwrite values** check box is not selected then the data field value changes are not applied to existing tables of this table type if the tables already contain the affected data field. If an existing table does not already have the data field then the field is added with the default value, regardless of the check box status.

If the **Overwrite values** check box is selected then all existing tables of the updated table type that already contain the data field have the contents of that field replaced with the value in the type editor data field.

#### 4.10.3.10 Manage data types

The Data Type Editor (Figure 37) provides a means of creating, modifying, and deleting primitive data type definitions (see paragraph 4.6.4 for more information on data types). When a project database is first created the primitive data types default to those shown in Table 5.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 72 of 166 |

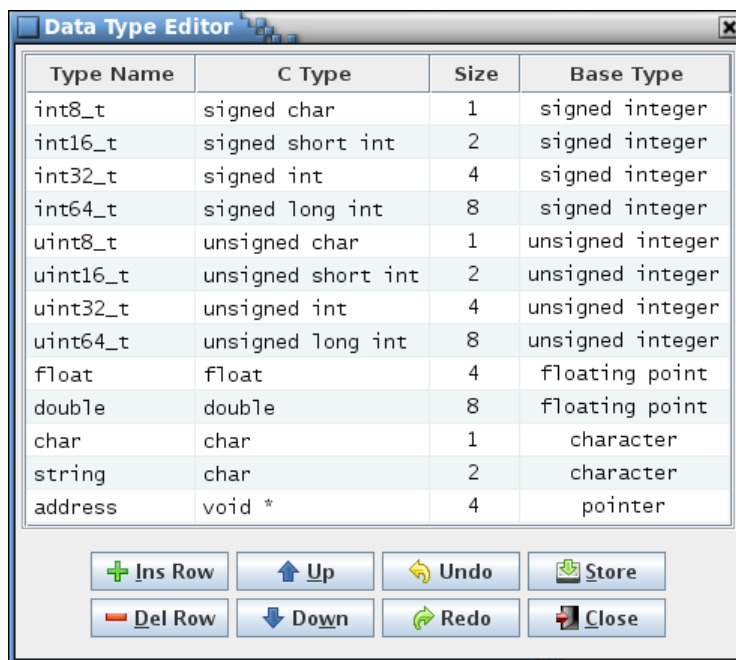


Figure 37. Data Type Editor dialog

The editor column descriptions are as follows:

- Type Name** The type name is the text that represents the data type in a data table cell. Data type names must adhere to C-language naming conventions; i.e., begin with an alphabetic or underscore character, followed by alphabetic, numeric, or underscore characters. The data type names are case insensitive and must be unique. If the type name is left blank then the text in the C Type column determines the data type name displayed in the data table cell.
- C Type** The C type is the C-language equivalent of the data type and may contain spaces. It is available to scripts and web applications (for example, a script can create a header file of `typedef` statements using the type name and C type name combinations). One or more trailing asterisks are allowed if the corresponding base type is 'pointer'. The C type is used as the data type in a data table cell if the Type Name is blank.
- Size** Size, in bytes, occupied by this data type. The size must be an integer greater than 0.
- Base Type** The data type's base type: signed integer, unsigned integer, floating point, character, or pointer. The base type and the size determine how the data type is handled by the application.

Each row in the table is a data type definition. The **Type Name** or **C Type** columns determine the data type displayed in the data type column drop down menus. At least one of these columns must contain text. The type name is used if it isn't blank. If the type name is blank then the C type name is used as the data type name. The **Size** and **Base Type** columns in a definition are required (i.e., it may not be blank).



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 73 of 166 |

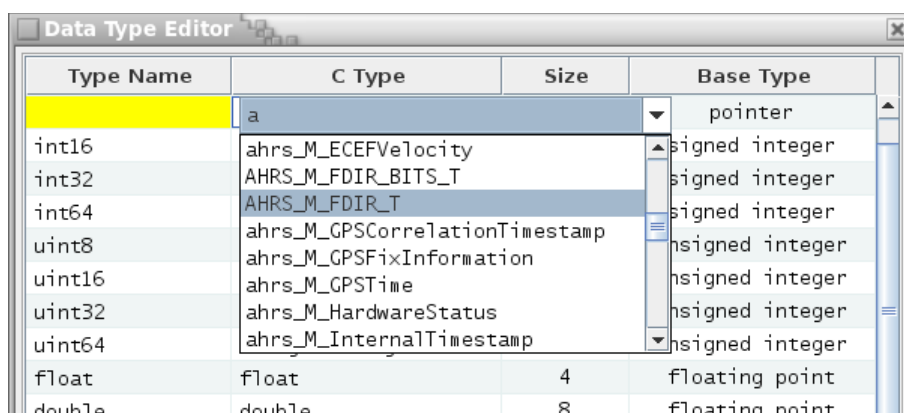
If a pointer base type is selected then an asterisk (\*) is automatically appended to the C type name (unless the cell is empty). Conversely, if the base type is changed from a pointer to something else then any trailing asterisk(s) in the C type name are automatically removed. A pointer to a pointer (or a pointer to a pointer to a pointer, etc.) can be indicated by appending the requisite number of asterisks. Below is an example of creating a pointer to a structure named **myStruct**:



| Type Name    | C Type     | Size | Base Type |
|--------------|------------|------|-----------|
| myStruct_ptr | myStruct * | 4    | pointer   |

Figure 38. Example pointer to a structure data type

As an aid to creating a data type that represents a pointer to a structure a pop-up list of alphabetically arranged prototype structure table names can be displayed from which a structure name can be selected. This pop-up, is displayed by pressing Ctrl-S and is only available when editing a cell in the **Type Name** or **C Type** column if the **Base Type** column for the edited row is blank or a pointer. Use the mouse or keyboard to highlight the structure name to insert. Once the desired structure name is highlighted either press the left mouse button or the Enter key. The structure name is inserted into the table cell, replacing any selected text (Figure 39). Press the Escape key to remove the structure name pop up dialog without inserting a structure name.



| Type Name | C Type                         | Size | Base Type      |
|-----------|--------------------------------|------|----------------|
| a         |                                |      | pointer        |
| int16     | ahrs_M_ECEfVelocity            |      | signed integer |
| int32     | AHRS_M_FDIR_BITS_T             |      | signed integer |
| int64     | AHRS_M_FDIR_T                  |      | signed integer |
| uint8     | ahrs_M_GPSCorrelationTimestamp |      | signed integer |
| uint16    | ahrs_M_GPSFixInformation       |      | signed integer |
| uint32    | ahrs_M_CPSTime                 |      | signed integer |
| uint64    | ahrs_M_HardwareStatus          |      | signed integer |
| float     | ahrs_M_InternalTimestamp       |      | signed integer |
| float     | float                          | 4    | floating point |
| double    | double                         | 8    | floating point |

Figure 39. Structure name pop-up

If a data type is currently in use in a data table then the size and base type may be constrained by the values in other columns on the same row of the affected table. For example, if a data type is a 2-byte integer and is used in a data table where the parameter is assigned a bit length of 10 bits then the data type size can't be changed to a single byte since a single byte's 8 bits is insufficient for the 10-bit parameter. The instance where the bit length exceeds the desired size must first be altered before the size can be updated. If an invalid size or base type is entered a dialog appears indicating the tables where the inconsistency exists, and the table cell reverts to its previous value.

The button commands are described below:

**Ins Row** Inserts an empty row below the currently selected cell's row. If no cell is selected then the new row is inserted at the end of the table.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 74 of 166 |

- Del Row** Deletes the row associated with each currently selected cell. If no row is selected then this has no effect. A data type cannot be deleted if it is currently used in a table; all references must be removed before the data type can be deleted.
- Up** Move the row(s) of the currently selected cell(s) up one row. The order of the data type definitions in the editor has no effect on data type usage, though it does determine the order of the types in the data type combo box lists. The capability to arrange the rows is solely for the user to group the data types as desired.
- Down** Move the row(s) of the currently selected cell(s) down one row. The order of the data type definitions in the editor has no effect on data type usage, though it does determine the order of the types in the data type combo box lists. The capability to arrange the rows is solely for the user to group the data types as desired.
- Undo** Undoes the last action performed (typing, paste, insert, delete, redo, etc.).
- Redo** Reverses the last action undone (typing, paste, insert, delete, undo, etc.).
- Store** Stores the changes made to data type definitions in the data type editor into the database. All tables are updated with the changes, including any tables currently open in a table editor.
- Close** Closes the data type editor window. If any changes have not been stored then a dialog appears allowing the user to confirm discarding the updates or to cancel closing the editor.

#### 4.10.3.11 Manage macros

The Macro Editor (Figure 40) provides a means of creating, modifying, and deleting macro definitions (see paragraph 4.6.6 for more information on macros).

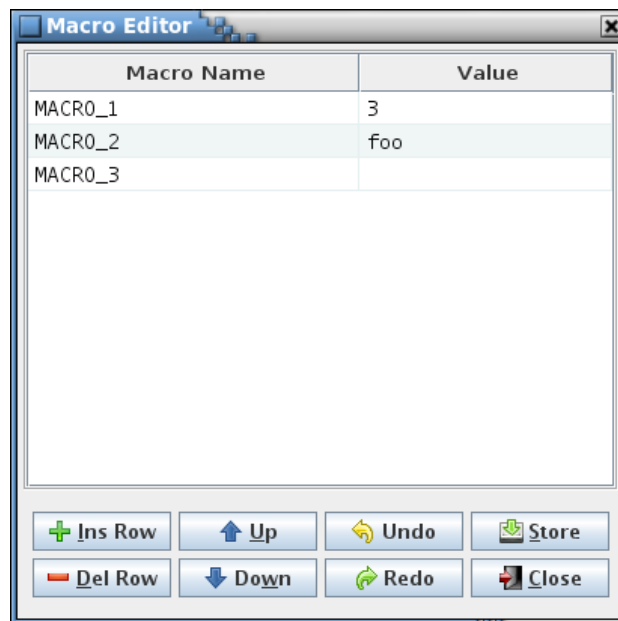


Figure 40. Macro Editor dialog

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 75 of 166 |

The editor column descriptions are as follows:

**Macro Name** The macro name is the text that represents the macro's value in a data table cell (the macro name is delimited and highlighted in the cell). Macro names are case insensitive and must be unique.

**Value** The macro value is the text that the macro name represents.

Each row in the table is a macro definition. Every definition requires a name, but the value may be blank. If a macro is currently in use in a table then the macro value is constrained by the input type of the column(s) in which the macro is referenced. For example, if a macro is inserted into a column of input type "Array index" then the macro's value can be blank or must be a number (or a series of numbers separated by commas), as required by the array index input type. If an invalid value is entered a dialog appears indicating the tables where the inconsistency exists, and the table cell reverts to its previous value.

The button commands are described below:

**Ins Row** Inserts an empty row below the currently selected cell's row. If no cell is selected then the new row is inserted at the end of the table.

**Del Row** Deletes the row associated with each currently selected cell. If no row is selected then this has no effect. A macro cannot be deleted if it is currently used in a table; all references must be removed before the macro can be deleted.

**Up** Move the row(s) of the currently selected cell(s) up one row. The order of the macro definitions in the editor has no effect on macro usage. The capability to arrange the rows is solely for the user to group the macros as desired.

**Down** Move the row(s) of the currently selected cell(s) down one row. The order of the macro definitions in the editor has no effect on macro usage. The capability to arrange the rows is solely for the user to group the macros as desired.

**Undo** Undoes the last action performed (typing, paste, insert, delete, redo, etc.).

**Redo** Reverses the last action undone (typing, paste, insert, delete, undo, etc.).

**Store** Stores the changes made to macro names or values in the macro editor into the database. All tables are updated with the changes, including any tables currently open in a table editor.

**Close** Closes the macro editor window. If any changes have not been stored then a dialog appears allowing the user to confirm discarding the updates or to cancel closing the editor.

#### 4.10.3.12 Assign message IDs

The **Assign messages IDs** command provides a method for automatically assigning a message identification (ID) number to appropriate tables (telemetry message IDs are assigned in the telemetry scheduler; see paragraph 4.10.4.2). A dialog appears (Figure 41) containing three input fields and a check box.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 76 of 166 |



Figure 41. Assign Table Message IDs dialog

The first field is for the name of the data field that represents the message ID number. The user must provide the name of the data field, exactly as it appears in the data tables, including capitalization and any internal spaces (leading and trailing spaces and tabs are ignored). The second field is the starting ID number, in hexadecimal. The last field is the ID interval which is used to calculate the next ID value in the sequence - the default is 1; any positive integer value is valid. The check box determines whether or not tables with an existing Message ID number are updated or left as is.

When **Okay** is selected the tables with the specified message ID data field name, in alphabetical order, are assigned a message ID number, beginning with the starting ID number, and with each subsequent ID number equal to the previous number plus the interval value. This also updates the database and the message ID number data fields for any open table editors. However, if the "Overwrite existing IDs" check box is not selected then any table that already has a message ID number assigned is ignored (i.e., it retains its existing ID value); in this case any newly assigned IDs are adjusted if needed to ensure a duplicate ID is not assigned. Select **Cancel** to exit the dialog without altering the message ID values.

#### 4.10.3.13 Show/edit fields

The purpose of the **Show/edit fields** command is to provide a means of displaying, editing, and removing data fields for one or more data tables and/or groups in a single editor (as opposed to displaying the data fields for a specific owner table in a table editor or group in the group manager). Selecting the **Show/edit fields** command produces a dialog displaying a table tree and a set of check boxes, one for each unique data field name currently in use by the project's data tables (see Figure 42 for an example; if no tables exist or no data fields are currently assigned then a warning dialog indicating that the issue appears instead). The user chooses the field(s) to display/edit by selecting the field's associated check box. The "Select all data fields" check box is used to alternately select and deselect all of the data field check boxes. The fields can be filtered by selecting one or more tables from the table tree – only the selected fields in the selected tables are displayed in the editor. If no tables are selected then no filtering occurs and the selected data field(s) are displayed for any table. Selecting the **Okay** button opens the data field editor, while the **Cancel** button closes the dialog without opening the editor.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 77 of 166 |

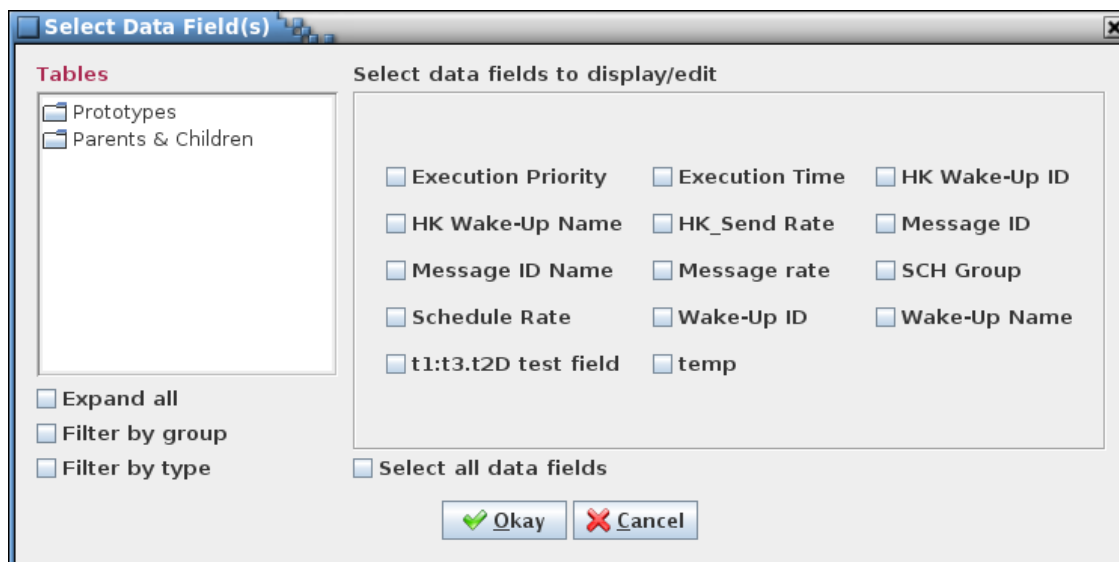


Figure 42. Example Select Data Field(s) dialog

An example of the editor dialog that appears is shown in Figure 43. The first column displays the data field owner. For a top-level structure table or non-structure table this is the table's name. For a child table the child's prototype and instance (or variable) name are displayed in the format *prototype.instance*, and are indented by an amount based on the number of levels the child is from its top-level structure. Fields that belong to a group display the group's name with "Group:" prepended.

The second column, Structure Path, displays the structure path for child tables, and is blank otherwise. If the field owner is not a structure table then the Structure Path column for this row has a gray background. The Structure Path column is not displayed if there is no structure table in the first column where that structure is a child of another structure. The Structure Path column lists each prototype and instance pair in the child structure's path leading back to its top-level parent. The top-level structure is shown first, then each subsequent child prototype and instance in the path. For example, note the row in Figure 43 for the table "ahrs\_M\_BeaconedTimestamp.beaconedTimestamp". Since its Structure Path column is not empty, the table is a child structure. Working upwards from the bottom of the list, "beaconedTimestamp" is a child of the structure "ahrs\_M\_OutData" (a structure that is of prototype "ahrs\_M\_OutData\_T"), which in turn is a child of the top-level structure "ahrs\_OutDataPacket\_T".

The remaining columns in the editor show the contents of the data fields chosen in the selection dialog. A cell with a gray background indicates that the associated table does not have the data field indicated by the cell's column; these cells may not be edited. A yellow background means that another cell or cells in the same column has an identical value (blank cells are ignored) – this can be used, for example, to detect duplicate message ID numbers. The rows can be sorted by selecting the column headers, as with other table editors in the application. Column order can be changed by dragging a column to a new position.

The data fields to display can be changed by pressing the **Select** button, causing the initial data field selection dialog to reappear. However, if there is an unstored change or field marked for removal a confirmation dialog appears first, allowing the user to choose between continuing with the selection

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 78 of 166 |

operation and discarding the changes, or canceling it and retaining the current selection with its unstored changes.

Data field values can be altered or the entire field removed via the editor. To change a field's value by highlight the cell and press the Enter key, or double left-click the mouse while the pointer is over the cell. The data type constraints set when the field was created (e.g., hexadecimal or positive integer) are enforced for the new field values. To remove a field entirely select the field using the mouse and press the **Remove** button. The field's background is displayed in red to indicate it is marked for removal. Multiple fields can be selected for removal. Selecting a marked field and pressing Remove again unmarks the field for removal, and the background color returns to normal.

If one or more cells is selected and the **Open** button is pressed then the table(s) associated with the selected cells are opened in a table editor. This button is ignored for a row containing a data field belonging to a group.

Selecting the **Print** button opens a printer selection dialog in order to print a copy of the editor table to the selected printer.

Unstored data field edits and removal selections can be undone by pressing the **Undo** button. Changes are undone in the order they were input. The Ctrl-Z key sequence performs the identical function. Conversely, undone changes can be reentered by pressing the **Redo** button or by the Ctrl-Y key sequence.

The **Store** button must be pressed to update the project's database with the data field value changes and removals. A confirmation dialog appears allowing the user to choose between continuing with the store operation or canceling it.

Selecting **Close** closes the data field editor dialog. If changes have been made to a data field that haven't been stored in the database, or there are one or more fields marked for removal then a confirmation dialog appears allowing the user to choose between continuing with the close operation and discarding the changes, or canceling it and keeping the editor open.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 79 of 166 |

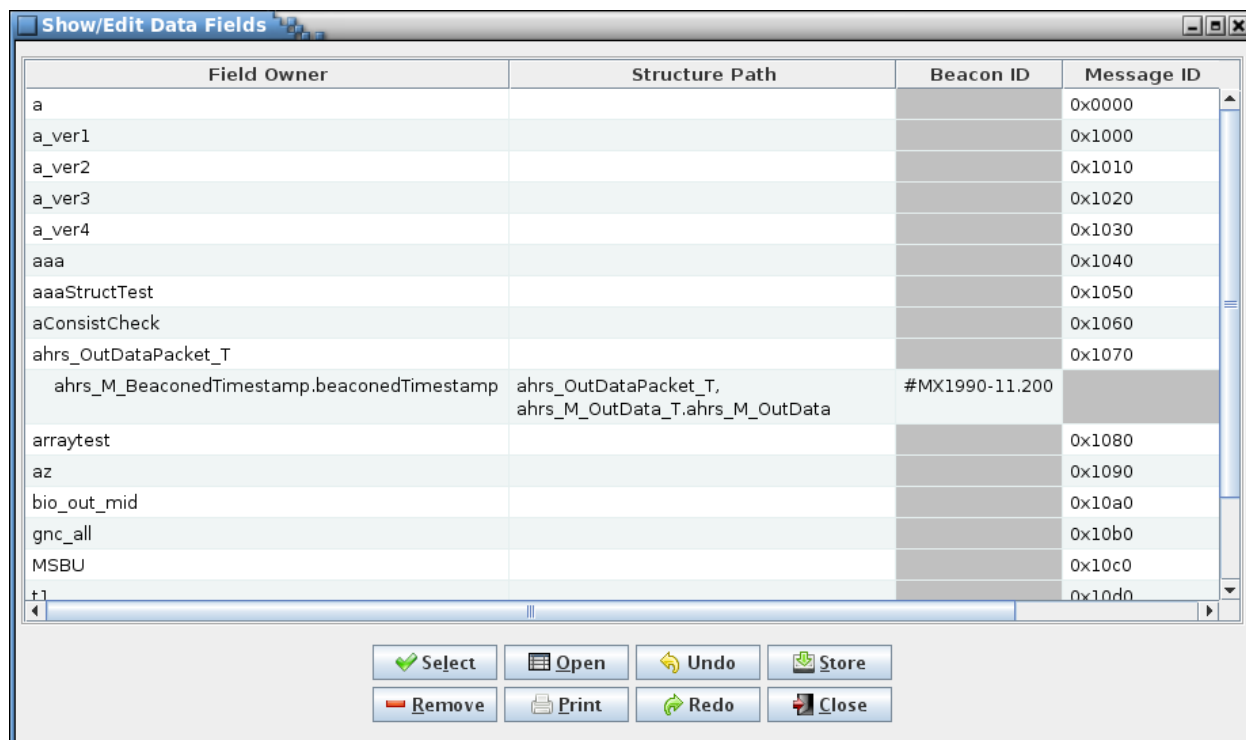


Figure 43. Example Show/Edit Data Fields dialog

#### 4.10.3.14 Search tables

The **Search tables** dialog provides a means of searching the project database data and internal tables for a specified text string (see Figure 44). Case sensitivity for the search is governed by the **Ignore text case** check box. The **Search data table cells only** check box, if selected, only displays matches found within the project database's data table cells and ignores those in the internal tables (see Appendix E.b; data table cell values stored in the custom values tables are included in the search). Enter the search text in the input field and select the **Search** button. The search results are displayed in the table at the bottom of the search dialog. The first column, **Table / Object**, shows the name of the data table or data object (table type definition, data field, group, script association, link, telemetry message, or scheduler entry) where a match is found. The second column, **Location**, describes the location of the match in the table/object. For a table the location is the column name in the table. A data object location depends on the type of object. For a data field this can be the field name, description, etc., whereas for a group or link this can be one of the tables or variables belonging to the group/link. The last column, **Context**, displays the string from the table or object containing the search text, with the search text highlighted.

The search results can be output to a file or printer by selecting the **Print** button. To exit the search dialog select the **Close** button.

