

# Core Flight System

## Command and Data Dictionary Utility

### User's Guide

Engineering Directorate  
Software, Robotics, and Simulation Division

Version 1.2.31  
May 2018



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



Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 2 of 295

## Contents

1.0	Description .....	6
2.0	Requirements .....	7
3.0	Installation .....	7
4.0	Operation .....	8
4.1	Getting Started .....	8
4.2	Mouse and Keyboard Navigation .....	14
4.3	PostgreSQL Setup .....	16
4.4	Event Log .....	16
4.5	Data Tables .....	17
4.5.1	Table types .....	17
4.5.2	Table groups .....	21
4.5.3	Table tree .....	22
4.5.4	Data types .....	24
4.5.5	Bit fields .....	25
4.5.6	Enumerations .....	26
4.5.7	Macros .....	26
4.6	Data Fields .....	27
4.6.1	Data field editor .....	28
4.7	Input Types .....	30
4.8	Data Streams .....	33
4.9	Command Menu .....	34
4.9.1	File menu .....	34
4.9.2	Project menu .....	52
4.9.3	Data menu .....	61
4.9.4	Scheduling .....	108
4.9.5	Script menu .....	120
4.9.6	Help menu .....	128
4.10	Scripts .....	129
4.10.1	JavaScript .....	130
4.10.2	Python .....	131
4.10.3	Ruby .....	131
4.10.4	Groovy .....	132
4.10.5	Scala .....	132
4.10.6	Command line execution .....	133
4.10.7	XTCE export .....	134
4.10.8	Data access methods .....	144
Appendix A.	Acronyms .....	205
Appendix B.	Definitions .....	206
Appendix C.	Import and Export Format .....	208
Appendix C.1.	CSV .....	213
Appendix C.2.	EDS .....	217
Appendix C.3.	JSON .....	223
Appendix C.4.	XTCE .....	236
Appendix D.	Error & Warning Messages .....	244
Appendix E.	Program Notes .....	269
Appendix E.1.	Key reference .....	269
Appendix E.2.	Program preferences .....	270
Appendix E.3.	CCDD class files .....	275

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 3 of 295

Appendix E.4.	PostgreSQL tables .....	281
Appendix E.5.	PostgreSQL Functions .....	289
Appendix E.6.	Known Issues.....	294

## Figures

Figure 1.	CCDD inputs and outputs.....	6
Figure 2.	CCDD main window.....	14
Figure 3.	Example array display .....	19
Figure 4.	Example dialog .....	20
Figure 5.	Table tree .....	22
Figure 6.	Table tree expansion.....	22
Figure 7.	Data field editor .....	28
Figure 8.	Select User dialog (no server connection) .....	34
Figure 9.	Select User dialog (server connected) .....	35
Figure 10.	Database server dialog.....	35
Figure 11.	Search event log dialog .....	37
Figure 12.	Web Server dialog .....	46
Figure 13.	Preferences dialog; look and feel preferences .....	46
Figure 14.	Example look and feel differences .....	47
Figure 15.	Font preferences.....	47
Figure 16.	Font selection dialog .....	48
Figure 17.	Color preferences.....	48
Figure 18.	Color selection dialog.....	49
Figure 19.	Size preferences .....	49
Figure 20.	Spacing preferences .....	50
Figure 21.	Path preferences .....	51
Figure 22.	Other settings.....	52
Figure 23.	Select Project dialog.....	53
Figure 24.	Create Project dialog.....	54
Figure 25.	Rename Project dialog .....	55
Figure 26.	Copy Project dialog .....	56
Figure 27.	Delete Project dialog.....	57
Figure 28.	Backup Project dialog.....	57
Figure 29.	Unlock Project(s) dialog .....	59
Figure 30.	Verification and termination dialog .....	60
Figure 31.	Example Perform Corrections dialog .....	61
Figure 32.	New Table dialog.....	62
Figure 33.	Edit Table dialog.....	63
Figure 34.	Example table editor .....	63
Figure 35.	Import data and fields into an existing table dialog .....	66
Figure 36.	Data table search dialog.....	67
Figure 37.	Example of macro name display and pop-up dialog in a data table.....	69
Figure 38.	Special indicator flag example .....	70
Figure 39.	Rename Table dialog.....	72
Figure 40.	Copy Table dialog.....	73
Figure 41.	Delete Table dialog .....	74
Figure 42.	Import table(s) dialog.....	75
Figure 43.	CSV export dialog .....	78
Figure 44.	EDS export dialog .....	79