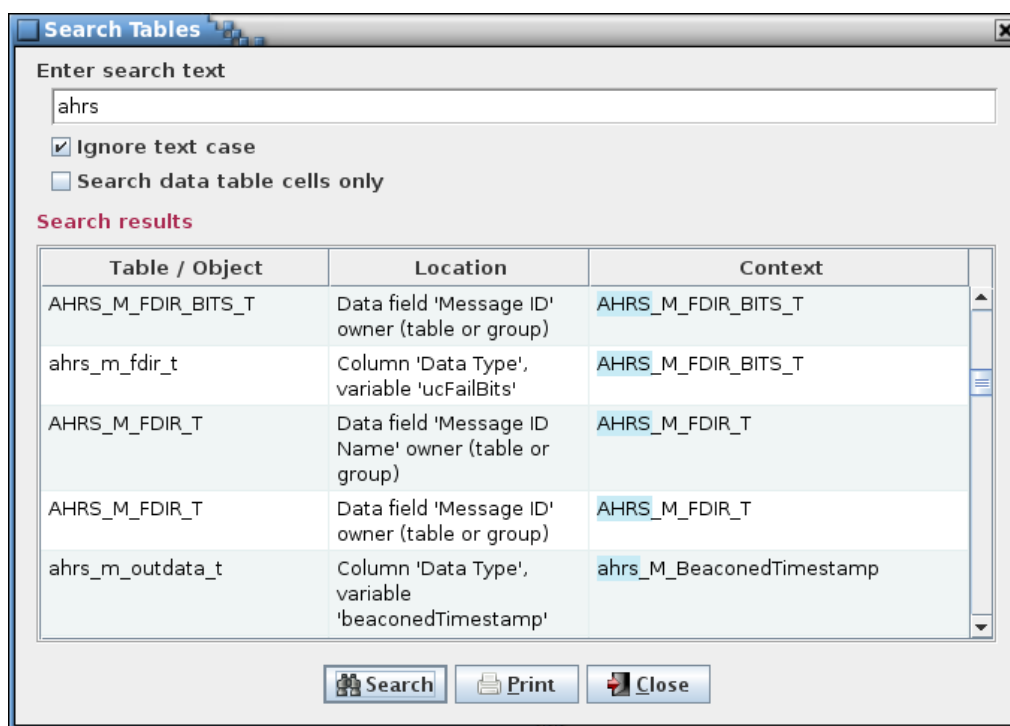


Figure 44. Search tables dialog

## 4.10.4 Scheduling

The scheduling commands are used to create and manage the information required to schedule telemetry downlink and application execution.

### 4.10.4.1 Manage links

The **Manage Links** command opens the Manage Links dialog (Figure 45). The link manager allows the user to create telemetry parameter linkages. These are simply groupings, selected by the user, of telemetry parameters (i.e., variables in the structures) with the same sample rate. The link information is used when assigning variables to telemetry messages in the telemetry scheduler (paragraph 4.10.4.2) to force the linked variables to be contained within the same message(s). The linkages created are specific to the data stream to which the linkage belongs. In other words, variables that are linked in one data stream do not have to be linked in another data stream.



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 81 of 166 |

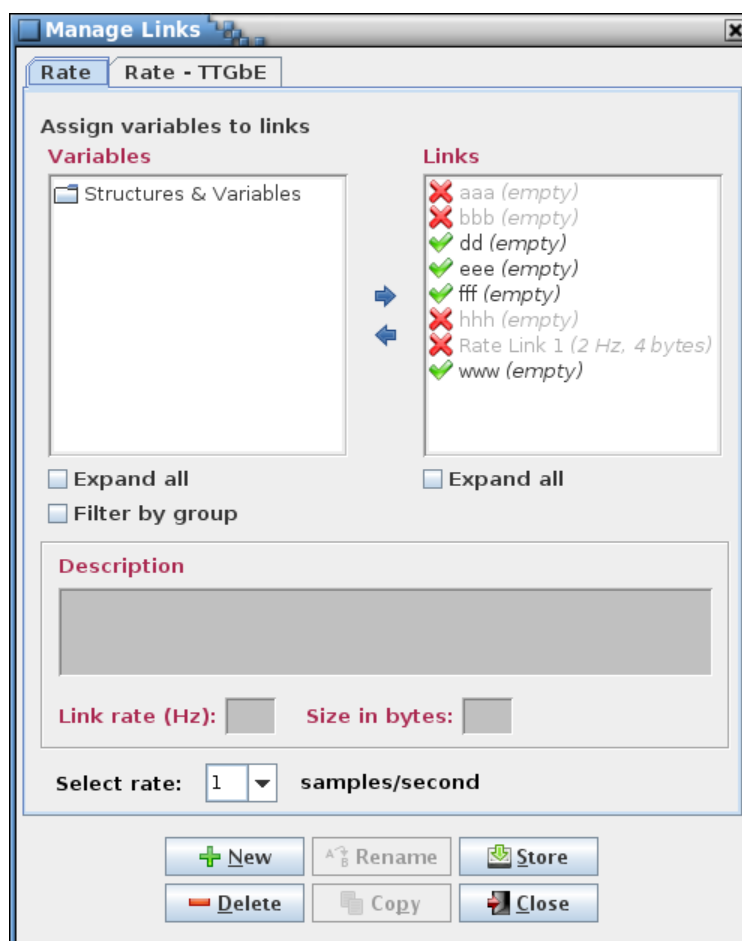


Figure 45. Manage Links dialog

The dialog's components are as follows. Along the top are the tabs that allow selection of the data stream in which to create, alter, or delete linkages. The upper left displays a tree showing structures and their members (under the heading "Variables"), both child structures and primitive variable types. Which variables are contained in the tree is determined by the rate chosen from the "Select rate:" combo box pull down menu near the bottom of the dialog. In the upper right is a tree showing the links and their member variables (under the heading "Links"). Between the trees are left and right arrows for adding or removing a variable from a link. Each tree also has one or more check boxes to filter the tree information. Below the trees is an input field for providing a description of a link. Underneath this is the link rate, in samples per second, and the total size in bytes of the link, which is the sum of the byte sizes of the variables assigned to the link. The description, rate, and size fields are active when a single link is selected in the link tree.

In the Links tree, displayed in parentheses next to each link name, is that link's rate and size in bytes (the same information that appears below the description field when this link is selected). A link's rate must match the selected sample rate (or the link must have no variables assigned) in order for it to be assigned variables from the variable tree. A check mark (✓) beside the link name indicates that the link can be assigned variables from the variable tree, and a red X (✗) is displayed if the link is incompatible with the selected sample rate.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 82 of 166 |

A variable may not be assigned to more than one link for a given data stream. Once assigned to a link the variable still appears in the variable tree but it is disabled (grayed out and not selectable). When a variable is removed from a link it becomes enabled again in the variable tree. Once an entire structure's complement of variables is assigned the structure itself is disabled in the variable tree, and if all structures are assigned then the "Structures & Variables" tree node itself is disabled.

To create a link select the **New** button and provide a link name and, optionally, a description, in the input dialog that appears (Figure 46). The description can be altered later in the main dialog. The new link name appears in the link tree. The link name may not be blank, nor is the name allowed to match that of an existing link in the selected data stream. The link name may contain alphanumeric, space, and punctuation characters. There is no constraint on the length of the name.

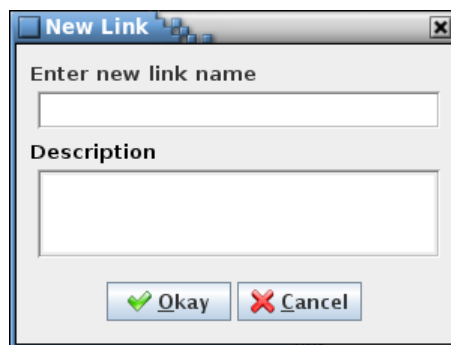


Figure 46. New Link dialog

To add variables to a link select the link in the Links tree using the mouse or keyboard. In the Variables tree expand the tree as needed and select one or more variables using the mouse or keyboard. Multiple variables can be selected simultaneously by holding the Ctrl or Shift keys down when making a selection. Selecting a structure automatically includes its child structures (and their children, etc.), and all variables associated with the structure(s). Choosing a child structure automatically includes its parent structure, and its parent's parent, etc., up to the top-level of the tree, but does not include any of its sibling variables (i.e., a variable having the same parent structure and at the same tree level as the chosen variable). The exception is if the selected variable is bit-packed with one or more variables; in this case all of the packed variables are automatically included (see paragraph 4.6.4.1). Finally, select the right arrow button in the center of the dialog. The variable(s) chosen appear in the selected link, and the link's tree is expanded to show the variable(s) added. Note that the variable hierarchy is preserved in the link's tree. More variables can be assigned to the link as described above.

To remove structures or variables from a link expand the link's tree and select the structure(s) and/or variable(s) to remove using the mouse or keyboard. Then select the left arrow button in the center of the dialog to delete the structure(s) or variable(s) from the link. A structure's children (and their children, etc.) and variables are removed along with the chosen structure. If a bit-packed variable is removed then all other variables packed together with it are removed as well, even if not explicitly selected.

To delete a link, first select it in the Links tree, then select the **Delete** button. Multiple links can be removed simultaneously if desired by highlighting them while using the Shift or Ctrl keys.

To rename a link, select a single link from the Links tree, then press the **Rename** button. An input dialog appears with the name of the selected link in the input field. Alter the name as desired and select **Okay**

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 83 of 166 |

to change the link's name. The renamed link name may not be blank, nor is the name allowed to match that of an existing link in the current data stream. Select **Cancel** to exit the input dialog without affecting the link's name.

A link, including its description, variable structure(s) and variable(s), can be copied from one data stream to another. First select one or more links from the Links tree to be copied, then press the **Copy** button. A dialog appears (Figure 47) with the name(s) of the selected link(s) in the link name text field. Below the link name field is an array of check boxes, one for each of the project's data stream names. The current data stream is grayed out and can't be selected (recall that a variable may belong to only one link in a given data stream). Select one or more data streams to which the link (or links) is to be copied. Press the **Copy** button to copy the link(s) to the selected data stream(s). If the targeted data stream doesn't support the link's sample rate or if the link name already exists in the stream then the link isn't copied to that stream; a dialog is displayed indicating which link and stream combination(s) were skipped (see example in Figure 48). Any variables in the link that are not assigned the identical sample rate in the target data stream are removed from that stream's copy of the link. Select **Cancel** to exit the copy dialog without copying the link.

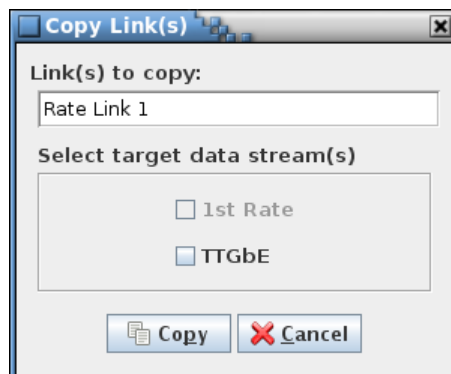


Figure 47. Copy Link(s) dialog

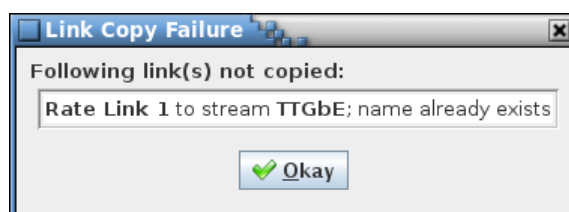


Figure 48. Example link copy failure dialog

A link's description can be added or changed by first selecting the link in the Links tree. The current description for the link appears in the Description input field. The description can then be changed as desired.

Changes to the links (descriptions and member variables) for all data streams are stored in the database only when the **Store** button is pressed. If changes have been made a confirmation dialog first appears. Select **Okay** to store the updates; select **Cancel** to exit the confirmation dialog without altering the database.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 84 of 166 |

Select the **Close** button to exit the link manager dialog. If there are any unsaved link changes in any of the data streams a dialog appears requesting confirmation to discard the changes. Select **Okay** to exit the link manager, losing any unsaved changes. Select **Cancel** to return to the link manager dialog.

#### 4.10.4.2 Telemetry

The **Telemetry** command opens the Telemetry Scheduler dialog (Figure 49). The telemetry scheduler is used to assign a project's variables to telemetry messages. The messages can then be used to build a CFS housekeeping "copy table", for example by using the copy table script provided with the CCDD application. The available messages are determined by the rate parameters. These parameters can be altered in the Rate Parameters dialog (see paragraph 4.10.4.4). Before the telemetry scheduler can be used the following must be done:

- Adjust the rate parameters to establish the correct boundaries for handling the project's telemetry
- Assign rates to the variables to be downlinked in the Edit Table dialog (see paragraph 4.10.3.2)
- (Optional) Assign variables that are desired to be sent down in the same message to a link using the link manager (see paragraph 4.10.4.1)

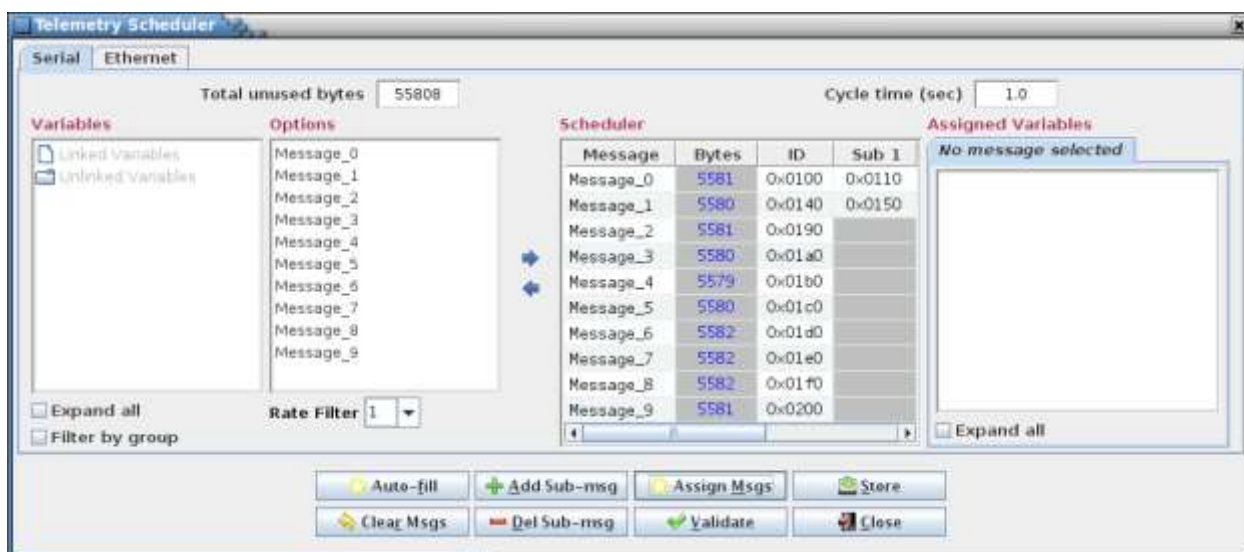


Figure 49. Telemetry Scheduler dialog

The Telemetry Scheduler dialog is composed of number of components. Along the top are the tabs for each defined data stream (see paragraph 4.9). Each stream has its own variable rates and message assignments. Selecting a tab displays the dialog components associated with that data stream. At the top left is displayed the total number of bytes remaining to be assigned. This value is equal to the maximum bytes per second (from the Rate Parameters dialog) minus the size in bytes of all the variables assigned to messages. At the upper right is the cycle time which is amount of time it will take for the messages to repeat; e.g., a cycle time of 2 seconds means that each message in the table will be sent down once every 2 seconds.

The **Variables** tree, at the left of the dialog, displays the variables available for downlink in tree format. Only variables assigned a rate are displayed in the variable tree. The tree is separated into two sections: **Linked Variables** and **Unlinked Variables**. **Linked Variables** displays the links per the currently selected

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 85 of 166 |

rate filter. Each link has the variables assigned to the link via the Link Manager. Unlinked Variables displays all the variables with a rate matching the selected rate filter. Variables assigned to a link are also displayed, but are grayed out and cannot be unselected. Beneath the variable tree are two check boxes that are used to expand the tree or filter it by group.

To the right of the **Variables** tree is the **Options** list. This list displays the available options, based on the selected rate filter, for assignment of the variables to the messages. For example, if the rate filter is set to 5 and there are 10 available messages then the options displayed are "Option 1: Messages 1, 3, 5, 7, 9" and "Option 2: Messages 2, 4, 6, 8, 10" (assuming the cycle time is one second).

The **Rate Filter**, just below the **Options** list, is a pull down list of the data stream's available rates. The selected rate is used to filter the **Variables** tree and the **Options** list. The variable tree only displays variables that are assigned the rate selected determined by the rate filter. The user can change the rate filter at any time to make the **Variables** tree and **Options** list update.

The **Scheduler** table, located on the right of the **Options** list, contains a table with every available message. The scheduler table has at least three columns: the "Message" column, which displays the message name; the "Bytes" column, which displays the remaining bytes for each message; the "ID" column, which display the message's ID value. Extra columns, labeled "Sub 1", "Sub 2", etc. are added if any message has a sub-message; for messages without the specified sub-message the column is grayed out. The "Bytes" column is updated as variables are added or removed from the message. A negative number indicates that the message is over assigned (i.e., insufficient bytes available to contain the assigned variables); the "Message" column is displayed in red in this case. The message names and the ID (and sub ID) values can be edited in the **Scheduler** table, or can be automatically assigned in the Assign Telemetry Messages dialog called via the **Assign Msgs** button.

The **Assigned Variables** tree, located to the right of the **Scheduler** table, shows the variables assigned to the most recently selected message in the **Scheduler** table.

In between the **Options** list and **Scheduler** table are two arrow buttons. The right arrow button assigns one or more variables to a message. The assignment process is described below. The left arrow button removes one or more variables from a message. The removal process is described below.

At the bottom of the telemetry manager dialog is the button panel. The buttons functions are as follows:

**Auto-fill** Assigns all the variables in the variable tree that are not yet assigned to messages. Auto-fill does this optimally so each message is filled as evenly as possible. If auto-fill is successful then all the variables are assigned to an appropriate message. If auto-fill is unable to assign every variable (due to insufficient room or no available option) it displays a dialog indicating how many variables are left unassigned.

**Clear Msgs** Removes all message and sub-message variable assignments.

**Add Sub-msg** Adds a sub-message to the message currently selected in the **Scheduler** table. Any number of sub-messages may be added. Adding a sub-message removes all of a message's sub-message variable assignments. This is done since the number of sub-messages affects the rate at which a sub-message is sent.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 86 of 166 |

**Del Sub-msg** Removes the currently selected sub-message in the **Scheduler** table. Deleting a sub-message removes all sub-message variable assignments for that message. This is done since the number of sub-messages affects the rate at which a sub-message is sent.

**Assign Msgs** Opens the Assign Telemetry Messages dialog (Figure 50). This dialog provides a means for assigning message names and/or message IDs to all messages and sub-messages based on a pattern, starting value, and interval value.

Figure 50. Assign telemetry message names and IDs dialog

To assign message names the “Assign message names” checkbox must be selected. A pattern for the names is entered in the “Name pattern” input field. This pattern must adhere to alphanumeric naming constraints (see paragraph 4.8) except that it also must contain a single “%<0#>d” format string somewhere after the first character. The format string is replaced with a sequence number when the names are assigned. The optional ‘0#’, where ‘#’ represents one or more digits, provides a means of padding the sequence number with leading zeroes so as to bring its length to # digits. The first message name uses the pattern and the “Starting number” field value; the “Message interval” value is added to the previous message’s number for each subsequent message name. For example, with the values as shown in Figure 50 the message names are “Message\_001”, “Message\_002”, “Message\_003”, etc., until all messages are named.

To assign message IDs the “Assign message IDs” check box must be selected. The “Starting ID” field is the starting ID number, in hexadecimal. The “ID interval” field is the interval used to calculate the next ID value in the sequence - the default is 1; any positive integer value is valid. The “Overwrite existing IDs” check box determines whether or not messages with an existing ID number are updated or left as is. The IDs are assigned beginning with the starting ID number and with each subsequent ID number equal to the previous number plus the interval value.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 87 of 166 |

When **Okay** is selected the message names and/or IDs, based on the check box states, are assigned to the messages in the **Options** list and **Scheduler** table. Press Cancel to exit the dialog without changing the message names or IDs. Note that the telemetry scheduler's **Store** button must be used to update the messages in the project database.

- Validate** Checks the variables in each message to ensure they are still valid for the message. A variable is deemed invalid if the variable no longer exists in the structure, or if it's bit length, data type size in bytes, or sample rate does not match the data for the variable in its structure table. Any invalid variables are removed from the message. Validation can be set to be performed automatically; see paragraph 4.10.4.6. Automatic validation always occurs when the telemetry scheduler's "copy table" output is accessed via the script data access method.
- Store** Stores the telemetry scheduler data in the project database. Any changes not stored before closing the telemetry scheduler dialog are lost.
- Close** Closes the telemetry scheduler dialog. If there are any unsaved changes in any of the data streams a dialog appears requesting confirmation to discard the changes. Select **Okay** to exit the telemetry scheduler, losing any unsaved changes. Select **Cancel** to return to the telemetry scheduler dialog.

The following describes the process to manually assign a variable to a message. First, one or more variables and/or links are selected in the **Variables** tree. A grayed out variable, structure, or link indicates that it is already assigned and can't be selected (linked variables also appear in the unlinked portion of the variable tree, but are grayed out and can't be assigned individually). After selecting one or more variable an option is chosen from the **Options** list. To aid in deciding which option to choose, the **Scheduler** table temporarily updates the "Bytes" column for the option's message(s), displaying the message size if that option is chosen. Also, the text of the "Message" column changes to either green, signifying there is enough room in the message for the variable(s), or red, signifying there is insufficient room. Changing which option is selected resets any of the temporary changes and updates the message(s) based on the new option. This allows the user to evaluate each option before selecting a choice. After the user has decided on an option, pressing the dialog's right arrow button assigns the selected variable(s) to each message in the selected option. Adding a linked or bit-packed variable also adds the variables associated with it; i.e., all members of the link are added, and all other variables bit-packed with the selected variable are added. Once a variable is assigned to a message it is grayed out in the **Variables** the tree so it can't be assigned more than once.

A variable or variables can be removed manually from the messages and sub-messages. First a message or sub-message is selected (either in the **Scheduler** table or the **Assigned Variables** tree), which causes the **Assigned Variables** tree to display the variables for the selected (sub-)message. The user selects from the tree one or more of the variables or structures that are to be removed and then presses the dialog's left arrow button. The selected variable(s) is removed from the all messages to which it is assigned. Removing a linked or bit-packed variable also removes the variables associated with it; i.e., all members of the link are removed, and all other variables bit-packed with the selected variable are removed. Any de-assigned variable is no longer grayed out in the **Variables** tree to signify it is available to be re-assigned.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 88 of 166 |

#### 4.10.4.3 Applications

The **Applications** command displays the Application Scheduler dialog (Figure 51). The Application Scheduler is used to schedule when a project's Apps should be ran. The available time slots for when an App can be executed is determined by the app parameters set in the **App Parameters** dialog. If the app parameters are not set then the App Manager will use the default parameters. Before using the App Manager it is recommend to...

- 1) Set the app parameters so the correct boundaries can be determined. This can be done in the **App Parameter** dialog.
- 2) Create apps that need to execute during your projects execution. Apps can be created in the **Group Manager** dialog.

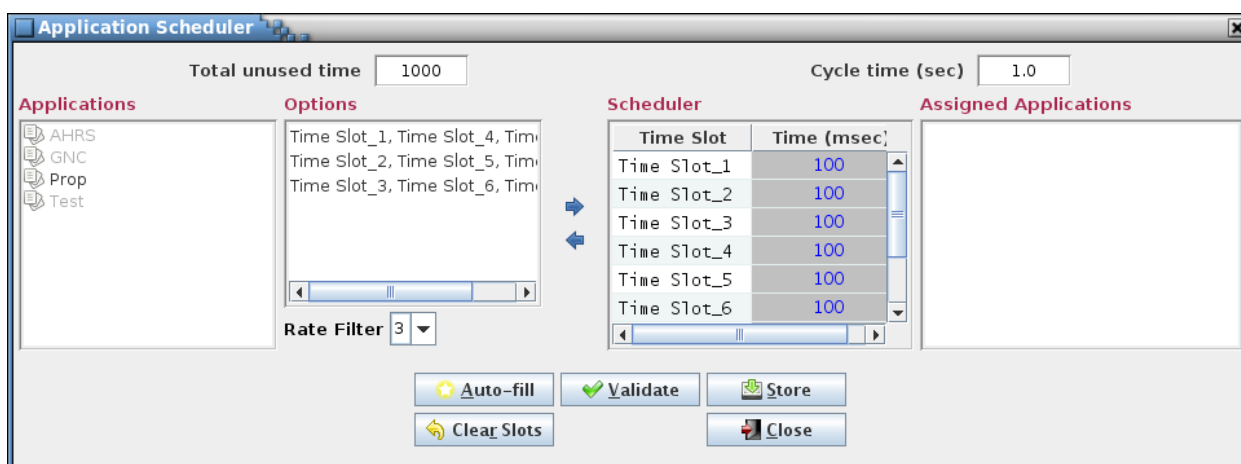


Figure 51. Application Scheduler dialog

The App Manager Dialog components are as follows.

The **App list**, which is located on the left of the dialog, displays all the available apps to be scheduled. Any app that has already been assigned will appear grayed out. Also any app's execution rate that does not match the rate filter will appear grayed out. Any gray out app will not be able to be selected.

The **Rate Filter**, which is located right of the app list and atop the option list, contains a list of all the available execution rates. The user can change which rate is selected by using the drop down box. Changing the rate will cause the **App List** to update to gray out any apps not at the selected rate. Also the options list will change to display options only for that given rate.

The **Option List**, which is located right of the app list and below the rate filter, contains a list of all the available options at the selected rate. The option list will be changed based on which rate is set in the rate filter. For example, if the rate is set to one and 40 time slots are available then the options will be "Option1: TimeSlot\_1", "Option2: TimeSlot\_2", "Option3: TimeSlot\_3.....", until all 40 time slots are listed with an option. (Assuming then cycle time is 1 second)

The **Scheduler Table**, which is located on the right of the dialog and left of the variable list, is a table of all the available time slots. One column displays the time slots and the other column displays the remaining available time for that time slot (In milliseconds). The available time decreases when apps are added to that time slot and increase when apps are removed. If the number of bytes available ever becomes negative then the time slots text will be changed to red to indicate a potential error.



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 89 of 166 |

The **Variable List**, located right of the scheduler table, displays any app assigned to a selected time slots. The **Variable List** will be updated anytime the user selects a time slot. (If multiple time slots are selected only the first selected time slot's apps will be displayed) This allows the user a quick way to view the apps currently assigned to a time slot.

The **Assignment Panel**, located at the center of the dialog, contains three buttons. Each button is used in assigning or removing apps. The buttons are described below.

**Right Arrow** Allows users to assign apps to a time slot.

**Left Arrow** Removes apps from the schedule table.

The **Button Panel**, which is located at the bottom of the App Manager Dialog, has three buttons which are described below.

**Auto-fill** Attempts to assign all the remaining apps to time slots. Auto fill will do this optimally so each time slot will be filled evenly. If the auto fill is successful then all the apps will be assigned to appropriate time slots. If the auto fill cannot assign every app it will display a window to the user telling them how many apps are left unassigned. The auto fill process is used to help streamline the assignment process.

**Clear Time Slots** Removes every app from the time slots, therefore presenting the user will an empty schedule table.

**Store** Stores the schedule table data to the database. If a user does not store before closing the App Manager Dialog then any changes made will be lost.

**Close** Closes the App Manager Dialog. (**Note:** Closing the dialog without storing the changes will cause any changes to be lost)

The assignment process is as follows. The user first selects an app from the **App List**. A user can select more than one app, however no grayed out app is selectable. After an app(s) has been selected the next step is to select an option to determine which time slots the app will be assigned too. To help decide which option is best, the user can select an option which will temporarily update the time columns of the time slots in the option. Also the text of the app in scheduler table will be change to either be green, to signify there is an enough room, or red, to signify there is not enough room. For example if the user selects an app that has run time of 2 milliseconds, the time column for each message in the selected option will be subtracted by 2 milliseconds. Also the color will be updated as stated earlier. Simply changing which option is selected will reset any of the temporarily changes and then update any time slots in the new select option. This allows the user to evaluate each option before choosing which one they want. After the user has decided on an option, the user can click the **Right Arrow** button to assign the selected app(s) to each time slot in the option. Adding an app(s) will gray it put from the **App List**.

The process for removing an app from the schedule table is as follows. First a user must select a time slot they want to remove from. This will cause the variable list to display the apps for the selected time slot. Then the user can select any of the apps from the app list they wish to remove. Once the user has selected the app(s), they can click the **Left Arrow** button to remove it from the time slot. Removing an app will can cause two things to happen. First, if the app was also in other time slots then the app will be removed from each time slot. Second, any removed app will become un-grayed out in the **App List** to signify it is now available to be re-assigned.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 90 of 166 |

The data created from the App Manager is used to create scheduler tables for the project. The schedule table is used to handle the execution of the projects app. The user can choose to use the provided scripts to create the scheduler table or choose to write their own script.

#### 4.10.4.4 Rate parameters

The **Rate parameters** command displays the dialog shown in Figure 52.

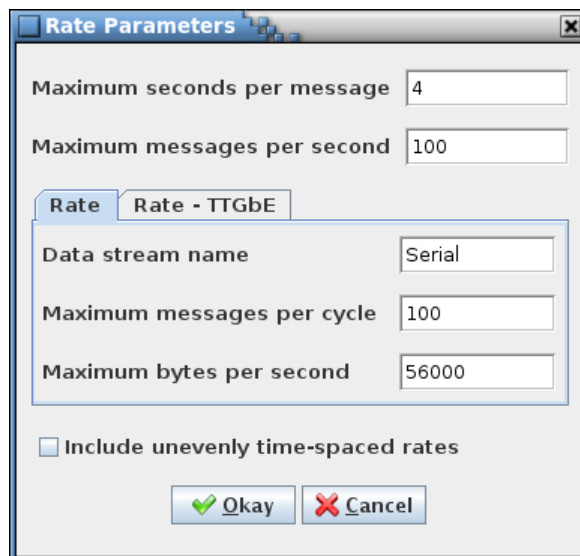


Figure 52. Rate Parameters dialog

This dialog is used to set the bounds for the sample rates for each defined data stream (See paragraph 4.9) and from these generate the selections in the drop down menu for the rate column(s) in the data tables, the link manager (see paragraph 4.10.4.1), and the telemetry scheduler (see paragraph 4.10.4.2). These parameters also define the total number of messages and maximum message size. Each of these parameters must be a positive, non-zero integer value. The first two parameters, maximum seconds per message and maximum messages per second, are common to all data streams, while the remaining parameters are assigned by data stream. A data stream is selected via the tabs, which reflect the rate column names, in the center of the dialog. The definitions of these values are as follows:

- Maximum seconds per message** The slowest period, in seconds, that a message is downlinked. Example: If 5 is entered then a 5 seconds per sample is the slowest rate allowed to be selected as the rate for a telemetered value. All rates between this and 1 second/sample that are multiples of the period are added to the rate list. Rates slower than 1 sample per second are displayed in the format "1/#" where # is the number of seconds between samples
- Maximum messages per second** Maximum number of telemetry messages that can be downlinked in a single second. For a cycle time of one second this value is the same as the **Maximum messages per cycle** value
- Data stream name** This value is specific for a data stream. Alternate name to associate with the rate column and used in the link manager for the tab name

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 91 of 166 |

(if no data stream name is entered the rate column name is used instead)

**Maximum messages per cycle** This value is specific for a data stream. The number of telemetry messages that are downlinked during a single cycle through the message list. For a cycle time of one second this value is the same as the **Maximum messages per second** value

**Maximum bytes per second** This value is specific for a data stream. Maximum number of bytes that can be downlinked during a single second

The evenly time-spaced sub-second rates are calculated using the above values. For example, given a total messages per cycle of 10 and a maximum messages per second of 10 then only rate values that are a factor of 10 – i.e., 1, 2, 5, and 10 samples per second – are available. The check box labeled **Include unevenly time-spaced rates**, when checked, causes the remaining, unevenly time-spaced rates to be included in the list of rates (in the example this is all values between 1 and 10 – i.e., 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10 samples per second).

#### 4.10.4.5 App parameters

The **App Parameters** command displays the Application Parameters dialog (Figure 53).

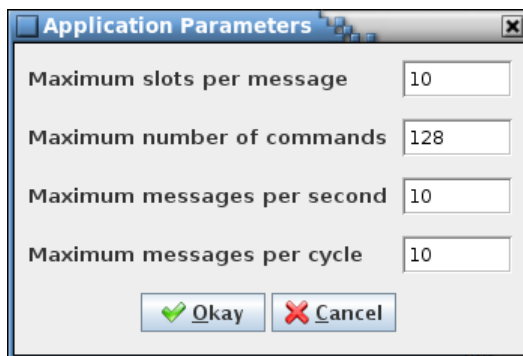


Figure 53. Application Parameters dialog

This dialog is used to set parameters for the application scheduler table. The maximum slots per message and the maximum number of commands define the boundaries, while the maximum messages per second and the maximum messages per cycle are used for scheduling the applications. Each parameter must be a positive, non-zero integer value. The definitions of these values are as follows:

**Maximum slots per message** The number of slots available in each time slot of the scheduler table. If 10 is entered then every time slot will have 10 available slots for an application. The Application Scheduler doesn't allow a time slot to have more applications assigned to it than this parameter.

**Maximum number of commands** The maximum number of commands that can be created for the scheduler table.

**Maximum messages per second** The maximum number of time slots that are available for an application to be scheduled in a second.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 92 of 166 |

#### Maximum messages per cycle

The number of time slots that are executed during a single cycle through the time slot list. For a cycle time of one second this value is the same as the **Maximum messages per second** value

#### 4.10.4.6 Auto validate

The **Auto validate** check box menu item, if checked, automatically validates the telemetry and application scheduler data each time the respective scheduler is opened. Validation requires extensive interaction with the project database and can therefore be time-consuming, therefore the default is to have automatic validation disabled. The automatic validation check box state is restored to that used in the previous session.

For the telemetry scheduler the validation process checks the variables in each telemetry message. A variable is deemed invalid if the variable no longer exists in the structure, or if it's bit length, data type size in bytes, or sample rate does not match the data for the variable in its structure table. Any invalid variables are removed from the message. Automatic validation always occurs when the telemetry scheduler's "copy table" output is accessed via the script data access method.

For the application scheduler the validation process checks the applications in each time slot. An application is invalid if the application no longer exists or if the execution rate doesn't match between the application and the time slot data. Any invalid applications are removed from the time slot. Automatic validation always occurs when the application scheduler's "scheduler table" output is accessed via the script data access method.

Manual validation may be performed via commands in the schedulers; see paragraphs 4.10.4.2 and 4.10.4.3 concerning the **Validate** buttons.

#### 4.10.5 Script menu

The **Script** menu contains commands for associating scripts with data tables and fields, and for executing the stored associations. Scripts are a means of accessing the table data in order to create output files (e.g., C header files or ITOS record files) or otherwise manipulate the data. The script languages supported by the application include JavaScript, Python, Ruby, and Groovy. Example scripts are provided with the application. These can be modified, or new scripts written as needed by the user. See paragraph 4.11 for more information on the use of scripts to access the table data.

##### 4.10.5.1 Manage

The **Manage** command provides the means for associating scripts and database tables. This is required before executing the scripts. The associated scripts and tables can be stored in the database so that frequently used associations can be quickly executed.

When the command is selected the Manage Script Associations dialog (Figure 54) is displayed. The dialog is divided into four sections: script selection, table selection, script associations, and command buttons.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 93 of 166 |

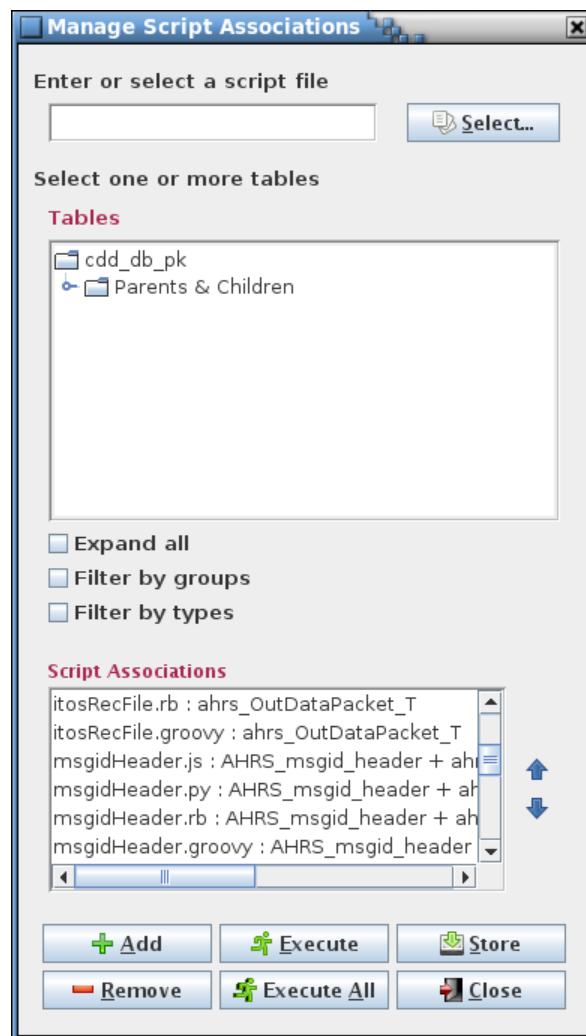


Figure 54. Manage Script Associations dialog

The script selection field and **Select...** button are used to select a script file. A script name (with file path) can be typed into the field; alternatively, pressing the **Select...** button displays a file selection dialog from which a script file can be located and selected. Script names must be a valid for use as a file name (e.g., may contain spaces, but not certain special characters, dependent on the operating system, such as a forward slash (/)).

The table tree displays all of the parent tables and their child tables. The user expands the tree branches and selects one or more tables (see paragraph 4.6.3 for more information on table trees). When a structure table is chosen all of its child tables are automatically included when the script association is executed; therefore the child tables do not have to be explicitly selected when creating the association.

The script associations list displays the script associations that are stored in the project database. Associations are grayed out if the script file doesn't exist on the local machine or an associated table doesn't exist in the project database. These disabled associations can be selected for removal, but can't be executed.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 94 of 166 |

After choosing a script and (if needed) table(s), selecting the **Add** button creates the script association, which appears in the Script Associations list below the table tree. A script may be used in more than one association, and a table may be used in any number of associations; however, duplicate associations, i.e., those utilizing the same script and table(s), are not added to the list.

A script can be added without associating it with a table. This is the case when the script performs actions that do not need data from one of the database data tables.

An association in the list may be deleted by selecting it using the mouse or keyboard, then pressing the **Remove** button. Multiple associations may be removed simultaneously by selecting more than one from the list by using the Shift or Ctrl keys.

The order of the associations may be adjusted by selecting one or more of them, then pressing the up or down arrow buttons to the right of the associations list. The ordering is preserved when the associations are stored and retrieved from the database, and can be useful for keeping affiliated associations near one another.

Script associations may be executed from within this dialog. This is similar to execution of the associations from the Execute Scripts dialog (see paragraph 4.10.5.2) except that in this dialog the associations do not have to be stored in the database to be executed. This provides a means to create a one-use association for immediate execution. Select one or more associations and press the **Execute** button to execute the selected association(s). Pressing the **Execute all** button executes all of the scripts in the list, regardless of their selection status. When script execution completes a status message is written to the event log. If an error occurred, preventing successful script completion, an error dialog also appears, providing details on the cause of the error.

The **Store** button stores the list of associations in the database in the order that they are displayed in the list. If changes have been made a confirmation dialog first appears. Select **Okay** to store the updates; select **Cancel** to exit the confirmation dialog without altering the database.

Select the **Close** button to exit the script associations dialog. If there are any unsaved association changes a dialog appears requesting confirmation to discard the changes. Select **Okay** to exit script associations dialog, losing any unsaved changes. Select **Cancel** to return to the Manage Script Associations dialog.

#### 4.10.5.2 Execute

Selecting the **Execute** command causes the Execute Script(s) dialog to appear (Figure 55).

The dialog displays a list of the stored script associations. Associations are grayed out if the script file doesn't exist on the local machine or an associated table doesn't exist in the project database. These disabled associations can't be selected for execution.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 95 of 166 |

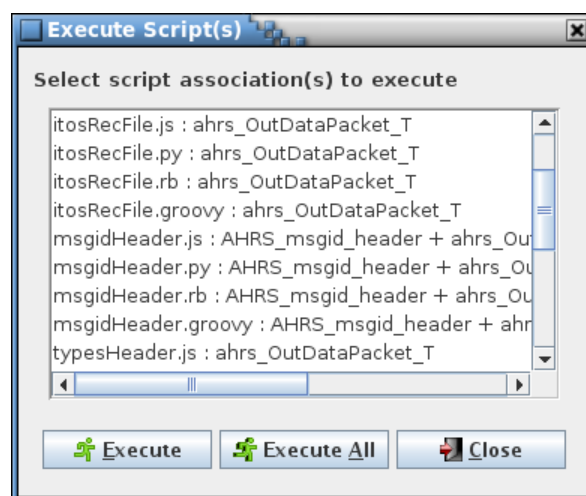


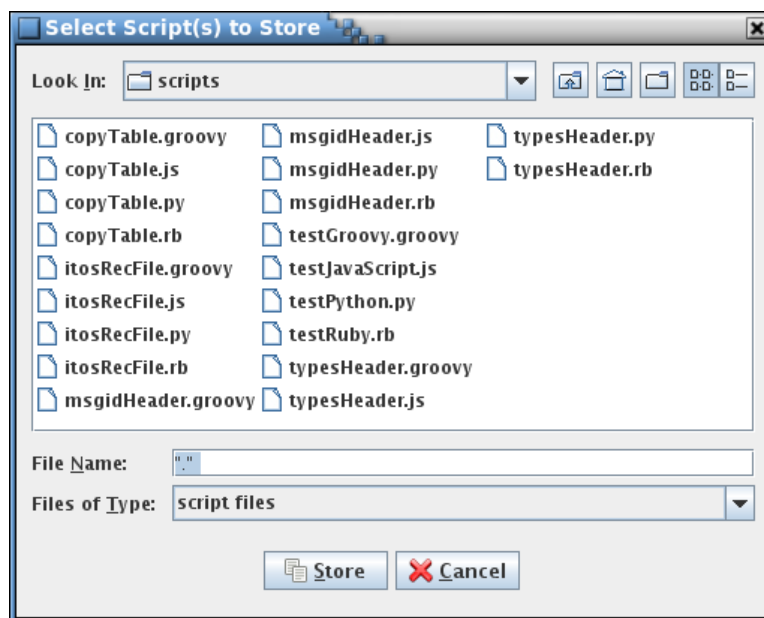
Figure 55. Execute Script(s) dialog

Select one or more associations and press the **Execute** button to execute the selected association(s). Pressing the **Execute all** button executes all of the scripts in the list, regardless of their selection status. When script execution completes a status message is written to the event log. If an error occurred, preventing successful script completion, an error dialog also appears, providing details on the cause of the error.

Select the **Close** button to exit the Execute Script(s) dialog.

#### 4.10.5.3 Store

The **Store** command is used to store scripts in the database. This can provide a means of script security and configuration management as well as allow all users, including those at remote sites, to access a common set of script files. Selecting the command causes a file selection dialog to appear (Figure 56).



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 96 of 166 |

Figure 56. Script selection dialog

Only files with extensions supported by the available script engines are shown. However, other files are displayed if "All Files" is selected from the "Files of Type" drop down menu. After selecting one or more files, selecting the **Store** button stores the contents of the files in the project, each as a separate table. Select the **Cancel** button to exit the dialog without storing any files. The **Retrieve** command (see paragraph 4.10.5.4) provides the means for retrieving the stored files from the project.

When a file is stored the application first searches it for the first line containing the text "description:". The search ignores case, so any combination of upper and lower case characters constitutes a match. If found, the remaining text on the same line in the file (sans any leading or trailing white space character(s)) is stored with the file as its description. The description appears alongside the file name in the Retrieve Script(s) dialog (Figure 57). If no match is found then the description text in the dialog is blank.

Note that this command can be used to store any text file in the database, not only script files.

#### 4.10.5.4 Retrieve

Selecting the **Retrieve** command causes the Retrieve Script(s) dialog to appear (Figure 57).



|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 97 of 166 |

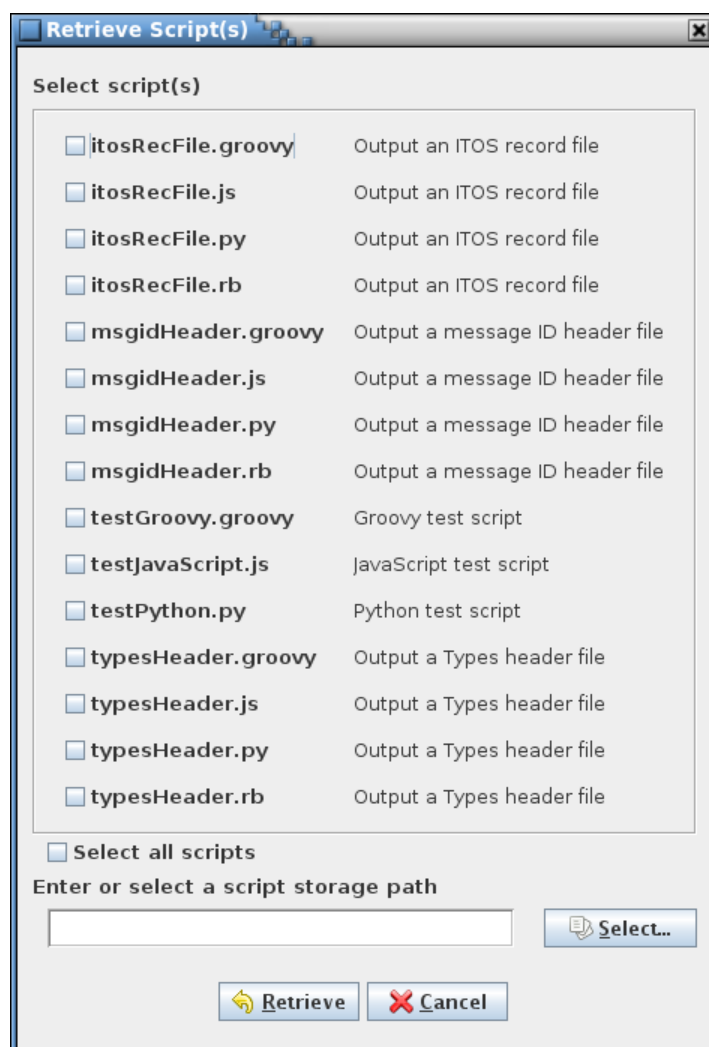


Figure 57. Retrieve Script(s) dialog

The dialog displays a list of the stored script (or other text) files. Using the check boxes, select one or more files. The “Select all scripts” check box is used to alternately select and deselect all of the individual script check boxes. Choose a folder in which to save the retrieved file(s), then press the **Retrieve** button to extract a copy of the file(s) from the project into the selected folder. Select the **Cancel** button to exit dialog without retrieving any files.

When a file is stored the application first searches it for the first line containing the text “description:”. The search ignores case, so any combination of upper and lower case characters constitutes a match. If found, the remaining text on the same line in the file (sans any leading or trailing white space character(s)) is stored with the file as its description. This is the description that appears alongside the file name in the dialog. If no match is found then the description text in the dialog is blank.

#### 4.10.5.5 Delete

Selecting the **Delete** command causes the Delete Script(s) dialog to appear (Figure 58).

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 98 of 166 |

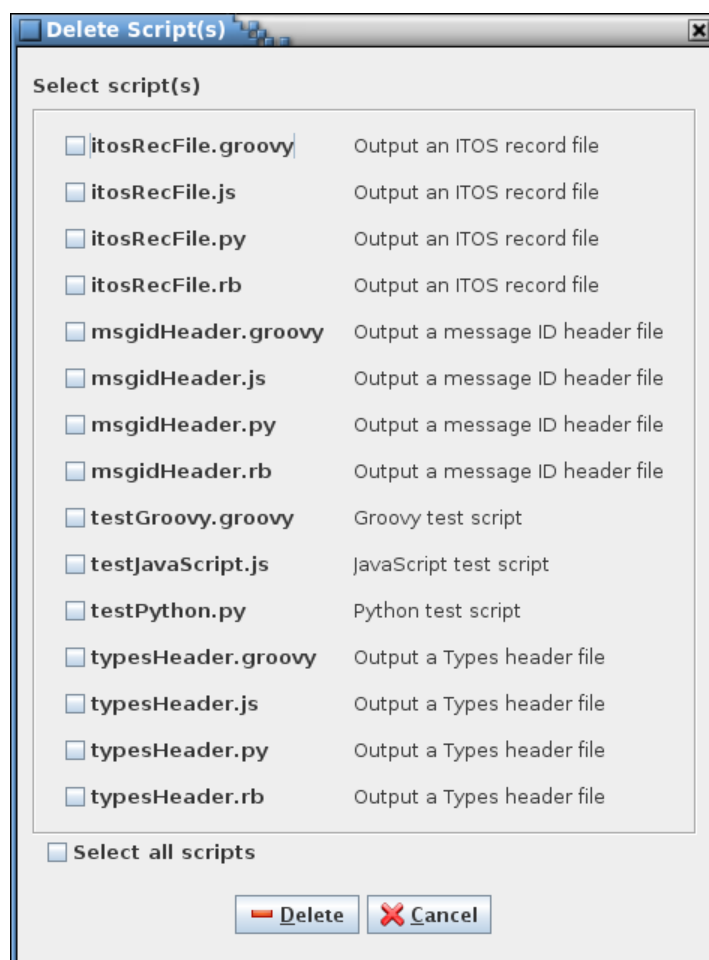


Figure 58. Delete Script(s) dialog

The dialog displays a list of the stored script (or other text) files. Using the check boxes, select one or more files. The “Select all scripts” check box is used to alternately select and deselect all of the individual script check boxes. Press the **Delete** button to delete the file(s) from the project. Select the **Cancel** button to exit the dialog without deleting any files.

#### 4.10.5.6 Search

The script **Search** dialog provides a means of searching the scripts stored within the project database for a specified text string (see Figure 59). Case sensitivity for the search is governed by the **Ignore text case** check box. Enter the search text in the input field and select the **Search** button. Leading and trailing white space characters are removed from the search text prior to the search. The search results are displayed in the table at the bottom of the search dialog. The first column, **Script**, shows the name of the script, and the second column, **Line number**, provides the line number in the script where a match is found. The last column, **Context**, displays the line from the script containing the search text, with the search text highlighted. Leading and trailing white space characters are removed from the context cells’ text.

The search results can be output to a file or printer by selecting the **Print** button. To exit the search dialog select the **Close** button.

|   |  |                |
|---|--|----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                |
|   | Doc. No. JSC-#####   | Baseline       |
|   | Date: January 2017   | Page 99 of 166 |

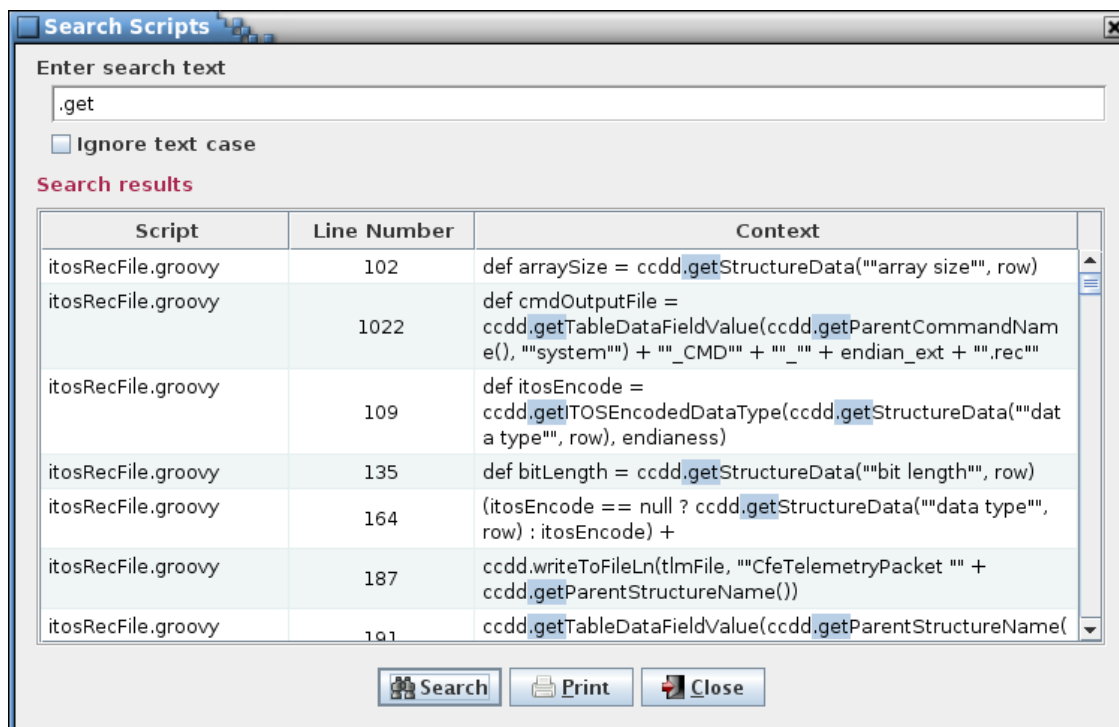


Figure 59. Script search dialog

## 4.10.6 Help menu

### 4.10.6.1 Guide

The **Guide** command displays a copy of this user's guide in PDF format.

### 4.10.6.2 About

Selecting the **About** menu item brings up an informational dialog (see Figure 60) providing the application's version information, and version numbers for the version of Java, PostgreSQL, and JDBC. Also shown is the list of available scripting languages and associated scripting engines, if any, and their respective version information.

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 100 of 166 |

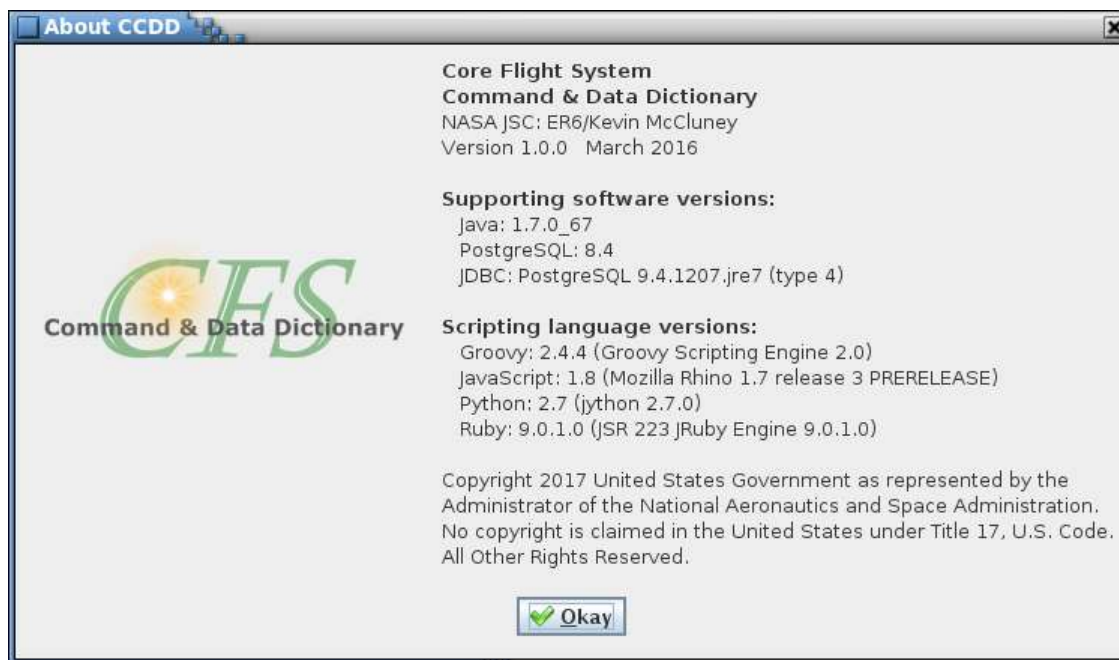


Figure 60. About dialog

## 4.11 Scripts

The CCDD application's script interface is the mean by which a project's data, stored in the database, is made available for manipulation by the user, primarily for formatting the data to create output files. CCDD supports the use of JVM-based scripting languages. Four of these languages, JavaScript, Python, Ruby, and Groovy, have been tested with the CCDD application, though any of the others should work as well. A language must be installed before it can be used by CCDD. The About dialog (see paragraph 4.10.6.2 and Figure 60) displays a list of the installed scripting languages. Examples of the use of scripts to produce output files include creation of:

- C header files for CFS applications
- CFS Housekeeping copy table
- ITOS record and display files

Scripts may be executed from within the application (see paragraphs 4.10.5.1 and 4.10.5.2) or from the command line (see Table 1).

The scripts have access to the database data via a set of script data access methods written in Java. Additional methods are provided for displaying dialog boxes (both output and input), writing to an output file, and making direct queries to the database. The methods are accessible using the class name **ccdd**:

```
ccdd.methodName(arguments...)
```

where *methodName* is the name of the script data access method (function) and *arguments...* are the parameters required by the particular method. Details on the script data access methods are provided in Table 7.

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 101 of 166 |

In order to access these methods the script requires that the data access class be imported; the import statement is dependent on the scripting language. The following paragraphs show the import statement required to be included in the script file for each of the tested scripting languages, as well as an example of using the script data access methods. For each scripting language the example accomplishes the same result and assumes one or more structure tables are associated with the script (see paragraph 4.10.5.1 for information on associating scripts with data tables). First, the script opens an output file names "myFileName". Then the names of the structures present in the structure table(s) supplied to the script are stored in an array named "structNames". A loop is then performed to write each structure's name to the output file. Finally, the output file is closed and the script terminates, returning control to the CCDD application. A status message is written to the event log to indicate script completion.

#### 4.11.1 JavaScript

JavaScript script files must end with the extension ".js". The JavaScript script must contain the following lines at or near the top of the file (this allows the script to work with both JavaScript 'Rhino' (Java 7 and earlier) and 'Nashorn' (Java 8 and later)):

```
try
{
    load("nashorn:mozilla_compat.js");
}
catch (e)
{
}
importClass(Packages.CCDD.CcddScriptDataAccessHandler);
```

Example code:

```
// Import the script data access method class
try
{
    load("nashorn:mozilla_compat.js");
}
catch (e)
{
}
importClass(Packages.CCDD.CcddScriptDataAccessHandler);

// Open the output file
var file = ccdd.openOutputFile("myFileName");

// Get the array of structure names
var structNames = ccdd.getStructureNames();

// Step through each name found
for (var index = 0; index < structNames.length; index++)
{
    // Write the structure name to the output file
    ccdd.writeToFileLn(file,
        "structNames["
        + index
        + "] = "
        + structNames[index]);
}
```

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 102 of 166 |

```
}

// Close the output file
ccdd.closeFile(file);
```

### 4.11.2 Python

Python script files must end with the extension “.py”. The Python script must contain the following line at or near the top of the file:

```
from CCDD import CcddScriptDataAccessHandler
```

Example code:

```
# Import the script data access method class
from CCDD import CcddScriptDataAccessHandler

# Open the output file
file = ccdd.openOutputFile("myFileName")

# Get the array of structure names
structNames = ccdd.getStructureNames()

# Step through each name found
for index in range(len(structNames)):

    # Write the structure name to the output file
    ccdd.writeToFileLn(file, "structNames[" + str(index) + "] = " +
        structNames[index])

# Close the output file
ccdd.closeFile(file)
```

### 4.11.3 Ruby

Ruby script files must end with the extension “.rb”. The Ruby script must contain the following line at or near the top of the file:

```
java_import Java::CCDD.CcddScriptDataAccessHandler
```

Example code:

```
# Import the script data access method class
java_import Java::CCDD.CcddScriptDataAccessHandler

# Open the output file
file = $ccdd.openOutputFile("myFileName")

# Get the array of structure names
structNames = $ccdd.getStructureNames()

index = 0

# Step through each structure name
structNames.each do |name|

    # Write the structure name to the output file
    $ccdd.writeToFileLn(file, "structNames[#{index}] = #{name}")
```

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. <i>JSC-#####</i>  | <i>Baseline</i> |
|   | Date: <i>January 2017</i>  | Page 103 of 166 |

```

        index += 1

    end

    # Close the output file
    $ccdd.closeFile(file)

```

#### 4.11.4 Groovy

Groovy script files must end with the extension “.groovy”. The Groovy script must contain the following line at or near the top of the file:

```
import CCDD.CcddScriptDataAccessHandler
```

Example code:

```

// Import the script data access method class
import CCDD.CcddScriptDataAccessHandler

// Open the output file
def file = ccdd.openOutputFile("myFileName")

// Get the array of structure names
def structNames = ccdd.getStructureNames()

// Step through each name found
for (def index = 0; index < structNames.length; index++)
{
    // Write the structure name to the output file
    ccdd.writeToFileLn(file,
        "structNames[" +
        index +
        "] = " +
        structNames[index])
}

// Close the output file
ccdd.closeFile(file)

```

#### 4.11.5 Command line execution

The CCDD command line option, `execute`, allows running scripts without use of the GUI. The script file and data table association must be specified on the command line. The command format is:

```
<script_name[:table[+table2[+...[+tableN]]][,...]]>
```

The project database, host, user, and password (if required) command line options must be specified, prior to the `execute` option, in order to access the project's database. If not specified, the last project database, user, and host accessed by the application in the most recent session is used. The script name must include its file path if the script is not located within the folder from which the CCDD application is executed. If multiple scripts are provided then these are run serially in the order specified.

Even though the GUI is not displayed, the event log is generated and all events (success, fail, command, and status events) are written to the log file. Information, warning, and error dialogs are not displayed; instead the text for these dialogs is output to the standard output (information) and standard error

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 104 of 166 |

(warning and error) streams. Dialogs requiring user input, however, are displayed, and script execution pauses until the dialog is dealt with.

When script execution completes the CCDD application terminates. The application returns a status indicating if the scripts executed successfully: 0 if all script execution succeeded, or 1 if any script did not complete successfully.

Following are examples of running scripts from the command line. Note that in these examples CCDD is an alias that executes the application with all the necessary class paths, etc. (see paragraph 4.1). The first example demonstrates executing the script *myScript* with no associated tables:

```
CCDD -project myProject -host localhost -user myUser -password myPassword
      -execute myScript
```

The next example executes *myScript* using the data from the table *myTable* (and its child tables, if applicable):

```
CCDD -project myProject -host 192.168.1.1 -port 5432 -user myUser -
      password myPassword -execute myScript:myTable
```

The third example executes *myScript* using the data from the tables *myTable1* and *myTable2* (and their child tables, if applicable):

```
CCDD -project myProject -user myUser -password myPassword -execute
      myScript:myTable1+myTable2
```

The last example executes *myScript1* using the data from the tables *myTable1* and *myTable2* (and their child tables, if applicable), then executes the script *myScript2* using the data from the table *myTable3* (and its child tables, if applicable):

```
CCDD -password myPassword -execute
      myScript1:myTable1+myTable2,myScript2:myTable3
```

#### 4.11.6 Data access methods

Table 7 provides details on each of the project data access methods available for use in the scripts. The first column is the method name. When calling the method from a script the method name must be preceded by `ccdd.` (or `$ccdd.` for Ruby scripts). The second column is a short description of the access method. The third column in the table gives the method input parameter type(s) and description(s), if any. The fourth column gives the output type and description, if any.

Certain methods require that the table type be supplied as a parameter. Convenience methods are provided in these cases for the Structure and Command table types. In place of supplying the table type as a parameter the method name incorporates the table type. For example, the method `getTableData()` has accompanying convenience methods `getCommandData()` and `getStructureData()`.



|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 105 of 166 |

| Method Name                     | Description   | Input(s)  | Output   |
|---------------------------------|---|---|--|
| closeFile                       | Close the specified output file   | PrintWriter: Output file<br>PrintWriter object obtained from the openOutputFile method  |  |
| createApplicationSchedulerTable | Create the application scheduler table  |   |  |
| getApplicationCommandTable      | Get the application scheduler command table   |   | String[]: Array containing the command table information   |
| getApplicationNames             | Get the array containing the groups that represent CFS applications   |   | String[]: Array containing names of the groups that represent CFS applications   |
| getApplicationSchedulerEntry    | Get the list of defines for the scheduler table   | int:  | String[][]: Array containing the defines list  |
| getApplicationSchedulerGroups   | Get the application scheduler groups  |   | String[]: Array containing the application scheduler groups  |
| getArrayFromString              | Divide the supplied string into a two-dimensional array (columns and rows) using the supplied separator characters or strings, and trim any leading or trailing white space characters from each array member | String: String to separate into an array<br>String: Character string to use to delineate the separation point(s) between columns. The separator is eliminated from the array members<br>String: Character string to use to delineate the separation point(s) between rows. The separator is eliminated from the array members. Use null if only one row is supplied | String[][]: Two-dimensional array representing the substrings in the supplied text after being parsed using the separator; returns null if the input text is empty |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 106 of 166 |

| Method Name                    | Description   | Input(s)   | Output   |
|--------------------------------|---|--|--|
| getArrayFromString             | Divide the supplied string into an array using the supplied separator character or string, and trim any leading or trailing white space characters from each array member   | String: String to separate into an array<br>String: Character string to use to delineate the separation point(s) between columns. The separator is eliminated from the array members | String[]: Array representing the substrings in the supplied text after being parsed using the separator; returns null if the input text is empty   |
| getCheckBoxDialog              | Display a dialog containing one or more check boxes. The user must press the Okay button to accept the check box input(s), or Cancel to close the dialog without accepting the input  | String: Text to display above the check box(es)<br>String[][]: Array containing the text and optional descriptions for the radio buttons to display in the dialog                    | boolean[]: An array containing the status for the check box(es) if the Okay button is pressed; returns null if no check box information is supplied or if the Cancel button is pressed                               |
| getCommandTableData            | Get the command table data at the row and column indicated, with any macro replaced by its corresponding value. The column is specified by name and is not case sensitive. Convenience method for getTableData that assumes the table type is "command" | String: Table column name (case insensitive)<br>int: Index of the row in the table data  | String: Contents of the specified command table's array at the row and column name provided, with any macro replaced by its corresponding value; returns null if an instance of the command table type doesn't exist |
| getCommandTableDataFieldValues | Get the data field value for all command tables that have the specified data field  | String: Data field name  | String: Array of command table names and the data field value; returns an empty array if the field name is invalid (i.e., no command table has the data field)   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 107 of 166 |

| Method Name                   | Description   | Input(s)  | Output   |
|-------------------------------|---|---|--|
| getCommandTableDataWithMacros | Get the command table data at the row and column indicated, with any macro name(s) left in place. The column is specified by name and is not case sensitive. Convenience method for getTableDataWithMacros that assumes the table type is "command" | String: Table column name (case insensitive)<br>int: Index of the row in the table data | String: Contents of the specified command table's array at the row and column name provided, with any macro name(s) left in place; returns null if an instance of the command table type doesn't exist |
| getCommandTableNameByRow      | Get the command table name to which the specified row's data belongs. Convenience method for getTableNameByRow that assumes the table type is "command"   | int: Index of the row in the table data   | String: Command table name to which the current row's parameter belongs; returns a blank if an instance of the command table type or the row doesn't exist   |
| getCommandTableNames          | Get the array of all command table names in the table data. Convenience method for getTableNames that specifies the table type as "command"   |   | String[]: Array of all command table names referenced in the table data; returns an empty array if an instance of the command table type doesn't exist   |
| getCommandTableNumRows        | Get the number of rows of data in the command table. Convenience method for getTableNumRows that assumes the table type is "command"  |   | int: Number of rows of data in the table of the type "command"; return -1 if an instance of the command table type doesn't exist   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 108 of 166 |

| Method Name             | Description  | Input(s)   | Output   |
|-------------------------|--|--|--|
| getCommandTypeNameByRow | Get the table type name referenced in the specified row of the command table type data. Convenience method that specifies the table type as "command". The data for all command types are combined. This method provides the means to retrieve the specific table type to which the row data belongs | int: Index of the row in the table data  | String: Command table type name to which the current row's parameter belongs; returns a blank if an instance of the command table type or the row doesn't exist                                    |
| getCopyTableColumnNames | Get the copy table column names  |  | String[]: Array containing the copy table column names   |
| getCopyTableEntries     | Get the copy table for the messages of the specified data stream   | String: data stream name<br>int: size of the message header in bytes. For example, the CCSDS header size is 12<br>String: name of the message ID name data field (e.g., 'Message ID name')<br>boolean: true to combine memory copy calls for consecutive variables in the copy table | String[][]: Array containing the copy table entries; returns blank if there are no entries for the specified data stream or if data stream name is invalid   |
| getDatabaseQuery        | Perform a query on the currently open database   | String: PostgreSQL-compatible database query statement   | String[][]: Two-dimensional array representing the rows and columns of data returned by the database query; returns null if the query produces an error, or an empty array if there are no results |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 109 of 166 |

| Method Name            | Description  | Input(s)                                 | Output   |
|------------------------|--|--|--|
| getDataStreamNames     | Get a string array containing all of the data stream names in the project  |  | String[]: Array containing the unique data stream names  |
| getDataTypeDefinitions | Get the array containing the user-defined data type names and their corresponding C-language, size (in bytes), and base data type values |  | String[][]: Array where each row contains a user-defined data type name and its corresponding C-language, size (in bytes), and base data type values |
| getDataTypeSizeInBytes | Get the number of bytes for the specified data type  | String: structure or primitive data type | int: Number of bytes required to store the data type; returns 0 if the data type doesn't exist   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 110 of 166 |

| Method Name    | Description   | Input(s) | Output   |
|----------------|---|----------|--|
| getDateAndTime | <p>Get the current time and date in the form:</p> <p><i>dow mon dd hh:mm:ss zzz yyyy</i></p> <p>where:</p> <p><i>dow</i> is the day of the week (Sun, Mon, Tue, Wed, Thu, Fri, Sat)</p> <p><i>mon</i> is the month (Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec)</p> <p><i>dd</i> is the day of the month (01 through 31), as two decimal digits</p> <p><i>hh</i> is the hour of the day (00 through 23), as two decimal digits</p> <p><i>mm</i> is the minute within the hour (00 through 59), as two decimal digits</p> <p><i>ss</i> is the second within the minute (00 through 61, as two decimal digits)</p> <p><i>zzz</i> is the time zone (and may reflect daylight saving time)</p> <p><i>yyyy</i> is the year, as four decimal digits</p> |          | String: Current date and time                  |
| getDefinesList | Get the schedule groups   |          | String[]: Array containing the schedule groups |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 111 of 166 |

| Method Name                  | Description   | Input(s)   | Output  |
|------------------------------|---|--|---|
| getFullVariableName          | Get a variable's full name which includes the variables in the structure path separated by underscores  | int: Index of the row in the table data  | String: The variable name for the specified row prepended with each structure variable in its path, separated by underscores  |
| getGroupDataFieldDescription | Get the description of the data field for the specified group's specified data field  | String: Group name<br>String: Data field name  | String: Data field's description; returns a blank if the group name or data field name is invalid   |
| getGroupDataFieldValue       | Get the contents of the data field for the specified table's specified data field   | String: Table name, including the path if this table references a structure<br>String: Data field name | String: Data field's value; returns a blank if the table type, table name, or data field name is invalid  |
| getGroupDescription          | Get the description for the specified group   | String: group name   | String: Description for the specified group; blank if the group has no description or the group doesn't exist   |
| getGroupNames                | Get an array of all group names   | boolean: true to get only the groups that represent a CFS application; false to get all groups         | String[]: Array containing the group names (application groups only if the input flag is true); returns an empty array if no groups exist                               |
| getInputDialog               | Display a dialog for receiving text input. The user must select Okay to accept the input, or Cancel to close the dialog without accepting the input | String: Text label to display beside the input text field  | String: The text entered in the dialog input field if the Okay button is pressed; returns null if no text or white space is entered, or if the Cancel button is pressed |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 112 of 166 |

| Method Name             | Description   | Input(s)   | Output  |
|-------------------------|---|--|---|
| getITOSEncodedDataType  | Convert a primitive data type into its ITOS encoded form  | String: data type; e.g., uint16, double<br><br>String: ITOS encoding type:<br><b>SINGLE_CHAR</b> to get the single character encoding (e.g., "I" for any integer type)<br><b>BIG_ENDIAN</b> to get the encoding as big endian<br><b>BIG_ENDIAN_SWAP</b> to get the encoding as a big endian with byte swapping<br><b>LITTLE_ENDIAN</b> to get the encoding as little endian<br><b>LITTLE_ENDIAN_SWAP</b> to get the encoding as a little endian with byte swapping | String: ITOS encoded form of the data type in the format requested; null if the data type is not recognized. Example: a data type of "int32" and ITOS encoding type of <b>LITTLE_ENDIAN</b> returns "I12345678" |
| getITOSLimitName        | Get the ITOS limit name based on the supplied index value   | int: 0 = redLow, 1 = yellowLow, 2 = yellowHigh, 3 = redHigh  | String: ITOS limit name ("redLow", "yellowLow", "yellowHigh", or "redHigh"); returns blank if the index is invalid  |
| getLinkApplicationNames | Get the array containing the application name data field values associated with the specified link's variable members. Each application name is listed only once in the array | String: Name of the application name data field  | String[]: Array containing the contents of the specified application name data field associated with each of the tables referenced by the link's variable members   |



|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 113 of 166 |

| Method Name         | Description   | Input(s)  | Output  |
|---------------------|---|---|---|
| getLinkDescription  | Return the description for the specified link; returns a blank if the link doesn't exist or the link has no description | String: Data stream name<br>String: Link name   | String: Link description; returns a blank if the data stream or link don't exist, or the link has no description  |
| getLinkRate         | Return the sample rate for the specified link; returns a blank if the link doesn't exist                                | String: Data stream name<br>String: Link name   | String: Text representation of the sample rate, in samples per second, of the specified link. For rates equal to or faster than 1 sample per second the string represents a whole number; for rates slower than 1 sample per second the string is in the form <i>number of samples / number of seconds</i> ; returns a blank if the data stream or link don't exist |
| getLongestString    | Get the character length of the longest string in the supplied string array   | String[]: array of strings<br>Integer: initial minimum width; null to use zero as the minimum             | int: character length of the longest string in the supplied string array  |
| getLongestStrings   | Get the character length of the longest string for each column in the supplied string array                             | String[][]: array of string arrays<br>Integer[]: initial minimum widths; null to use zero as the minimums | Integer[]: character lengths of the longest string in each column of the supplied string array  |
| getMacroDefinitions | Get the array containing the macro names and their corresponding values   |   | String[][]: Array where each row contains a macro names and its corresponding value   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 114 of 166 |

| Method Name                | Description   | Input(s) | Output   |
|----------------------------|---|----------|--|
| getNumberOfSlots           | Get the number of time slots for the scheduler table  |          | int: Number of time slots for the command table  |
| getParentCommandTableName  | Get the name of the parent command table. If multiple parent command tables are combined in the table data then the first parent name (alphabetically) is returned (use the getParentCommandTableNames method to retrieve the names of all parent command tables in the table data). Convenience method for getParentTableName that assumes the table type is "command" |          | String: Parent command table's name; returns a blank if an instance of the command table type doesn't exist                              |
| getParentCommandTableNames | Get the name(s) of the parent command table(s). Convenience method for getParentTableNames that assumes the table type is "command". If only a single parent command table is expected then the getParentCommandTableName method can be used instead  |          | String[]: Array containing the parent command table names; returns an empty array if an instance of the command table type doesn't exist |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 115 of 166 |

| Method Name                  | Description   | Input(s)   | Output   |
|------------------------------|---|--|--|
| getParentStructureTableName  | Get the name of the parent structure table. If multiple parent structure tables are combined in the table data then the first parent name (alphabetically) is returned (use the getParentStructureTableNames method to retrieve the names of all parent structure tables in the table data). Convenience method for getParentTableName that assumes the table type is "structure" |  | String: Parent structure table's name; returns a blank if an instance of the structure table type doesn't exist                              |
| getParentStructureTableNames | Get the name(s) of the parent structure table(s). Convenience method for getParentTableNames that assumes the table type is "structure". If only a single parent structure table is expected then the getParentStructureTableName method can be used instead  |  | String[]: Array containing the parent structure table names; returns an empty array if an instance of the structure table type doesn't exist |
| getParentTableName           | Get the name of the parent table for the supplied table type. If multiple parent tables for the supplied type are combined in the table data then the first parent name (alphabetically) is returned (use the getParentTableNames method to retrieve the names of all parent tables of a specified type in the table data)  | String: Table type; e.g., "structure", "command", etc. | String: Parent table's name for the type specified; returns a blank if an instance of the table type doesn't exist                           |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 116 of 166 |

| Method Name         | Description  | Input(s)  | Output  |
|---------------------|--|---|---|
| getParentTableNames | Get the name(s) of the parent table(s) for the supplied table type. If only a single parent table is expected then the getParentTableName method can be used instead | String: Table type; e.g., "structure", "command", etc.  | String[]: Array containing the parent table names for the type specified; returns an empty array if an instance of the table type doesn't exist   |
| getPathByRow        | Get the path to which the specified row's data belongs   | String: Table type; e.g., "structure", "command", etc.<br>int: Index of the row in the table data | String: The path to the current row's parameter; returns a blank if an instance of the table type doesn't exist. The path starts with the top-level table name. For structure tables the top-level name is followed by a comma and then the parent structure and variable name(s) that define(s) the table's path. Each parent and its associated variable name are separated by a period. Each parent/variable pair in the path is separated by a comma. The format is:<br><br><i>top-level&lt;,variable1.parent1&lt;,variable2.parent2&lt;...&gt;&gt;&gt;</i> |
| getProject          | Get the name of the project database   |   | String: Name of the project database  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 117 of 166 |

| Method Name                              | Description   | Input(s)   | Output  |
|--|---|--|---|
| getRadioButtonDialog                     | Display a dialog containing radio buttons. The radio buttons are mutually exclusive; only one can be selected at a time. The user must press the Okay button to accept the radio button input, or Cancel to close the dialog without accepting the input                                  | String: Text to display above the radio buttons<br>String[][]: Array containing the text and optional descriptions for the radio buttons to display in the dialog        | String: The text for the selected radio button if the Okay button is pressed; returns null if no radio button is selected or if the Cancel button is pressed  |
| getScriptName                            | Get the name of the script file being executed  |  | String: Script file name  |
| getStructureDataByVariableName           | Get the data from the specified "Structure" table in the specified column for the row with the specified variable name, with any macro replaced by its corresponding value. Convenience method that assumes the table type is "Structure" and the variable name column is "Variable Name" | String: full table path, which includes the parent table name and the data type + variable name pairs<br>String: variable name<br>String: column name (case insensitive) | String: Contents of the table defined by the table path, variable name, and column name specified; returns null if an instance of the table type, the column name, or the variable name doesn't exist                                       |
| getStructureDataByVariableNameWithMacros | Get the data from the specified "Structure" table in the specified column for the row with the specified variable name, with any macro name(s) left in place. Convenience method that assumes the table type is "Structure" and the variable name column is "Variable Name"               | String: full table path, which includes the parent table name and the data type + variable name pairs<br>String: variable name<br>String: column name (case insensitive) | String: Contents of the table defined by the table path, variable name, and column name specified, with any macro name(s) left in place; returns null if an instance of the table type, the column name, or the variable name doesn't exist |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 118 of 166 |

| Method Name                      | Description   | Input(s)  | Output   |
|----------------------------------|---|---|--|
| getStructuresByReferenceOrder    | Get an array containing the names of the structures in the order in which they are referenced; that is, the structure array is arranged so that a child structure appears in the array prior to the parent structure(s) that reference it                   |   | String[]: Array containing the names of the structures in the order in which they are referenced   |
| getStructureTableData            | Get the structure table data at the row and column indicated, with any macro replaced by its corresponding value. The column is specified by name and is not case sensitive. Convenience method for getTableData that assumes the table type is "structure" | String: Table column name (case insensitive)<br>int: Index of the row in the table data | String: Contents of the specified structure table's array at the row and column name provided, with any macro replaced by its corresponding value; returns null if an instance of the structure table type doesn't exist |
| getStructureTableDataFieldValues | Get the data field value for all structure tables that have the specified data field  | String: Data field name   | String: Array of structure table names and the data field value; returns an empty array if the field name is invalid (i.e., no structure table has the data field)   |
| getStructureTableDataWithMacros  | Get the structure table data at the row and column indicated, with any macro name(s) left in place. The column is specified by name and is not case sensitive. Convenience method for getTableDataWithMacros that assumes the table type is "structure"     | String: Table column name (case insensitive)<br>int: Index of the row in the table data | String: Contents of the specified structure table's array at the row and column name provided, with any macro name(s) left in place; returns null if an instance of the structure table type doesn't exist               |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 119 of 166 |

| Method Name                    | Description   | Input(s)                                | Output   |
|--------------------------------|---|---|--|
| getStructureTableITOSPathByRow | Get the structure path to which the specified row's data belongs, formatted for use in an ITOS record statement   | int: Index of the row in the table data | String: The path to the current row's parameter formatted for use in an ITOS record statement; returns a blank if an instance of the table type doesn't exist. The path starts with the top-level table name. The top-level name is followed by a period and then the variable name(s) that define(s) the table's path. Each variable in the path is separated by an underscore. The format is:<br><br><i>top-level&lt;.variable1_parent1&lt;.variable2_parent2&lt;...&gt;&gt;&gt;</i> |
| getStructureTableNameByRow     | Get the structure table name to which the specified row's data belongs. Convenience method for getTableNameByRow that assumes the table type is "structure" | int: Index of the row in the table data | String: Structure table name to which the current row's parameter belongs; returns a blank if an instance of the structure table type or the row doesn't exist   |
| getStructureTableNames         | Get the array of all structure table names in the table data. Convenience method for getTableNames that specifies the table type as "structure"             |   | String[]: Array of all structure table names referenced in the table data; returns an empty array if an instance of the structure table type doesn't exist   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 120 of 166 |

| Method Name                        | Description  | Input(s)                                | Output  |
|------------------------------------|--|---|---|
| getStructureTableNumRows           | Get the number of rows of data in the structure table. Convenience method for getTableNumRows that assumes the table type is "structure"   |   | int: Number of rows of data in the table of the type "structure"; return -1 if an instance of the structure table type doesn't exist  |
| getStructureTableVariablePathByRow | Get the structure path to which the specified row's data belongs, showing only the top-level structure and variable names. This format is used when referencing a structure table's data fields  | int: Index of the row in the table data | String: The path to the current row's parameter; returns a blank if an instance of the table type doesn't exist. The path starts with the top-level table name. The top-level name is followed by a comma and then the variable name(s) that define(s) the table's path. Each variable in the path is separated by a comma. The format is:<br><br><i>top-level&lt;,variable1&lt;,variable2&lt;...&gt;&gt;&gt;</i> |
| getStructureTypeNameByRow          | Get the table type name referenced in the specified row of the structure table type data. Convenience method that specifies the table type as "structure". The data for all structure types are combined. This method provides the means to retrieve the specific table type to which the row data belongs | int: Index of the row in the table data | String: Structure table type name to which the current row's parameter belongs; returns a blank if an instance of the structure table type or the row doesn't exist   |



|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 121 of 166 |

| Method Name             | Description   | Input(s)  | Output   |
|-------------------------|---|---|--|
| getTableData            | Get the data at the row and column indicated, with any macro replaced by its corresponding value, for the table type specified. The column is specified by name | String: Table type; e.g., "structure", "command", etc.<br>String: Table column name (case insensitive)<br>int: Index of the row in the table data | String: Contents of the specified table's array at the row and column name provided with any macro replaced by its corresponding value; returns null if an instance of the table type, the column name, or the row doesn't exist |
| getTableDataFieldValues | Get the data field value for all tables that have the specified data field  | String: Data field name   | String: Array of table names and the data field value; returns an empty array if the field name is invalid (i.e., no table has the data field)   |
| getTableDataFieldValues | Get the data field value for all tables of the specified type that have the specified data field  | String: Table type; e.g., "structure", "command", etc.<br>String: Data field name   | String: Array of table names of the specified type and the data field value; returns an empty array if the field name is invalid (i.e., no table has the data field)   |
| getTableDataWithMacros  | Get the table data at the row and column indicated, with any macro name(s) left in place. The column is specified by name and is not case sensitive             | String: Table type; e.g., "structure", "command", etc.<br>String: Table column name (case insensitive)<br>int: Index of the row in the table data | String: Contents of the specified table's array at the row and column name provided, with any macro name(s) left in place; returns null if an instance of the table type, the column name, or the row doesn't exist              |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 122 of 166 |

| Method Name              | Description  | Input(s)   | Output   |
|--------------------------|--|--|--|
| getTableDataByColumnName | Get the data from the a table in the specified column for the row in the matching column name that contains the matching name, with any macro name replaced by its corresponding value | String: table type<br>String: full table path<br>String: name of the column containing that matching name (case insensitive)<br>String text to match in the matching column - this determines the row. The first row in the matching column that matches the matching name determines the row used to retrieve the data value<br>String: name of the column from which to retrieve the data value (case insensitive) | String: Contents of the table defined by the table type, table path, matching column name, matching name, and data column name specified, with any macro name replaced by its corresponding value; returns null if an instance of the table type, the matching column, the data column, or the matching name doesn't exist |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 123 of 166 |

| Method Name                        | Description   | Input(s)   | Output  |
|------------------------------------|---|--|---|
| getTableDataByColumnNameWithMacros | Get the data from the a table in the specified column for the row in the matching column name that contains the matching name, with any macro name(s) left in place | String: table type<br>String: full table path<br>String: name of the column containing that matching name (case insensitive)<br>String text to match in the matching column - this determines the row. The first row in the matching column that matches the matching name determines the row used to retrieve the data value<br>String: name of the column from which to retrieve the data value (case insensitive) | String: Contents of the table defined by the table type, table path, matching column name, matching name, and data column name specified, with any macro name(s) left in place; returns null if an instance of the table type, the matching column, the data column, or the matching name doesn't exist |
| getTableDataFieldDescription       | Get the description of the data field for the specified table's specified data field  | String: Table name, including the path if this table references a structure<br>String: Data field name   | String: Data field's description; returns a blank if the table name or data field name is invalid   |
| getTableDataFieldValue             | Get the contents of the data field for the specified table's specified data field   | String: Table name, including the path if this table references a structure<br>String: Data field name   | String: Data field's value; returns a blank if the table name or data field name is invalid   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 124 of 166 |

| Method Name              | Description  | Input(s)  | Output  |
|--------------------------|--|---|---|
| getTableDescriptionByRow | Get the description of the table at the row indicated for the table type specified       | String: Table type; e.g., "structure", "command", etc.<br>int: Index of the row in the table data | String: Table name for the specified table type to which the current row's parameter belongs; returns a blank if an instance of the table type or the row doesn't exist |
| getTableNameByRow        | Get the table name for the type specified to which the specified row's parameter belongs | String: Table type; e.g., "structure", "command", etc.<br>int: Index of the row in the table data | String: Table name for the specified table type to which the current row's parameter belongs; returns a blank if an instance of the table type or the row doesn't exist |
| getTableNames            | Get an array of all tables referenced in the table data for all table types              |   | String[]: Array of all table names referenced in the table data; empty array if no tables exists in the data  |
| getTableNames            | Get an array of all tables referenced in the table data of the specified table type      | String: Table type; e.g., "structure", "command", etc.  | String[]: Array of all table names represented by the table type; returns an empty array if an instance of the table type doesn't exist                                 |
| getTableNumRows          | Get the number of rows of data in the table of the type specified                        | String: Table type; e.g., "structure", "command", etc.  | int: Number of rows of data in the table of the type specified; return -1 if an instance of the table type doesn't exist  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 125 of 166 |

| Method Name                 | Description   | Input(s)   | Output   |
|-----------------------------|---|--|--|
| getTelemetryMessageIDs      | Get the messages ID names and their corresponding ID values for the specified data stream   | String: data stream name   | String: Array containing the message ID names and ID values; returns blank if there are no entries for the specified data stream or if data stream name is invalid |
| getTypeDataFieldDescription | Get the description of the data field for the specified table type's specified data field   | String: Table type name<br>String: Data field name   | String: Data field's description; returns a blank if the table type name or data field name is invalid   |
| getTypeDataFieldValue       | Get the contents of the data field for the specified table type's specified data field  | String: Table type name<br>String: Data field name   | String: Data field's value; returns a blank if the table type name or data field name is invalid   |
| getTypeNameByRow            | Get the table type name referenced in the specified row of the specified table type data. Multiple structure (and command) types are allowed. The data for all structure (command) types are combined. This method provides the means to retrieve the specific table type to which the row data belongs | String: Table type name. All structure table types are combined and are referenced by the type name "Structure", and all command table types are combined and are referenced by the type name "Command"<br><br>int: Index of the row in the table data | String: Table type name to which the current row's parameter belongs; returns a blank if an instance of the table type or the row doesn't exist                    |
| getUser                     | Get the name of the user executing the script   |  | String: Name of the user executing the script  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 126 of 166 |

| Method Name         | Description  | Input(s)  | Output  |
|---------------------|--|---|---|
| getVariableLinks    | Get the array of link names to which the specified variable belongs  | String: variable path and name  | String[]: Array containing the links to which the specified variable is a member; returns an empty array if the variable does not belong to a link  |
| getVariableOffset   | Get the byte offset of the specified variable relative to its parent structure. The variable's path, including parent structure and variable name, is used to verify that the specified target has been located; i.e., not another variable with the same name | String: Parent structure name of the variable being checked<br>String: A comma separated string of each data type and variable name of each variable in the current search path | int: The byte offset to the target variable relative to its parent structure; returns -1 if the parent-variable path combination is invalid   |
| getVariablePaths    | Get an array containing the path to each parent structure and its variables  |   | String[][]: Two-dimensional array containing the path for each structure variable. The parent structures are sorted alphabetically. The variables are displayed in the order of appearance within the structure (parent or child) |
| isStructureShared   | Determine if the supplied data type is a primitive type  | String: data type   | boolean: true if the specified structure is referenced by more than one table; false otherwise  |
| isDataTypePrimitive | Determine if the specified structure is referenced by more than one parent structure   | String: name of the structure to check  | boolean: true if the supplied data type is a primitive; false otherwise   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 127 of 166 |

| Method Name           | Description  | Input(s)   | Output  |
|-----------------------|--|--|---|
| openOutputFile        | Open the specified file for writing. The PrintWriter object that is returned is used by the file writing methods to specify the output file  | String: Output file path + name  | PrintWriter: PrintWriter object; returns null if the file could not be opened |
| showErrorDialog       | Display an error dialog showing the supplied text. The dialog's header and icon indicate that the text describes an error condition. The Okay button must be pressed before the script can continue  | String: Text to display in the dialog box  |   |
| showInformationDialog | Display an informational dialog showing the supplied text. The dialog's header and icon indicate that the text describes information useful to the user; e.g., script status. The Okay button must be pressed before the script can continue | String: Text to display in the dialog box  |   |
| showWarningDialog     | Display a warning dialog showing the supplied text. The dialog's header and icon indicate that the text describes a warning condition. The Okay button must be pressed before the script can continue  | String: Text to display in the dialog box  |   |
| writeToFile           | Write the supplied text to the specified output file PrintWriter object  | PrintWriter: Output file<br>PrintWriter object obtained from the openOutputFile method<br>String: Text to write to the output file |   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 128 of 166 |

| Method Name       | Description   | Input(s)  | Output |
|-------------------|---|---|--------|
| writeToFileFormat | Write the supplied formatted text in the indicated format to the specified output file PrintWriter object | PrintWriter: Output file<br>PrintWriter object obtained from the openOutputFile method<br><br>String: Print format string to write to the output file<br><br>Object...: variable list of arguments referenced by the format specifiers in the format string |        |
| writeToFileLn     | Write the supplied text to the specified output file PrintWriter object and append a line feed character  | PrintWriter: Output file<br>PrintWriter object obtained from the openOutputFile method<br><br>String: Text to write to the output file  |        |

Table 7. Script Data Access Methods



|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. <i>JSC-#####</i>  | <i>Baseline</i> |
|   | Date: <i>January 2017</i>  | Page 129 of 166 |

## Appendix A. Acronyms

|      |                                       |
|------|---------------------------------------|
| CCDD | CFS Command & Data Dictionary         |
| cFE  | Core Flight Executive                 |
| CFS  | Core Flight System                    |
| CPU  | Central Processing Unit               |
| CSV  | comma-separated values                |
| DBU  | Database Backup                       |
| EDS  | Electronic Data Sheet                 |
| GUI  | Graphical User Interface              |
| HK   | Housekeeping                          |
| I/O  | Input/Output                          |
| ID   | Identifier                            |
| ITOS | Integrated Test and Operations System |
| JAR  | Java Archive                          |
| JDBC | Java Database Connectivity            |
| JRE  | Java Runtime Environment              |
| JSON | JavaScript Object Notation            |
| JVM  | Java Virtual Machine                  |
| L&F  | Look and Feel                         |
| OID  | Object Identifier                     |
| OS   | Operating System                      |
| PDF  | Portable Document Format              |
| PNG  | Portable Network Graphics             |
| SQL  | Structured Query Language             |
| XML  | Extensible Markup Language            |
| XTCE | XML Telemetric and Command Exchange   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 130 of 166 |

## Appendix B. Definitions

|                  |  |
|------------------|--|
| Array definition | In a structure table, the row where the variable name and array size are specified   |
| Array member     | In a structure table, the rows following the array definition row (when arrays are expanded) that display the individual variables that belong to the array. The variable name begins with the array definition's variable name and has the array index, encased in square brackets, appended. The array member rows are displayed in ascending index order, starting with a zero index. The array size column for each member row displays the total number of members in the array   |
| Child table      | A structure table that is referenced as a data type for a variable in another structure table  |
| Database         | An collection of data within a PostgreSQL server organized as tables. A CCDD <i>project</i> is a database representing the data for a CFS project  |
| Encoded type     | The byte order for primitive data types composed of two or more bytes. CCDD recognized four encodings: <ul style="list-style-type: none"> <li>little endian      bytes are store with the least significant byte first</li> <li>big endian      bytes are stores with the least significant byte last</li> <li>little endian, swap      similar to little endian, except that each byte pair is reversed; applies only to integer and unsigned integer data types composed of four bytes</li> <li>big endian, swap      similar to big endian, except that each byte pair is reversed; applies only to integer and unsigned integer data types composed of four bytes</li> </ul> |
| Instance table   | A structure table that is a child of a parent structure table  |
| Macro            | An alphanumeric string, bounded by special delimiter characters, that can be inserted into a data table cell to represent text defined by the user   |
| Parent table     | A prototype table that is not referenced from within another table. Sometimes referred to as a root table  |
| Primitive type   | A primitive data type is a basic data type (e.g., integer, float), as opposed to a structure, which is a group of primitive and/or structure data types. The primitive data types recognized by the CCDD application can be altered; see paragraph 4.6.4.  |
| Project          | Synonymous with the term <i>database</i> except when referring to the PostgreSQL default database, postgres  |
| Prototype table  | A table created via the <b>Data   New table(s)</b> command, based on one of the table types  |
| Table type       | A table template created using the Table Type Manager (see paragraph 4.10.3.6). Any number of tables may be created of a given table type  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 131 of 166 |

## Appendix C. Error & Warning Messages

The table below lists all of the error and warning messages, in alphabetical order, that can occur in the CCDD application and the causes. An error message implies that the intended operation cannot be successfully completed. An attempt is automatically made to revert any changes made to the database in the event an error occurs during a database update. If this reversion is unsuccessful then the database is likely corrupted. A command line error message results in immediate program termination, but for other errors the application continues to run. A warning message indicates that though the operation was unsuccessful the user can effect a change to correct the problem.

| Type    | Message   | Cause   |
|---------|---|---|
| Warning | # array member row(s) ignored due to missing array definition(s)                      | The number of rows indicated, #, were ignored when pasting data into a table. The cause is that one or more rows in the pasted data represent an array member, but an array definition does not precede the member(s). Include the array definition row when pasting array member information   |
| Warning | All application parameters must be entered  | An input text field is empty in the application parameter dialog. Enter a valid value in each of the fields   |
| Warning | All rate parameters must be entered   | An input text field is empty in the rate parameter dialog. Enter a valid value in each of the fields  |
| Warning | Application parameter values must be positive integer values                          | The value in one or more application parameter dialog input text fields contains a zero, negative, or non-integer value. Enter an integer value greater than or equal to 1 in each of the fields  |
| Warning | At least one data stream(s) must be selected  | No target data stream is selected in the link copy dialog when the Okay button is pressed. Choose at least one data stream or press the Cancel button   |
| Warning | Auto-fill unable to assign <i>number</i> applications                                 | The application scheduler was unable to assign <i>number</i> applications to a time slot  |
| Warning | Auto-fill unable to assign <i>number</i> variables                                    | The telemetry scheduler was unable to assign <i>number</i> variables to an output message   |
| Warning | Base data type inconsistent with data type usage in table(s) ' <i>table name(s)</i> ' | The base data type entered in the data type editor's Base Type column was changed from an integer (signed or unsigned) to a non-integer, and the indicated table(s), <i>table name(s)</i> , has a non-empty bit length or enumeration column. The associated data type for a bit length parameter or an enumerated parameter must be an integer. Clear the bit length and enumeration columns for the table(s) referencing this data type and then change the base type |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 132 of 166 |

| Type    | Message   | Cause  |
|---------|---|--|
| Warning | Bit length exceeds the size of the data type                                      | Either the bit length entered for a parameter in a data table is larger than the size, in bits, of the associated data type, or the data type changed to a smaller sized integer with a size in bits less than the current bit length. Decrease the bit length or choose a data type containing more bytes                                   |
| Warning | Bit length exceeds the size of the data type in table(s) ' <i>table name(s)</i> ' | The size entered in the data type editor's Base Type column for an integer base type (signed or unsigned) was reduced and the data type is used with parameters having a bit length specified that exceeds the capacity of the new size. Reduce or clear the bit length for the table(s) referencing this data type and then change the size |
| Warning | Cannot assign application to a time slot  | The application scheduler was unable to assign an application to a time slot when the user attempted manual assignment   |
| Warning | Cannot assign variable to a message   | The telemetry scheduler was unable to assign a variable to an output message when the user attempted manual assignment   |
| Warning | Cannot close backup file ' <i>backup file name</i> ' (file channel)               | An error occurred preventing closing the file channel used when copying the backup file <i>backup file name</i>  |
| Warning | Cannot close backup file ' <i>backup file name</i> ' (file input stream)          | An error occurred preventing closing the file input stream used when copying the backup file <i>backup file name</i>   |
| Warning | Cannot close export file ' <i>path+file name</i> '                                | The export .csv file failed to close after being written   |
| Warning | Cannot close import file ' <i>path+file name</i> '                                | The import .csv file failed to close after being read  |
| Error   | Cannot close project database ' <i>database name</i> '                            | An error occurred preventing closing project database <i>database name</i> . Detail on the cause is logged in the event log  |
| Warning | Cannot close script file ' <i>path+file name</i> '                                | The script file failed to close after being read   |
| Warning | Cannot close user's guide ' <i>file name</i> '                                    | An error occurred preventing closing the input stream used when copying the user's guide file <i>file name</i> from the CCDD .jar file   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 133 of 166 |

| Type    | Message  | Cause  |
|---------|--|--|
| Error   | Cannot connect to project database ' <i>database name</i> '          | An attempt to connect to the project database <i>database name</i> failed. Detail on the cause is logged in the event log. This can occur due to lack of access permission by the user to the database, if the selected project is already open by another instance of the CCDD application, or if the locked status flag remained set due to abnormal application termination |
| Error   | Cannot connect to server ' <i>server name</i> '                      | An attempt to connect to the default database, postgres, failed. Detail on the cause is logged in the event log. This may occur if the postgres server is not running  |
| Error   | Cannot copy project database ' <i>database name</i> '                | An error occurred preventing copying of the project database <i>database name</i> . Detail on the cause is logged in the event log   |
| Error   | Cannot copy table ' <i>table name</i> '                              | The attempt to copy table <i>table name</i> in the project database failed. Detail on the cause is logged in the event log   |
| Error   | Cannot copy table type ' <i>table type</i> '                         | The attempt to copy table type <i>table type</i> in the project database failed. Detail on the cause is logged in the event log  |
| Warning | Cannot create event log file   | The event log file cannot be created. Check that file permissions allow read/write operations to the directory in which the CCDD application was executed  |
| Error   | Cannot create export file ' <i>path+file name</i> '                  | The export .csv file <i>file name</i> cannot be created in the directory <i>path</i> . Check that the file permissions allow the user to write to this directory   |
| Error   | Cannot create functions in project database ' <i>database name</i> ' | The SQL and pgplsql functions and/or the default tables cannot be created in the project database <i>database name</i> . Detail on the cause is logged in the event log. This can occur due to lack of access permission by the user to the database   |
| Error   | Cannot create output file ' <i>path+file name</i> '                  | The output file <i>file name</i> cannot be created in the directory <i>path</i> . Check that the file permissions allow the user to write to this directory  |
| Error   | Cannot create project database ' <i>database name</i> '              | An error occurred preventing creation of the project database <i>database name</i> . Detail on the cause is logged in the event log  |
| Error   | Cannot create script file ' <i>path+file name</i> '                  | The script file <i>file name</i> cannot be created in the directory <i>path</i> . Check that the file permissions allow the user to write to this directory  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 134 of 166 |

| Type    | Message  | Cause  |
|---------|--|--|
| Error   | Cannot create tables ' <i>table names</i> '  | The attempt to create tables <i>table names</i> in the project database failed. Detail on the cause is logged in the event log   |
| Error   | Cannot create web server   | The attempt to instantiate the embedded Jetty web server failed. Detail on the cause is logged in the event log  |
| Warning | Cannot delete data type ' <i>data type</i> '; data type is referenced by a data table  | An attempt was made to delete the data type <i>data type</i> , but the data type is in use in one of the data tables. A data type can't be removed until all references to it are first eliminated. Remove the data type reference(s) and then delete the data type              |
| Warning | Cannot delete macro ' <i>macro name</i> '; macro is referenced by a data table         | An attempt was made to delete the macro <i>macro name</i> , but the macro is in use in one of the data tables. A macro can't be removed until all references to it are first eliminated. Remove the macro reference(s) and then delete the macro                                 |
| Error   | Cannot delete project database ' <i>database name</i> '                                | An error occurred preventing deletion of the project database <i>database name</i> . Detail on the cause is logged in the event log  |
| Error   | Cannot delete table type ' <i>table type</i> ' <and table(s) ' <i>table name(s)</i> '> | The attempt to delete table type <i>table type</i> and its associated table(s) <i>table name(s)</i> , if any, from the project database failed. Detail on the cause is logged in the event log   |
| Error   | Cannot delete table(s) ' <i>table name(s)</i> '  | The attempt to delete table(s) <i>table name(s)</i> in the project database failed. Detail on the cause is logged in the event log   |
| Error   | Cannot disable auto-commit   | The attempt to disable the auto-commit mode for database changes failed. If this occurs subsequent database transactions are likely to fail. Restart the application; the affected project database may require manual unlocking. Detail on the cause is logged in the event log |
| Error   | Cannot execute script ' <i>script name</i> ' using table(s) ' <i>table name(s)</i> '   | An error occurred during execution preparation of the script <i>script name</i> . Detail on the cause is logged in the event log   |
| Error   | Cannot export as EDS XML to file ' <i>file name</i> '; cause ' <i>error cause</i> '    | Exporting the Table(s) to file <i>file name</i> in EDS XML format failed due to the specified cause  |
| Error   | Cannot export as XTCE XML to file ' <i>file name</i> '; cause ' <i>error cause</i> '   | Exporting the table(s) to file <i>file name</i> in XTCE XML format failed due to the specified cause   |
| Error   | Cannot import EDS XML from file ' <i>file name</i> '; cause ' <i>error cause</i> '     | Importing the table(s) from file <i>file name</i> in EDS XML format failed due to the specified cause  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 135 of 166 |

| Type  | Message   | Cause   |
|-------|---|---|
| Error | Cannot import file ' <i>file name</i> ' into table; unrecognized file type          | Importing the data from file <i>file name</i> into a table failed due to the file type not being recognized. The file extension must end in .csv, .xtce, or .eds            |
| Error | Cannot import XTCE XML from file ' <i>file name</i> '; cause ' <i>error cause</i> ' | Importing the project from file <i>file name</i> in XTCE XML format failed due to the specified cause   |
| Error | Cannot load internal table ' <i>table name</i> '                                    | The attempt to load the data from internal table <i>table name</i> in the project database failed. Detail on the cause is logged in the event log                           |
| Error | Cannot load rate values   | The attempt to load the table paths(s) in the custom values table matching the specified rate column and sample rate failed. Detail on the cause is logged in the event log |
| Error | Cannot load table ' <i>table name</i> '   | The attempt to load the data from table <i>table name</i> in the project database failed. Detail on the cause is logged in the event log                                    |
| Error | Cannot load table members   | The attempt to load the table and child table relations failed. Detail on the cause is logged in the event log  |
| Error | Cannot locate backup file ' <i>path+file name</i> '                                 | The project database restore file <i>file name</i> cannot be found in the specified directory <i>path</i>   |
| Error | Cannot locate event log file ' <i>path+file name</i> '                              | The event log file <i>file name</i> cannot be found in the directory <i>path</i>  |
| Error | Cannot locate import file ' <i>path+file name</i> '                                 | The import .csv file <i>file name</i> cannot be found in the specified directory <i>path</i>  |
| Error | Cannot modify data in table ' <i>table name</i> '                                   | The attempt to update the contents of table <i>table name</i> in the project database failed. Detail on the cause is logged in the event log                                |
| Error | Cannot obtain column order for table ' <i>table name</i> '                          | The attempt to query the project database for the column order for table <i>table name</i> . Detail on the cause is logged in the event log                                 |
| Error | Cannot obtain comment for internal table ' <i>table name</i> '                      | The attempt to query the project database for the comment on internal table <i>table name</i> failed. Detail on the cause is logged in the event log                        |
| Error | Cannot obtain comment for project database ' <i>database name</i> '                 | The comment for the project database <i>database name</i> cannot be retrieved. Detail on the cause is logged in the event log   |
| Error | Cannot obtain comment for table ' <i>table name</i> '                               | The attempt to query the project database for the comment on table <i>table name</i> failed. Detail on the cause is logged in the event log                                 |
| Error | Cannot obtain database version number   | The database's version number cannot be obtained. Detail on the cause is logged in the event log  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 136 of 166 |

| Type    | Message   | Cause  |
|---------|---|--|
| Error   | Cannot obtain description for table ' <i>table name</i> '                       | The attempt to query the project database __values table for the description of the table <i>table name</i> failed. Detail on the cause is logged in the event log   |
| Error   | Cannot obtain JDBC version number   | The JDBC version number cannot be obtained. Detail on the cause is logged in the event log   |
| Error   | Cannot open output file ' <i>path+file name</i> '                               | The output file <i>file name</i> cannot be opened in the directory <i>path</i> . Check that the file permissions allow the user to read from this file and directory   |
| Warning | Cannot print search results; cause ' <i>error cause</i> '                       | An error occurred during an attempt to print the search results due to the cited cause   |
| Error   | Cannot read backup file ' <i>path+file name</i> '; cause ' <i>error cause</i> ' | The backup file <i>file name</i> , chosen to restore a project database, cannot be read for the reason <i>error cause</i> . Check that the file permissions allow the user so read from this file and directory  |
| Warning | Cannot read event log file  | The event log file cannot be read. Check that user has file read permissions for the file and directory  |
| Error   | Cannot read import file ' <i>path+file name</i> '                               | The import .csv file <i>file name</i> cannot be read in the directory <i>path</i> . Check that the file permissions allow the user so read this file and directory   |
| Error   | Cannot read script file ' <i>path+file name</i> '                               | The script file <i>file name</i> cannot be read in the directory <i>path</i> . Check that the file permissions allow the user so read this file and directory  |
| Error   | Cannot register database driver ' <i>driver name</i> '                          | An error occurred registering the JDBC database driver <i>driver name</i> . This can be caused by setting an invalid server type   |
| Error   | Cannot rename project database ' <i>database name</i> '                         | An error occurred preventing renaming of the project database <i>database name</i> . Detail on the cause is logged in the event log  |
| Error   | Cannot rename table ' <i>table name</i> '                                       | The attempt to rename table <i>table name</i> in the project database failed. Detail on the cause is logged in the event log   |
| Error   | Cannot rename type for table ' <i>table name</i> '                              | The attempt to rename the type for table <i>table name</i> in the project database failed. Detail on the cause is logged in the event log  |
| Error   | Cannot replace existing backup file ' <i>path+file name</i> '                   | The project database backup file <i>file name</i> already exists in the directory <i>path</i> , but cannot be removed so as to be replaced by a new backup file of the same name. Check that the file permissions allow the user to write to this file and directory |
| Warning | Cannot replace existing export file ' <i>path+file name</i> '                   | The existing file <i>path+file name</i> cannot be replaced with the EDS or XTCE export file. Check that user has file read and write permissions for the file and directory  |



|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 137 of 166 |

| Type  | Message  | Cause   |
|-------|--|---|
| Error | Cannot replace export file ' <i>path+file name</i> '               | The export .csv file <i>file name</i> already exists in the directory <i>path</i> , but cannot be removed so as to be replaced by a new file of the same name. Check that the file permissions allow the user to write to this file and directory |
| Error | Cannot replace output file ' <i>path+file name</i> '               | The output file <i>file name</i> already exists in the directory <i>path</i> , but cannot be removed so as to be replaced by a new file of the same name. Check that the file permissions allow the user to write to this file and directory      |
| Error | Cannot replace script file ' <i>path+file name</i> '               | The script file <i>file name</i> already exists in the directory <i>path</i> , but cannot be removed so as to be replaced by a new file of the same name. Check that the file permissions allow the user to write to this file and directory      |
| Error | Cannot respond to web server request                               | An error occurred in writing the output to the output stream for a request for data from the application via the web server. Detail on the cause is logged in the event log   |
| Error | Cannot retrieve clipboard values; cause ' <i>error cause</i> '     | An error occurred in retrieving the values from the clipboard for a paste operation   |
| Error | Cannot retrieve <i>list type</i> list                              | An error occurred retrieving the list of data <i>list type</i> from the project database. Detail on the cause is logged in the event log. The may be due to database corruption or a database server error  |
| Error | Cannot revert changes to internal table ' <i>table name</i> '      | Following an update error on internal table <i>table name</i> , another error prevented reverting any changes made to the project database. Detail on the cause is logged in the event log  |
| Error | Cannot revert changes to project database ' <i>database name</i> ' | Following an update error on project database <i>database name</i> , another error prevented reverting any changes made to the database. Detail on the cause is logged in the event log   |
| Error | Cannot revert changes to table ' <i>table name</i> '               | Following an update error on one or more tables during the table consistency check, another error prevented reverting any changes made to the project database. Detail on the cause is logged in the event log                                    |
| Error | Cannot revert changes to table(s)                                  | Following an update error on one or more tables during the table consistency check or during a macro update, another error prevented reverting any changes made to the table(s). Detail on the cause is logged in the event log                   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 138 of 166 |

| Type  | Message   | Cause   |
|-------|---|---|
| Error | Cannot revert copying table ' <i>table name</i> '   | Following a failed attempt to copy table <i>table name</i> in the project database, another error prevented reverting any changes made to the database. Detail on the cause is logged in the event log  |
| Error | Cannot revert copying table type ' <i>table type</i> '  | Following an error during copying a table type, another error prevented reverting any changes made to the project database. Detail on the cause is logged in the event log  |
| Error | Cannot revert deleting table type ' <i>table type</i> ' and table(s) ' <i>table name(s)</i> ' | Following a failed attempt to table type <i>table type</i> and its associated table(s) <i>table name(s)</i> delete one or more tables from the project database, another error prevented reverting any changes made to the database. Detail on the cause is logged in the event log |
| Error | Cannot revert deleting table(s) ' <i>table name(s)</i> '                                      | Following a failed attempt to delete one or more tables from the project database, another error prevented reverting any changes made to the database. Detail on the cause is logged in the event log   |
| Error | Cannot revert name change to table ' <i>table name</i> '                                      | Following a failed attempt to rename table <i>table name</i> in the project database, another error prevented reverting any changes made to the database. Detail on the cause is logged in the event log  |
| Error | Cannot revert type name change table ' <i>table name</i> '                                    | Following a failed attempt to change the type of table <i>table name</i> in the project database, another error prevented reverting any changes made to the database. Detail on the cause is logged in the event log  |
| Error | Cannot revert type updates to table(s) ' <i>table name(s)</i> '                               | Following a failed attempt to update table(s) <i>table name(s)</i> in the project database, another error prevented reverting any changes made to the database. Detail on the cause is logged in the event log  |
| Error | Cannot revert updates to data fields  | Following a failed attempt to update the data fields in the project database, another error prevented reverting any changes made to the database. Detail on the cause is logged in the event log  |
| Error | Cannot set comment for database ' <i>database name</i> '                                      | The attempt to update the lock status, which is stored in the project database comment, for database ' <i>database name</i> ' failed. Detail on the cause is logged in the event log  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 139 of 166 |

| Type    | Message   | Cause  |
|---------|---|--|
| Error   | Cannot store internal table ' <i>table name</i> '                                     | The attempt to store the data to internal table <i>table name</i> in the project database failed. Detail on the cause is logged in the event log   |
| Warning | Cannot store program preference values; cause ' <i>error cause</i> '                  | The program preference keys could not be stored in the preference storage node due to the cited cause  |
| Error   | Cannot update comment for table ' <i>table name</i> '                                 | The attempt to update comment for table <i>table name</i> failed. Detail on the cause is logged in the event log   |
| Error   | Cannot update data fields   | The attempt to update the data fields in the internal table ( <i>_fields</i> ) failed. Detail on the cause is logged in the event log  |
| Error   | Cannot update data types  | The attempt to update the data types in a data table or the internal table ( <i>_data_types</i> ) failed. Detail on the cause is logged in the event log   |
| Error   | Cannot update macros  | The attempt to update the macros in a data table or the internal table ( <i>_macros</i> ) failed. Detail on the cause is logged in the event log   |
| Error   | Cannot update table type ' <i>type name</i> ' <and table(s) ' <i>table name(s)</i> '> | The attempt to update table type <i>type name</i> (and tables of that type, <i>table name(s)</i> , if any) in the project database failed. Detail on the cause is logged in the event log                      |
| Warning | Cannot write to event log   | The event log file cannot be written. Check that user has file write permissions for the file and directory  |
| Error   | Cannot write to export file ' <i>path+file name</i> '                                 | An I/O error occurred while writing to the export .csv file <i>file name</i> in the directory <i>path</i>  |
| Error   | Cannot write to script file ' <i>path+file name</i> '                                 | An I/O error occurred while writing to the script file <i>file name</i> in the directory <i>path</i>   |
| Warning | Column ' <i>column name</i> ' expects a boolean value                                 | The text pasted into column <i>column name</i> is non-boolean (true/false) and the column only displays boolean (in the form of a check box); the text is ignored  |
| Warning | Column name ' <i>column name</i> ' already in use                                     | The column name <i>column name</i> is already used in the table type being edited. A different column name must be chosen  |
| Warning | Column name ' <i>column name</i> ' already in use (database)                          | The database converts the column names to one that is valid for use in PostgreSQL. The database form of the column names in the table type being edited must be unique. A different column name must be chosen |
| Warning | Column name ' <i>column name</i> ' already in use (hidden)                            | The column name <i>column name</i> is already used by a hidden column in the table type being edited. A different column name must be chosen   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 140 of 166 |

| Type    | Message   | Cause   |
|---------|---|---|
| Warning | Data must be provided for column ' <i>column name 1</i> ' or column ' <i>column name 2</i> ' [row <i>row number</i> ] | One or both columns <i>column name 1</i> and <i>column name 2</i> in the data type editor in row <i>row number</i> require a value, but both are empty. Enter a value in at least one of the columns  |
| Warning | Data must be provided for column ' <i>column name</i> ' [row <i>row number</i> ]                                      | The column <i>column name</i> in the table type editor or the data field editor in row <i>row number</i> requires a value, but is empty. Enter a value in the column  |
| Warning | Data must be provided for column ' <i>column name</i> ' [row <i>row number</i> ]                                      | The column <i>column name</i> in the table type, data type, or data field editor in row <i>row number</i> requires a value, but is empty. Enter a value in the column   |
| Warning | Data stream(s) <i>stream name(s)</i> already contain(s) a link with the name ' <i>link name</i> '; skipped            | The target data stream(s) <i>stream name(s)</i> already contains a link with the name <i>link name</i> when copying a link from one data stream to another. Change the name for the copy to one not already present in the target data stream   |
| Warning | Data type name is already in use  | The data type name entered in the data type editor's User Name column is already in use by another data type. User-defined data type names must be unique. Alter the data type name to one not in use   |
| Warning | Data type size must be a positive integer   | The value entered for a data type's size is less than 1 or is not an integer. Enter a valid size value  |
| Warning | Database connection parameter(s) missing  | One or more of the server connection parameters, server type, server host, or user name, are missing. the <b>Change user</b> and <b>Server properties</b> commands are used to set these parameters   |
| Error   | EDS conversion setup failed; cause ' <i>error cause</i> '   | An error occurred during setup for conversion of the project database to EDS XML format due to the cited cause  |
| Warning | Enumeration ' <i>enumeration</i> ' definition in table ' <i>table name</i> ' missing initial non-negative integer     | One or more of the enumeration definitions in enumeration <i>enumeration</i> in table <i>table name</i> imported from a XTCE XML file does not have a non-negative integer as the first enumeration parameter. XTCE XML enumerations must be in the format specified in paragraph 4.6.5 |
| Warning | Enumeration expects an integer data type in table ' <i>table name</i> ' for command ' <i>command name</i> '           | Command <i>command name</i> in command table <i>table name</i> has a value in an enumeration column, but the associated data type is a non-integer. Either remove the enumeration or change the data type   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 141 of 166 |

| Type               | Message  | Cause   |
|--------------------|--|---|
| Error              | Error obtaining metadata for internal table ' <i>table name</i> '              | An error occurred while attempting to read the metadata for the internal table <i>table name</i> during project database verification. The metadata provides information on the table's columns (number, names, and data types) |
| Error              | Error obtaining metadata for table ' <i>table name</i> '                       | An error occurred while attempting to read the metadata for the table <i>table name</i> during project database verification. The metadata provides information on the table's columns (number, names, and data types)          |
| Error              | Error obtaining project database ' <i>database name</i> ' metadata             | An error occurred while attempting to read the metadata for project database <i>database name</i> during project database verification. The metadata provides information on the number of tables and their names               |
| Error              | Error verifying project database ' <i>database name</i> ' consistency          | An error occurred while perform updates to project database <i>database name</i> internal tables during project database verification. Detail on the cause is logged in the event log   |
| Command Line Error | Error: <i>argument</i> must be $\geq$ <i>minimum</i> and $\leq$ <i>maximum</i> | The command line argument <i>argument</i> expects a numeric value between the values <i>minimum</i> and <i>maximum</i> , inclusive  |
| Command Line Error | Error: <i>argument</i> must be one of the following: <i>valid inputs</i>       | The command line argument <i>argument</i> is provided an argument value that is not one of the valid inputs, <i>valid inputs</i> , for this command   |
| Command Line Error | Error: <i>argument</i> not a number  | The command line argument <i>argument</i> expects a numeric value which isn't provided  |
| Command Line Error | Error: mainsize width or height not a number, or too many/few values           | The width or height contains a non-numeric (0-9) character, or other than 2 values are given  |
| Warning            | Field name ' <i>field name</i> ' already in use                                | The data field <i>field name</i> is already in use for this table. Each field within a table must be unique. Alter the field name   |
| Warning            | Field size must be a positive integer  | The value entered for a data field's size is less than 1 or is not an integer. Enter a valid size value   |
| Error              | File ' <i>path+file name</i> ' is not a backup file                            | The file chosen to restore a project database is not in the expected format. The file is either corrupted or the wrong file was chosen  |
| Warning            | Format invalid for import file ' <i>file name</i> '                            | The selected import file <i>file name</i> is not in the expected format. Correct the import file format or select another file to import  |
| Error              | Format invalid for import file ' <i>path+file name</i> '                       | The import .csv file <i>file name</i> is not in the expected format. The file is either corrupted or the wrong file was chosen  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 142 of 166 |

| Type    | Message  | Cause   |
|---------|--|---|
| Warning | Group name is already in use                               | The group name entered in the group name text field is already in use by another group. Group names must be unique. Alter the group name to one not in use  |
| Warning | Group name must be entered                                 | The group name text field is empty. Enter a valid group name into the text field  |
| Warning | ID interval must be a positive integer                     | The message ID interval value in the Assign Telemetry Messages or Assign Table Message IDs dialog is invalid. Enter a positive integer value  |
| Warning | Illegal character(s) in data type C type name              | The C type name in the data type editor table cell contains one or more illegal characters. C type names can consist of multiple words, separated by one or more spaces, which must begin with a letter or underscore and contain only letters, numerals, and underscores (an ending asterisk is legal if the corresponding base type is 'pointer' or blank). Remove the illegal character(s) |
| Warning | Illegal character(s) in data type name                     | The user data type name in the data type editor table cell contains one or more illegal characters. Data type names must begin with a letter or underscore and contain only letters, numerals, and underscores. Remove the illegal character(s)   |
| Warning | Illegal character(s) in macro name                         | The macro name in the macro editor table cell contains one or more illegal characters. Macro names must begin with a letter or underscore and contain only letters, numerals, and underscores. Remove the illegal character(s)  |
| Warning | Illegal character(s) in project name                       | The project name text field contains one or more illegal characters. The project name is also the database name, and database names must begin with a letter or underscore and contain only letters, numerals, and underscores. Remove the illegal character(s)   |
| Warning | Illegal character(s) in table name<br><i>'table name'</i>  | The table name text field contains one or more illegal characters. Table names must begin with a letter or underscore and contain only letters, numerals, and underscores. Remove the illegal character(s)  |
| Error   | Import file <i>'path+file name'</i><br>information missing | The import file <i>path+file name</i> has no tag (e.g., <i>_description_</i> or <i>_column_names_</i> ) prior to the table information  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 143 of 166 |

| Type    | Message   | Cause   |
|---------|---|---|
| Warning | Imported data type ' <i>data type name</i> ' doesn't match the existing definition  | The data type <i>data type name</i> imported from a CSV, XTCE XML, or EDS XML file already exists, but the data type definition (e.g., a size or base type) does not match the existing one. Either the existing data type or the imported one must be renamed, or the difference(s) eliminated   |
| Warning | Imported macro ' <i>macro name</i> ' doesn't match the existing definition  | The macro <i>macro name</i> imported from a CSV, XTCE XML, or EDS XML file already exists, but the macro value does not match the existing one. Either the existing macro or the imported one must be renamed, or the difference eliminated   |
| Warning | Imported table type ' <i>table type name</i> ' doesn't match the existing definition  | The table type <i>table type name</i> imported from a CSV, XTCE XML, or EDS XML file already exists, but the table type definition (e.g., a column name or input type) does not match the existing one. Either the existing table type or the imported one must be renamed, or the difference(s) eliminated   |
| Warning | Incorrect number of columns indicated for table ' <i>table name</i> ' in the column order table for user ' <i>user name</i> '                         | Detected during project database verification, the number of columns for table <i>table name</i> in the internal table __orders doesn't match the actual number of columns for that table's type. If updated the column order is reset to the default   |
| Warning | Internal table ' <i>table name</i> ' column ' <i>column name</i> ' type mismatch (expected: ' <i>expected type</i> ', actual: ' <i>actual type</i> ') | Detected during project database verification, the data type for the column <i>column name</i> in the internal table <i>table name</i> is found to not be of the type expected for this column (e.g., an integer type is specified while the table shows a text type). If updated the data type is changed to the one expected                          |
| Warning | Internal table ' <i>table name</i> ' column <i>column index</i> name mismatch (expected: ' <i>expected name</i> ', actual: ' <i>actual name</i> ')    | Detected during project database verification, the column indicated by its index is found to have a name other than the name expected for this column. If updated the name is changed to the one expected; however, the data in the column may be incorrect as well. For this case deleting the internal table (with loss of its data) may be necessary |
| Warning | Internal table ' <i>table name</i> ' has too many columns   | Detected during project database verification, the internal table <i>table name</i> is found to have more columns than the number expected. If updated any extra columns are removed  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 144 of 166 |

| Type    | Message  | Cause  |
|---------|--|--|
| Warning | Internal table ' <i>table name</i> ' is missing one or more columns  | Detected during project database verification, the internal table <i>table name</i> is found to be missing one or more columns. If updated the table is deleted (with loss of its data)  |
| Warning | Interval must be a positive integer  | The value in the table or telemetry message ID interval field is blank, or is a non-positive integer value. Enter a positive integer value for the interval  |
| Error   | Invalid application parameter(s): using the default values instead   | The application scheduler parameters stored in the project database internal table ( <i>__app_scheduler</i> ) comment are invalid. Default values replace these parameters. Detail on the cause is logged in the event log   |
| Error   | Invalid application scheduler applications detected; <i>number</i> removed                                 | The application scheduler internal table ( <i>__app_scheduler</i> ) references applications that do not exist in the project database. The <i>number</i> invalid application references are removed  |
| Warning | Invalid characters in field ' <i>field name</i> '; <i>data type</i> expected                               | The value in the data field <i>field name</i> text field contains characters that are inconsistent with the data field's data type, <i>data type</i> . Remove the illegal characters   |
| Warning | Invalid characters in message ID   | The message contains an invalid character in the telemetry scheduler Scheduler ID column cell. Enter an ID in hexadecimal format (the leading '0x' is optional)  |
| Warning | Invalid characters in message name   | The message contains an invalid character in the telemetry scheduler Scheduler Message column cell. Enter a name beginning with an underscore or alphabetical character, and containing only alphanumeric and underscore characters  |
| Warning | Invalid import file extension  | The file name extension for the selected import file is not one of the ones recognized by the application; only the extensions .csv, .xtce, and .eds are valid. Select a file with a valid extension, or, if the selected file is in one of the recognized formats, add the extension to the file name |
| Warning | Invalid input type for column ' <i>column name</i> '; <i>input type</i> expected                           | The value entered in a cell in the column <i>column name</i> does not match the expected input type, <i>input type</i> , as specified in the table type definition   |
| Warning | Invalid input value for column ' <i>column name</i> '; command argument names must be unique for a command | The command argument name entered in a cell in the column <i>column name</i> has a duplicate elsewhere in that column, and the cell values must be unique, as specified in the table type definition   |



|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 145 of 166 |

| Type    | Message  | Cause  |
|---------|--|--|
| Warning | Invalid input value for column ' <i>column name</i> '; input value ' <i>input value</i> ' must be unique | The value, <i>input value</i> , entered in a cell in the column <i>column name</i> has a duplicate elsewhere in that column, and the cell values must be unique, as specified in the table type definition   |
| Warning | Invalid output file name   | The file name entered for exporting one or more tables in CSV, XTCE XML, or EDS XML formats is not valid. Enter a valid file name  |
| Error   | Invalid rate parameter(s): using the default values instead  | The rate parameters stored in the project database internal table ( <i>_tlm_scheduler</i> ) comment are invalid. Default values replace these parameters. Detail on the cause is logged in the event log   |
| Error   | Invalid telemetry scheduler variables detected; <i>number</i> removed                                    | The telemetry scheduler internal table ( <i>_tlm_scheduler</i> ) references variables that do not exist in the project database. The <i>number</i> invalid variable references are removed   |
| Error   | Invalid web server request   | The request for data from the CCDD application via the web server is unrecognized; an unknown data stream name, incorrect number of parameters, or incorrect parameter type was passed to the telemetry scheduler request; or an error occurred while attempting to parse the data from the database for the request. Detail on the cause is logged in the event log |
| Warning | Link name is already in use  | The link name entered in the link name text field is already in use by another link. Link names must be unique. Alter the link name to one not in use  |
| Warning | Link name must be entered  | The link name text field is empty. Enter a valid link name into the text field   |
| Warning | Macro name is already in use   | The macro name entered in the macro editor's name column is already in use by another macro. Macro names must be unique. Alter the macro name to one not in use  |
| Warning | Macro value is not consistent with macro usage in table(s) ' <i>table name(s)</i> '                      | The macro value entered in the macro editor's value column does not match the input type of a column in one or more tables, <i>table name(s)</i> , where the macro is used. Alter the macro value to be consistent with the input type in every column for which the macro is referenced   |
| Warning | Message ID is already in use   | The message ID is a duplicate of another in the telemetry scheduler Scheduler ID column. Enter a unique message name   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 146 of 166 |

| Type    | Message  | Cause  |
|---------|--|--|
| Warning | Message ID name must be entered  | The table message ID name is missing from the Message ID Name text field. Enter a valid message ID name  |
| Warning | Message interval must be a positive integer  | The message number interval value in the Assign Telemetry Messages dialog is invalid. Enter a positive integer value   |
| Warning | Message name is already in use   | The message name is a duplicate of another in the telemetry scheduler Scheduler Message column. Enter a unique message name  |
| Warning | Message name must be entered   | The message name is missing from the telemetry scheduler Scheduler Message column cell. Enter a valid message name   |
| Warning | Message name or ID assignment must be selected   | The "Assign message name" and "Assign message ID" check boxes in the Assign Telemetry Messages dialog are both unchecked and the <b>Okay</b> button is selected. Select at least one of these check boxes or press the <b>Cancel</b> button  |
| Warning | Message name pattern must be in the format: <i>startText</i> <0#>d< <i>endText</i> > where <i>startText</i> and <i>endText</i> consist of alphanumeric characters and/or underscores, <i>startText</i> begins with a letter or underscore, and # is one or more digits. Note: 0# and <i>endText</i> are optional | The message name pattern in the Assign Telemetry Messages dialog is not in the expected format. The pattern must contain only alphanumeric characters, contain a single '#' character, and begin with either an underscore or alphabetical character. Change the pattern to match the valid format |
| Warning | Message starting number must be an integer >= 0  | The message starting number in the Assign Telemetry Messages dialog is invalid. Enter a positive integer value or zero   |
| Warning | Must enter or select a script  | No script is selected when the Add button is pressed in the script association manager dialog. Enter or select a script file   |
| Warning | Must select a project to delete  | No project is selected from the Delete Project dialog when the Delete button is pressed. Select one or more projects from the dialog or press the Cancel button  |
| Warning | Must select a project to open  | No project (other than the currently open one) is selected from the Open Project dialog when the Open button is pressed. Select a project from the dialog or press the Cancel button   |
| Warning | Must select a project to unlock  | No project is selected from the Unlock Project dialog when the Unlock button is pressed. Select a project from the dialog or press the Cancel button   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 147 of 166 |

| Type    | Message   | Cause   |
|---------|---|---|
| Warning | Must select a script location   | No folder is selected in which to save the script(s) retrieved from the project when the Retrieve button is pressed in the Retrieve Script dialog. Enter or select a script location, or press the Cancel button      |
| Warning | Must select a script to delete  | No script is selected from the Delete Script(s) dialog when the Delete button is pressed. Select a script from the dialog or press the Cancel button  |
| Warning | Must select a script to retrieve  | No script is selected from the Retrieve Script(s) dialog when the Retrieve button is pressed. Select a script from the dialog or press the Cancel button  |
| Warning | Must select a script to store   | No script is selected from the Store Script(s) dialog when the Store button is pressed. Select a script from the dialog or press the Cancel button  |
| Warning | Must select a table from the tree   | No table is selected from the table tree and is required for the action requested by the user. Select a table from the tree   |
| Warning | Must select at least one data field   | No data field is selected from the list of fields in the data field table editor selection dialog. Select at least one data field check box   |
| Warning | No 'look & feel' exists   | No look & feel is available to load   |
| Warning | No columns in import file ' <i>path+file name</i> ' match those in the target table | The import .csv file <i>file name</i> has no columns defined that match those in the table to which the file is being imported; no data is added to the table from the file. Check the import file's column names     |
| Warning | No data field exists  | No data field is available to select in the data field table editor selection dialog  |
| Warning | No other user exists  | An attempt was made to change to another user when no other user exists in the server   |
| Warning | No project exists for which user ' <i>user name</i> ' has access                    | The user <i>user name</i> does not have permission to access any of the project databases existing in the server. The user's permissions must be upgraded or a project database created for which the user has access |
| Warning | No role exists  | No user or role exists in the server from which to choose   |
| Warning | Number of columns differ in table ' <i>table name</i> '; continue?                  | The number of columns for table <i>table name</i> in the CSV import file doesn't match the number in the table type definition  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 148 of 166 |

| Type    | Message   | Cause  |
|---------|---|--|
| Warning | Platform does not allow key press simulation                          | Copy, paste, and insert menu commands in the table and table type editors are handled by simulating the equivalent control key presses. The platform on which the application is running does not support this type of simulation. Use the actual key press sequences to perform the desired operation |
| Warning | Problem occurred when setting the look & feel to <i>look&amp;feel</i> | An exception occurred while attempting to set the look & feel to the one selected. This can occur if the look & feel is not supported by the platform, or if there is a problem with access to the look & feel information   |
| Warning | Project ' <i>project name</i> ' has no table type defined             | The project database <i>project name</i> has no __types internal table or the __types table is empty. Create table types using the table type editor and store these in the project's database   |
| Warning | Project ' <i>project name</i> ' has no scripts                        | The user attempted to retrieve a script from the project database <i>project name</i> , but the project does not have any scripts stored in it   |
| Warning | Project ' <i>project name</i> ' has no tables                         | The project database <i>project name</i> contains no data tables. Create tables using the <b>Table   New</b> command   |
| Error   | Project database ' <i>database name</i> ' backup failed               | An error occurred preventing backing up project database <i>database name</i> . Detail on the cause is logged in the event log   |
| Error   | Project database ' <i>database name</i> ' restore failed              | An error occurred preventing restoring project database <i>database name</i> . Detail on the cause is logged in the event log  |
| Error   | Project database, user name, and/or host missing                      | The project database, user name, and or host server are not provided when executing a script from the command line   |
| Warning | Project must be selected  | No project is selected when renaming or copying a project database. Choose a project from the radio button list  |
| Warning | Project name already in use   | The project (database) name already exists on the server. Choose another name that does not match an existing project's database   |
| Warning | Project name must be entered  | The project name text field is empty. Enter a valid project name into the text field   |
| Warning | Project name too long ( <i>maximum length</i> characters maximum)     | The project name entered into the project name text field exceeds the maximum allowed. The maximum length for a database name in PostgreSQL is 63 characters. Shorten the name to within the length limit  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 149 of 166 |

| Type    | Message   | Cause   |
|---------|---|---|
| Warning | Project owner must be selected  | No owner is selected when creating a project database. Choose an owner from the radio button list   |
| Warning | Rate parameter values must be positive integer values   | The value in one or more rate parameter dialog input text fields contains a zero, negative, or non-integer value. Enter an integer value greater than or equal to 1 in each of the fields                             |
| Error   | Script query failed   | A project database query executed from within a script using the getDatabaseQuery script data access method failed. Detail on the cause is logged in the event log  |
| Warning | Search text cannot be blank   | A project database or script search was attempted without a text string for which to search entered in the search dialog. Enter a text string prior to attempting a search  |
| Warning | Server port must be a positive integer  | The value entered into the server port field in the web server properties dialog is invalid. Enter a port number (positive integer value)   |
| Warning | Server port must be blank or a positive integer   | The value entered into the server port field in the PostgreSQL server properties dialog is invalid. Either clear the field or enter a port number (positive integer value)  |
| Warning | Starting ID must be in the format <0x>#, where # is one or more hexadecimal digits                  | The starting message ID value in the Assign Telemetry Messages or Assign Table Message IDs dialog is invalid. Enter a hexadecimal value. The "0x" prefix is optional  |
| Warning | System data field name missing  | The Export EDS dialog system data field name field is empty. Enter a valid value for the system field   |
| Warning | System data field name, version, validation status, and/or classification missing                   | The Export XTCE dialog system data field name, version, validation status, and/or classification field is empty. Enter a valid value for each missing field   |
| Warning | Table ' <i>table name</i> ' printing failed   | Output of the table <i>table name</i> to a printer or file was unsuccessful. This can be due to the printer being offline   |
| Warning | Table ' <i>table name</i> ' column ' <i>column name</i> ' data type is invalid ( <i>data type</i> ) | Detected during project database verification, column <i>column name</i> in table <i>table name</i> is found to have an invalid data type. Updating replaces the data type with that from the table's type definition |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 150 of 166 |

| Type    | Message  | Cause  |
|---------|--|--|
| Warning | Table ' <i>table name</i> ' column ' <i>column name</i> ' rows <i>row number 1</i> and <i>row number 2</i> have duplicate values | Detected during project database verification, the values in table <i>table name</i> on rows <i>row number 1</i> and <i>row number 2</i> in column <i>column name</i> are found to have the same value when the indicated column for this table's type is specified to contain only unique values. If updated the value in row <i>row number 2</i> is replaced with a blank        |
| Warning | Table ' <i>table name</i> ' contains a recursive reference to ' <i>recursion table name</i> '                                    | The table <i>table name</i> has the condition wherein the table <i>recursion table name</i> contains a reference to itself as a variable or as a variable in one of its child tables (or in one of their child tables, etc.). Remove the recursive table reference   |
| Warning | Table ' <i>table name</i> ' has an unknown column (' <i>column name</i> ')   | Detected during project database verification, table <i>table name</i> is found to have a column <i>column name</i> that is not defined for this table's type. If updated the column is deleted  |
| Warning | Table ' <i>table name</i> ' is an unknown type (' <i>table type</i> ')   | Detected during project database verification, the table type <i>table type</i> specified for table <i>table name</i> is not one of the defined table types. If updated the table is deleted   |
| Warning | Table ' <i>table name</i> ' is missing column ' <i>column name</i> '   | Detected during project database verification, table <i>table name</i> is found to be missing a column <i>column name</i> that is defined for this table's type. If updated the column, with blank values for any rows, is added   |
| Warning | Table ' <i>table name</i> ' row <i>row number</i> column ' <i>column name</i> ' type mismatch                                    | Detected during project database verification, the value in row <i>row number</i> , column <i>column name</i> in table <i>table name</i> is found to have a value that is inconsistent with the data type specified in this table's table type for this column (e.g., text in an integer-only cell). If updated the value in the row and column indicated is replaced with a blank |
| Warning | Table ' <i>table name</i> ' row <i>row number</i> index mismatch   | Detected during project database verification, row <i>row number</i> in table <i>table name</i> is found to have the wrong row index. Row indices, stored in a hidden column, start at 1 for the first row and increment sequentially for each additional row. If updated the row indices are set to the expected values   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 151 of 166 |

| Type    | Message   | Cause  |
|---------|---|--|
| Warning | Table ' <i>table name</i> ' variable ' <i>variable name</i> ' array member <i>array index</i> array size doesn't match the array definition | Detected during project database verification, in table <i>table name</i> the array member <i>variable name</i> [ <i>array index</i> ] is found to have a value in the array size column that differs from that in the array's array definition. If updated the array size for the specified array member is changed to match the array definition |
| Warning | Table ' <i>table name</i> ' variable ' <i>variable name</i> ' array member <i>array index</i> data type doesn't match the array definition  | Detected during project database verification, in table <i>table name</i> the array member variable name is found to have a value in the data type column that differs from that in the array's array definition. If updated the data type for the specified array member is changed to match the array definition                                 |
| Warning | Table ' <i>table name</i> ' variable ' <i>variable name</i> ' has an extra array member   | Detected during project database verification, in table <i>table name</i> the array variable <i>variable name</i> is found to have more members than its array size allows. If updated any extra array member rows are deleted   |
| Warning | Table ' <i>table name</i> ' variable ' <i>variable name</i> ' is missing array member <i>array index</i>                                    | Detected during project database verification, in table <i>table name</i> the array variable <i>variable name</i> is found to have fewer members than its array size allows. If updated any missing array member rows are added  |
| Warning | Table ' <i>table name</i> ' variable ' <i>variable name</i> ' is missing the array definition   | Detected during project database verification, in table <i>table name</i> the array member variable name is found to have no accompanying array definition. If updated the missing array definition row is added   |
| Error   | Table export completed with errors  | An error occurred when attempting to export one or more tables to a file. A separate error dialog appears describing the specific error; this error appears at the end of the export operation   |
| Error   | Table import completed with errors  | An error occurred when attempting to import one or more tables from one or more files. A separate error dialog appears describing the specific error; this error appears at the end of the import operation  |
| Warning | Table name ' <i>table name</i> ' is a duplicate   | The table name, <i>table name</i> , appears more than once in the list of new table names entered in the table name text field. Table names must be unique. Alter the table name to one not in use   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 152 of 166 |

| Type    | Message  | Cause   |
|---------|--|---|
| Warning | Table name ' <i>table name</i> ' is already in use   | The table name, <i>table name</i> , entered in the table name text field is already in use by another table. Table names must be unique. Alter the table name to one not in use   |
| Warning | Table name ' <i>table name</i> ' matches a primitive data type                               | The table name, <i>table name</i> , entered in the table name text field matches a primitive data type's name (e.g., uint32, float). Alter the table name to meet the table naming constraints  |
| Warning | Table name ' <i>table name</i> ' matches a reserved word                                     | The table name, <i>table name</i> , entered into the table name text field matches that of a reserved word in PostgreSQL. ). Alter the table name to meet the table naming constraints  |
| Warning | Table name ' <i>table name</i> ' too long ( <i>maximum length</i> characters maximum)        | The table name, <i>table name</i> , entered into the table name text field exceeds the maximum allowed. The maximum length for a table name in PostgreSQL is 63 characters. Shorten the name to within the length limit   |
| Warning | Table name must be entered   | The table name text field is empty. Enter a valid table name into the text field  |
| Warning | Table type must be selected  | No table type is selected from the list. Select a table type  |
| Warning | Table(s) not exported ' <i>table name(s)</i> '; output file already exists or file I/O error | The table(s) <i>table name(s)</i> selected for exportation were skipped due to the output file already existing and the option to overwrite existing files was not selected, or that a file I/O error occurred (for example, insufficient file permission in the target folder) |
| Warning | Table(s) not imported ' <i>table name(s)</i> '; table already exists                         | The table(s) <i>table name(s)</i> selected for importation were skipped due to the table already existing and the option to overwrite existing tables was not selected  |
| Warning | Too many/few data field inputs   | The number of inputs for a data field following the <i>_data_field_</i> tag in the import file is incorrect (should be seven per data field definition)   |
| Warning | Too many/few macro inputs  | The number of inputs for a macro following the <i>_macros_</i> tag in the import file is incorrect (should be two per macro definition)   |
| Warning | Too many/few table name and type inputs  | The number of inputs following the <i>_name_type_</i> tag in the import file is incorrect (should be two)   |
| Warning | Too many/few table type column inputs  | The number of column description inputs following the table type name and description row under the <i>_table_type_</i> tag in the import file is incorrect (should be six)   |



|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 153 of 166 |

| Type    | Message  | Cause   |
|---------|--|---|
| Warning | Too many/few table type name and description inputs  | The number of inputs immediately following the <code>_table_type_</code> tag in the import file is incorrect (should be two)  |
| Warning | Type name is already in use  | The table type entered in the table type name text field is already in use by another table type. Table type names must be unique. Alter the table type name to one not in use  |
| Warning | Type name must be entered  | The table type name text field is empty. Enter a valid table type name into the text field  |
| Warning | Unknown column name ' <i>column name</i> ' in table ' <i>table name</i> '; continue?                               | The column <i>column name</i> for table <i>table name</i> in the CSV import file doesn't not exist in the table type definition   |
| Warning | Unknown input type ' <i>text</i> '   | The text <i>text</i> pasted into the Table Type Editor's Input Type column does not match a known input type; the text is ignored   |
| Warning | Unknown internal table ' <i>table name</i> '   | Detected during project database verification, the table <i>table name</i> is found to have a name that indicates it is an internal table, but it is not one of the recognized internal tables. If updated the table is deleted   |
| Warning | User name must be entered  | The user name field in the database login dialog is empty. Enter a valid user name into the text field. The user name field is only present if a connection to the database server cannot be established; otherwise a list of radio buttons representing the user list is displayed |
| Warning | User's guide ' <i>file name</i> ' cannot be opened; Desktop class unsupported                                      | The CCDD user's guide file cannot be opened. This is due to the Java Desktop class not being available in the operating system  |
| Warning | User's guide ' <i>file name</i> ' cannot be opened; file I/O error or no application registered to open .pdf files | The CCDD user's guide file cannot be opened. This is due to either a file I/O error or having no application registered in the operating system to open .pdf files (the help file is in PDF format)   |
| Warning | User's guide ' <i>file name</i> ' cannot be opened; file missing   | The CCDD user's guide file cannot be opened. This is due to the file not being included in the CCDD.jar file  |
| Error   | Web server failed to start   | The attempt to start the embedded Jetty web server failed. Detail on the cause is logged in the event log   |
| Error   | Web server failed to stop  | The attempt to stop the embedded Jetty web server failed. Detail on the cause is logged in the event log  |
| Error   | XTCE conversion setup failed; cause ' <i>error cause</i> '   | Conversion of the project to XTCE XML format failed for the cause specified   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | <i>Baseline</i> |
|   | Date: <i>January 2017</i>  | Page 154 of 166 |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 155 of 166 |

## Appendix D. Known Issues

1. Concurrent operation is not supported. Simultaneously interacting with the same project from more than one instance of the application can result in unexpected results or corruption of the project database.
2. If the user lacks administrator privileges then when the program starts in Windows a message similar to the following may be displayed at the command prompt:

```
Sep 10, 2014 3:06:17 PM java.util.prefs.WindowsPreferences <init>
WARNING: Could not open/create prefs root node Software\JavaSoft\Prefs
at root 0x80000002. Windows RegCreateKeyEx(...) returned error code
5.
```

This is a result of Windows attempting to create a global registry entry for the program preferences, even though only a user entry is requested. The user entry is successfully created/updated, so the warning message may be ignored. The message can be eliminated by executing the application once as an administrator since this adds the missing key. Adding the `Prefs` key manually is also an option.

3. When using the GTK+ look and feel in Linux, or any look and feel in Windows, the Files selection box does not highlight the files initially selected when the file choosing dialog is opened. The file name list does reflect the currently selected files, however.
4. When using certain Microsoft wireless mice running under Microsoft Windows the mouse wheel rotation is misinterpreted in Java applications. The issue has to do with the higher resolution capabilities of these mice. To allow a mouse with this problem to work properly with Java perform the following steps (note that if the scrolling problem returns following a reboot, then uninstall the mouse and mouse drivers and redetect the mouse - in Device Manager the mouse description should show as "HID-compliant mouse"; the steps below can then be performed):
  - a. **Control Panel** → **Mouse**
  - b. **Mouse Properties** → **Hardware** tab
  - c. Select the problematic mouse from the list ("HID-compliant mouse")
  - d. Click the **Properties** button
  - e. Go to the **Details** tab
  - f. Select "Device Instance Path" from the combo box
  - g. A value will be displayed (e.g.:  
HID\VID\_045E&PID\_0745&MI\_01&COL01&8&5538EC&0&0000); note this value. This is the path of the registry key that corresponds to this instance of the mouse
  - h. Open the registry editor and navigate to:
  - a. HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Enum\<value noted in step 7>\Device Parameters
  - i. In Device Parameters, add the following DWORD (32 bit) registry keys:
    - a. HScrollHighResolutionDisable = 1
    - b. VScrollHighResolutionDisable = 1
    - c. Delta = 120 (decimal)
  - j. Unplug, then plug back in the mouse transceiver to re-initialize the driver
  - k. The wheel scrolling should work in Java after this. If the scroll speed is too fast then perform the remaining steps

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 156 of 166 |

- l. **Control Panel → Mouse**
- m. **Mouse Properties → Wheel**
- n. Under **Vertical Scrolling** set "Roll the wheel one notch to scroll: The following number of lines at a time:" to 1
- o. Select the **OK** button
- p. Open the **Mouse and Keyboard Center**
- q. Under **Basic Settings** select **Wheel**
- r. Adjust the **Wheel Vertical Scrolling** slider to the slowest setting

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 157 of 166 |

## Appendix E. Program Notes

### a. CCDD class files

Following is a list and description of the Java class files specific to the CCDD application.

|  |  |
|--|--|
| <b>CcddAppearanceDialog</b>            | Class that creates and manages the Appearance dialog used for selecting the application look & feel. The dialog is built on the CCDDDialogHandler class                                  |
| <b>CcddApplicationParameterDialog</b>  |  |
| <b>CcddApplicationParameterHandler</b> |  |
| <b>CcddApplicationSchedulerDialog</b>  |  |
| <b>CcddApplicationSchedulerInput</b>   |  |
| <b>CcddApplicationSchedulerTable</b>   |  |
| <b>CcddAssignmentTreeHandler</b>       |  |
| <b>CcddAssignTableMsgIDDialog</b>      |  |
| <b>CcddAssignTelemetryMsgIDDialog</b>  |  |
| <b>CcddBackgroundCommand</b>           |  |
| <b>CcddButtonPanelHandler</b>          | Generic utility class for creating and handling button panels in the dialogs and frames created within the application   |
| <b>CcddClasses</b>                     | Collection of common classes used by other CCDD classes  |
| <b>CcddCommandLineHandler</b>          | Class for reading and executing the command line options   |
| <b>CcddCommonTreeHandler</b>           |  |
| <b>CcddConstants</b>                   | Class containing constant values used by the other classes   |
| <b>CcddCopyTableHandler</b>            |  |
| <b>CcddCSVHandler</b>                  | Class for handling conversion of a database project into CSV format  |
| <b>CcddDataTypeEditorDialog</b>        |  |
| <b>CcddDataTypeHandler</b>             |  |
| <b>CcddDbCommandHandler</b>            | Class for handling database commands   |
| <b>CcddDbControlHandler</b>            | Class containing the methods for connecting to, creating, copying, renaming, and deleting databases  |
| <b>CcddDbManagerDialog</b>             | Interface for the user to set the connection parameters to the database, and for creating, copying, renaming, and deleting databases. The dialog is built on the CCDDDialogHandler class |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 158 of 166 |

|                                   |  |
|-----------------------------------|--|
| <b>CcddbTableCommandHandler</b>   | Class containing the methods for creating, altering, copying, renaming, and deleting the database tables   |
| <b>CcddbVerificationHandler</b>   | Class that executes the database information consistency check   |
| <b>CcddDialogHandler</b>          | Generic utility class for creating and handling all of the dialogs created within the application  |
| <b>CcddEditorPanelHandler</b>     | Class for creating the table editor panel in which a table, description, and data fields are displayed   |
| <b>CcddEDSHandler</b>             | Class for handling conversion of a database project into EDS XML format  |
| <b>CcddEventLogDialog</b>         | Class for displaying and updating the session and stored event logs. The dialog is built on the CCDDFrameHandler class                                       |
| <b>CcddFieldEditorDialog</b>      | Class for handling data field operations. The dialog is built on the CCDDDialogHandler class   |
| <b>CcddFieldHandler</b>           | Class for handling the data field editor   |
| <b>CcddFieldTableEditorDialog</b> | Interface for inspecting and assigning values to data input fields. The dialog is built on the CCDDDialogHandler class                                       |
| <b>CcddFileIOHandler</b>          | Class containing file input and output methods   |
| <b>CcddFrameHandler</b>           | Generic utility class for creating and handling all of the frame windows created within the application  |
| <b>CcddGroupHandler</b>           | Class for handling table grouping operations   |
| <b>CcddGroupManagerDialog</b>     | Interface for the user to create, alter, or delete table groups. The dialog is built on the CCDDDialogHandler class  |
| <b>CcddGroupTreeHandler</b>       | Class containing the methods for creating and manipulating a table group tree  |
| <b>CcddImportExportInterface</b>  |  |
| <b>CcddInformationTreeHandler</b> | Generic utility class for manipulating information trees   |
| <b>CcddJTableHandler</b>          | Generic utility class for creating and handling all of the tables created within the application, including the data, type, and field tables                 |
| <b>CcddKeyboardHandler</b>        | Class for controlling keyboard input   |
| <b>CcddLinkHandler</b>            | Class containing methods to manipulate variable linkages   |
| <b>CcddLinkManagerDialog</b>      | Interface for the user to create, modify, or delete variable links, and to assign variables to the links. The dialog is built on the CCDDDialogHandler class |
| <b>CcddLinkManagerHandler</b>     |  |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. <i>JSC-#####</i>  | <i>Baseline</i> |
|   | Date: <i>January 2017</i>  | Page 159 of 166 |

|                                     |  |
|-------------------------------------|--|
| <b>CcddLinkTreeHandler</b>          | Class containing the methods for creating and manipulating a variable link tree  |
| <b>CcddMacroEditorDialog</b>        | Interface for the user to create, modify, or delete macros and macro values. The dialog is built on the CCDDDialogHandler class  |
| <b>CcddMacroHandler</b>             |  |
| <b>CcddMain</b>                     | The CCDD main application class handles flow and execution of the menu bar items   |
| <b>CcddPatchHandler</b>             | Class used to contain code to update the project database when a schema change is made. The code is written to execute only if the database has not already been updated |
| <b>CcddRateParameterDialog</b>      | Interface for assigning the telemetry sample rate parameters. The dialog is built on the CCDDDialogHandler class   |
| <b>CcddRateParameterHandler</b>     | Class that handles retrieval from and storage to the project database of the rate parameter values, and calculation of the sample rates based on the rate parameters     |
| <b>CcddSchedulerDbIOHandler</b>     |  |
| <b>CcddSchedulerDialogInterface</b> |  |
| <b>CcddSchedulerEditorHandler</b>   |  |
| <b>CcddSchedulerHandler</b>         |  |
| <b>CcddSchedulerInputInterface</b>  |  |
| <b>CcddSchedulerValidator</b>       |  |
| <b>CcddScriptDataAccessHandler</b>  | Class containing the methods whereby scripts can access the database information   |
| <b>CcddScriptExecutiveDialog</b>    | Interface for the user to select script associations to execute. The dialog is built on the CCDDDialogHandler class  |
| <b>CcddScriptHandler</b>            | Class that handles obtaining the table data and executing the associated script  |
| <b>CcddScriptManagerDialog</b>      | Interface for the user to associate scripts and data tables. The dialog is built on the CCDDDialogHandler class  |
| <b>CcddScriptStorageDialog</b>      | Interface for the user to select script files to store to or retrieve from the database. The dialog is built on the CCDDDialogHandler class                              |
| <b>CcddSearchDialog</b>             | Interface for the user to perform text string searches of the project database data tables and stored scripts  |
| <b>CcddServerPropertyDialog</b>     |  |
| <b>CcddTableEditorDialog</b>        | Class for handling data table editing. The dialog is built on the CCDDEditorPanelHandler class   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 160 of 166 |

|                                     |   |
|-------------------------------------|---|
| <b>CcddTableEditorHandler</b>       | Class that handles the commands associated with one or more data table editors. The dialog is built on the CCDDFrameHandler class |
| <b>CcddTableManagerDialog</b>       | Interface for the user create, edit, copy, rename, and delete data tables. The dialog is built on the CCDDDialogHandler class     |
| <b>CcddTableTreeHandler</b>         | Class containing the methods for creating and manipulating a data table tree  |
| <b>CcddTableTypeEditorDialog</b>    | Class for handling table type editing. The dialog is built on the CCDDEditorPanelHandler class                                    |
| <b>CcddTableTypeEditorHandler</b>   | Class that handles the commands associated with the table type editors. The dialog is built on the CCDDFrameHandler class         |
| <b>CcddTableTypeHandler</b>         | Class for handling interactions with the table types  |
| <b>CcddTableTypeManagerDialog</b>   | Interface for the user to create, edit, copy, rename, and delete table types. The dialog is built on the CCDDDialogHandler class  |
| <b>CcddTelemetrySchedulerDialog</b> |   |
| <b>CcddTelemetrySchedulerInput</b>  |   |
| <b>CcddUndoManager</b>              |   |
| <b>CcddUtilities</b>                | Class containing common utility methods used by other CCDD classes  |
| <b>CcddWebDataAccessHandler</b>     |   |
| <b>CcddWebServer</b>                |   |
| <b>CcddXTCEHandler</b>              | Class for handling conversion of a database project into XTCE XML format  |
| <b>docs</b>                         | Dummy class required for the docs folder contents to be accessible  |
| <b>images</b>                       | Dummy class required for the images folder contents to be accessible  |

## b. PostgreSQL tables

Data tables created by the user have the columns defined in the table's type definition. In addition, each data table has two initial columns that do not appear in the data table when it is edited within the application. These two columns represent the primary key (column name `_key_`) and the row index (column name `_index_`). The primary key column contains a unique, positive, sequential integer value automatically assigned by the database to each row. This value is used by the application to select specific rows in the table for modification and deletion. The row index column contains a unique, positive, sequential integer value assigned by the CCDD application. The database does not guarantee a particular order to the rows of data stored for a table; i.e., when the table's data is retrieved the row order may not be the same as the order displayed in the table editor when the data was stored. To



|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 161 of 166 |

overcome this, when a data table is loaded from the database its row index values are used to restore the row order to that specified by the user using the table editor.

In addition to the tables created by the user for containing the project's data, CCDD uses a number of internal tables for keeping track of certain information. These tables are denoted by the prefix '\_\_\_' (two underscores) and do not show up in the table trees. These tables, with their descriptions and formats, are described below:

|  |                  |   |
|--|------------------|---|
| <b>Table name:</b> <code>__app_scheduler</code>                                    |                  |   |
| <b>Description:</b> Contains the information produced by the application scheduler |                  |   |
| <b>Columns:</b>  | time_slot        | Time slot to which the application belongs in the format <i>&lt;Time Slot #&gt;</i> , where # is the time slot index  |
|  | application_info | Application information for the specified time slot. The information is composed of the application name, rate (in Hertz), maximum allotted run time (in seconds), priority, application wake-up ID (in hexadecimal), application wake-up name, application housekeeping send rate, housekeeping application wake-up name, housekeeping application wake-up ID (in hexadecimal), and scheduler group, separated by commas |

|  |               |  |
|--|---------------|--|
| <b>Table name:</b> <code>__associations</code>                           |               |  |
| <b>Description:</b> Contains the script file and data table associations |               |  |
| <b>Columns:</b>  | script_file   | Script file path and file name   |
|  | member_tables | This column contains the name(s) of the table(s) associated with the script file name. If multiple tables are associated then these are separated by plus signs with spaces on either side ( + ). The table names are in the format <i>&lt;top-level table name&gt;</i> for top-level tables, or <i>&lt;child table's prototype name&gt;.&lt;child table's variable name&gt;</i> for a table that is a child of another table. Child tables of an associated table are automatically included when loading the data for script execution |

|  |           |  |
|--|-----------|--|
| <b>Table name:</b> <code>__data_types</code>                               |           |  |
| <b>Description:</b> Contains the information for the data type definitions |           |  |
| <b>Columns:</b>  | user_name | User-defined data type name  |
|  | c_name    | C-language data type name  |
|  | size      | Data type size in bytes  |
|  | base_type | Base data type (signed integer, unsigned integer, floating point, character, or other) |
|  | index     | OID value; used to uniquely identify a row in the table                                |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 162 of 166 |

**Table name:** \_\_fields

**Description:** Contains the data field definitions and values for all of the project's data tables. Each row in the table describes a single data field. The order that the data fields appear in this table is the same as the order of the fields when displayed with a data table

**Columns:**

|                     |   |
|---------------------|---|
| owner_name          | Path (applicable for structure table instances) and name of the table to which this data field belongs. This column contains the parent and path to the table belonging to the group, separated by commas. This is in the format <i>&lt;parent table name&gt;,&lt;level 1 child table's prototype name&gt;.&lt;level 1 child table's variable name&gt;[,&lt;level 2 child table's prototype name&gt;.&lt;level 2 child table's variable name&gt;[,level 3, etc.]]</i> . Default data fields (i.e., those applied to each table of a given table type when created) are denoted by having a table name in the format <i>Type:&lt;table type name&gt;</i> |
| field_name          | Field name. This is the text displayed beside the input text field  |
| field_description   | Description of the field. The description is used as the tool tip text when the mouse pointer hovers over the data field  |
| field_size          | Width of the input text field in characters. Due to character width variations when using variable-spaced fonts the actual character width can be larger than this value  |
| field_type          | Determines the allowable values that can be input into the data field. The field types are Text, Integer, Positive integer, Non-negative integer, Float, Hexadecimal, Break, and Separator  |
| field_required      | true if the data field requires a value; false if the field may be left empty. The application does not enforce entering a value into a required field, but simply uses this designation to highlight the fields that have this flag set  |
| field_applicability | Determines, when creating tables of this type, if the data field is added: 'All tables' if the data field should be added when creating any table of this type; 'Parents only' if the field is only added to parent tables; 'Children only' if the field is only added to child tables (only applicable for structure table types)  |
| field_value         | Data entered by the user into the data field's text input field. Leading and trailing white space characters are automatically stripped off by the application before storing the value   |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-####  | Baseline        |
|   | Date: January 2017   | Page 163 of 166 |

**Table name:** `__groups`

**Description:** Contains the information for the user-defined data table groups

**Columns:**

|                            |  |
|----------------------------|--|
| <code>group_name</code>    | Group name   |
| <code>member_tables</code> | The first row for a group contains the group's description, prefixed by a number and a comma. The number is non-zero if the group represents a CFS application. The description is used as the tool tip text when the mouse pointer hovers over the group name in a table tree. For subsequent rows with the same group name this column contains the parent and path to the table belonging to the group, separated by commas. This is in the format <i>&lt;parent table name&gt;,&lt;level 1 child table's prototype name&gt;.&lt;level 1 child table's variable name&gt;[,&lt;level 2 child table's prototype name&gt;.&lt;level 2 child table's variable name&gt;[,level 3, etc.]]</i> |

**Table name:** `__links`

**Description:** Contains the information for the user-defined variable linkages

**Columns:**

|                               |   |
|-------------------------------|---|
| <code>rate_name</code>        | Name of the rate column from which the rate for the variables in this link are taken  |
| <code>link_name</code>        | Link name   |
| <code>member_variables</code> | The first row for a link contains the link's rate, in samples per second, and description, separated by a comma. The description is used as the tool tip text when the mouse pointer hovers over the link name in the link tree. For subsequent rows with the same link name this column contains the parent, table path, and variable belonging to the link, separated by commas. This is in the format <i>&lt;parent table name&gt;,&lt;level 1 child table's prototype name&gt;.&lt;level 1 child table's variable name&gt;[,&lt;level 2 child table's prototype name&gt;.&lt;level 2 child table's variable name&gt;[,level 3, etc.]],&lt;data type&gt;.&lt;variable name&gt;</i> |

**Table name:** `__macros`

**Description:** Contains the information for the macro definitions

**Columns:**

|                         |   |
|-------------------------|---|
| <code>macro_name</code> | Macro name  |
| <code>value</code>      | Macro value   |
| <code>index</code>      | OID value; used to uniquely identify a row in the table |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 164 of 166 |

**Table name:** `__orders`

**Description:** Contains the information for the table column orders, based on user

|                 |                           |  |
|-----------------|---------------------------|--|
| <b>Columns:</b> | <code>user_name</code>    | User name for which this column ordering applies   |
|                 | <code>table_path</code>   | Path to a table in the format:<br><br><code>rootTable[,structure1.variable1[,structure2.variable2[,...]]]</code><br><code>rootTable</code> is the top-level table. For a non-structure table or a top-level structure table this is the entire table path. For a structure table that is a child of another table the path contains the top-level structure table ( <code>rootTable</code> ) followed by structure name and variable name pairs leading to the target child table, separated by commas |
|                 | <code>column_order</code> | Contains the column numbers, as defined in the <code>__types</code> table, separated by colons (:), in the order in which the columns are displayed when the user, <code>user_name</code> , is viewing the table specified by <code>table_path</code>  |

**Table name:** `__script_<script name>`

**Description:** Contains the contents of the script file `<script name>`

|                 |                          |                                   |
|-----------------|--------------------------|-----------------------------------|
| <b>Columns:</b> | <code>line_number</code> | Script file line number           |
|                 | <code>Line_text</code>   | Line of text from the script file |

**Table name:** `__tlm_scheduler`

**Description:** Contains the information produced by the telemetry scheduler for the telemetry messages

|                 |                              |   |
|-----------------|------------------------------|---|
| <b>Columns:</b> | <code>rate_name</code>       | Rate name   |
|                 | <code>message_name</code>    | Message name in the format <code>&lt;Message #.#&gt;</code> where the first number is the message index and the second is the sub-index for the message   |
|                 | <code>message_id</code>      | Message ID number, in hexadecimal   |
|                 | <code>member_variable</code> | Contains the variable's rate (in hertz) followed by a backslash (\), then the parent, table path, and variable belonging to the message, separated by commas. This is in the format <code>&lt;parent table name&gt;,&lt;level 1 child table's prototype name&gt;.&lt;level 1 child table's variable name&gt;,&lt;level 2 child table's prototype name&gt;.&lt;level 2 child table's variable name&gt;[,level 3, etc.]],&lt;data type&gt;.&lt;variable name&gt;</code> |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 165 of 166 |

**Table name:** \_\_table\_types

**Description:** Contains the table type definitions for the project's data tables

|                 |                    |  |
|-----------------|--------------------|--|
| <b>Columns:</b> | type               | Table type name  |
|                 | index              | Sequential index, starting with 0, that dictates the order in which the columns appear in a table of this type. Column order can subsequently be changed by the user   |
|                 | column_name        | Column name as used in the database. This version of the column name has the capitalization removed and spaces replaced with underscores (_)   |
|                 | column_name_user   | Column name as seen by the user. This version of the name preserves the capitalization and spaces that the user specified when defining the column name, and is used as the column name in the data table  |
|                 | column_description | Description of the column. Used as the tool tip text when the mouse pointer hovers over a table's column header  |
|                 | input_type         | Name of the column's input data type (e.g., Positive integer, Enumeration). The input data type determines what values may be entered into then column   |
|                 | row_value_unique   | 't' (true) if the value in this column cannot match the value in any other rows of this column; 'f' (false) if the value is allowed to be duplicated in other rows of this column  |
|                 | column_required    | 't' (true) if the column requires a value; 'f' (false) if the column may be left empty. The application does not enforce entering a value into a required column, but simply uses this designation to highlight the columns that have this flag set  |
|                 | structure_allowed  | 't' (true) is this column allows inputs when the data type column for this row contains a structure table name; 'f' (false) if this column is to be grayed out and not allow input when the data type column for this row contains a structure name. If no data type column (a column with the input type of Primitive & Structure) is present in this table type definition then this column is ignored |
|                 | pointer_allowed    | 't' (true) is this column allows inputs when the data type column for this row contains a pointer; 'f' (false) if this column is to be grayed out and not allow input when the data type column for this row contains a pointer. If no data type column (a column with the input type of Primitive & Structure) is present in this table type definition then this column is ignored                     |

|   |  |                 |
|---|--|-----------------|
| Johnson Space Center<br>Engineering Directorate | Core Flight System Command and Data Dictionary Utility<br>User's Guide |                 |
|   | Doc. No. JSC-#####   | Baseline        |
|   | Date: January 2017   | Page 166 of 166 |

**Table name:** `__values`

**Description:** Contains the description and individual data table cell values for all of the project's data tables

**Columns:**

|                          |   |
|--------------------------|---|
| <code>table_path</code>  | <p>Path to a table in the format:</p> <p><i>rootTable[,structure1.variable1[,structure2.variable2[,...]]]</i></p> <p><i>rootTable</i> is the top-level table. For a non-structure table or a top-level structure table this is the entire table path. For a structure table that is a child of another table the path contains the top-level structure table (<i>rootTable</i>) followed by structure name and variable name pairs leading to the target child table, separated by commas</p> |
| <code>column_name</code> | <p>Column name as seen by the user (versus the database version, which is all lower case and has any spaces replaced with underscores (_))</p>  |
| <code>value</code>       | <p>If the <code>column_name</code> column is empty then this column contains the table description. If the column name is not empty this column contains the value entered by the user into the specified table and column cell. Leading and trailing white space characters are automatically stripped off by the application before storing the value</p>   |