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 4 of 295

Figure 45.	JSON export dialog .....	80
Figure 46.	XTCE export dialog .....	81
Figure 47.	Manage Groups dialog .....	82
Figure 48.	New Group dialog .....	83
Figure 49.	Table type editor .....	85
Figure 50.	New table type dialog .....	87
Figure 51.	Data Type Editor dialog .....	91
Figure 52.	Example pointer to a structure data type .....	92
Figure 53.	Structure name pop-up .....	93
Figure 54.	Macro Editor dialog .....	94
Figure 55.	Example of macro name selection pop-up dialog in a macro value cell .....	95
Figure 56.	Assign Message IDs dialog .....	96
Figure 57.	Reserved Message ID Editor dialog .....	97
Figure 58.	Example Show all IDS dialog .....	98
Figure 59.	Example Duplicate Message IDs dialog .....	99
Figure 60.	Project data field management dialog .....	99
Figure 61.	Example Select Data Field(s) dialog .....	100
Figure 62.	Example Show/Edit Data Fields dialog .....	101
Figure 63.	Byte alignment selection dialog .....	103
Figure 64.	Example padding adjustment progress/cancellation dialog .....	103
Figure 65.	Structure table showing highlighted padding variables .....	104
Figure 66.	Example variable paths & names dialog .....	106
Figure 67.	Search tables dialog .....	107
Figure 68.	Manage Links dialog .....	108
Figure 69.	New Link dialog .....	109
Figure 70.	Copy Link(s) dialog .....	110
Figure 71.	Example link copy failure dialog .....	111
Figure 72.	Telemetry Scheduler dialog .....	112
Figure 73.	Telemetry message auto-fill progress/cancellation dialog .....	113
Figure 74.	Assign telemetry message names and IDs dialog .....	114
Figure 75.	Application Scheduler dialog .....	116
Figure 76.	Application time slot auto-fill progress/cancellation dialog .....	117
Figure 77.	Rate Parameters dialog .....	118
Figure 78.	Application Parameters dialog .....	119
Figure 79.	Manage Script Associations dialog .....	121
Figure 80.	Execute Script(s) dialog .....	124
Figure 81.	Halt script execution dialog .....	124
Figure 82.	Script selection dialog .....	125
Figure 83.	Retrieve Script(s) dialog .....	126
Figure 84.	Delete Script(s) dialog .....	127
Figure 85.	Script search dialog .....	128
Figure 86.	About dialog .....	129
Figure 87.	XTCE export script method flow .....	143
Figure 88.	Structure table for import/export format examples .....	208
Figure 89.	Command table for import/export examples .....	209
Figure 90.	Structure table type definition for import/export example .....	210
Figure 91.	Command table type definition for import/export example .....	211
Figure 92.	Data field, data type, and macro definitions for import/export example .....	212

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 5 of 295

## Tables

Table 1.	Command line arguments.....	12
Table 2.	Structure column names and input data types.....	18
Table 3.	Command column names and input data types .....	21
Table 4.	Variable tree icons .....	24
Table 5.	Default primitive data types .....	25
Table 6.	Web data access commands.....	45
Table 7.	XML special data fields.....	79
Table 8.	Overridable XTCE export methods.....	142
Table 9.	Script Data Access Methods.....	204

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 6 of 295
---	---	---------------------------------

## 1.0 Description

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

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 PostgreSQL database server can be run locally or centralized on a remote host for easier access by multiple users. Data can be imported into the application from files in comma-separated values (CSV), JavaScript Object Notation (JSON), electronic data sheet (EDS), and extensible markup language (XML) telemetric and command exchange (XTCE) formats. Data can be exported from the application to files in CSV, JSON, EDS, and XTCE 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 related 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.

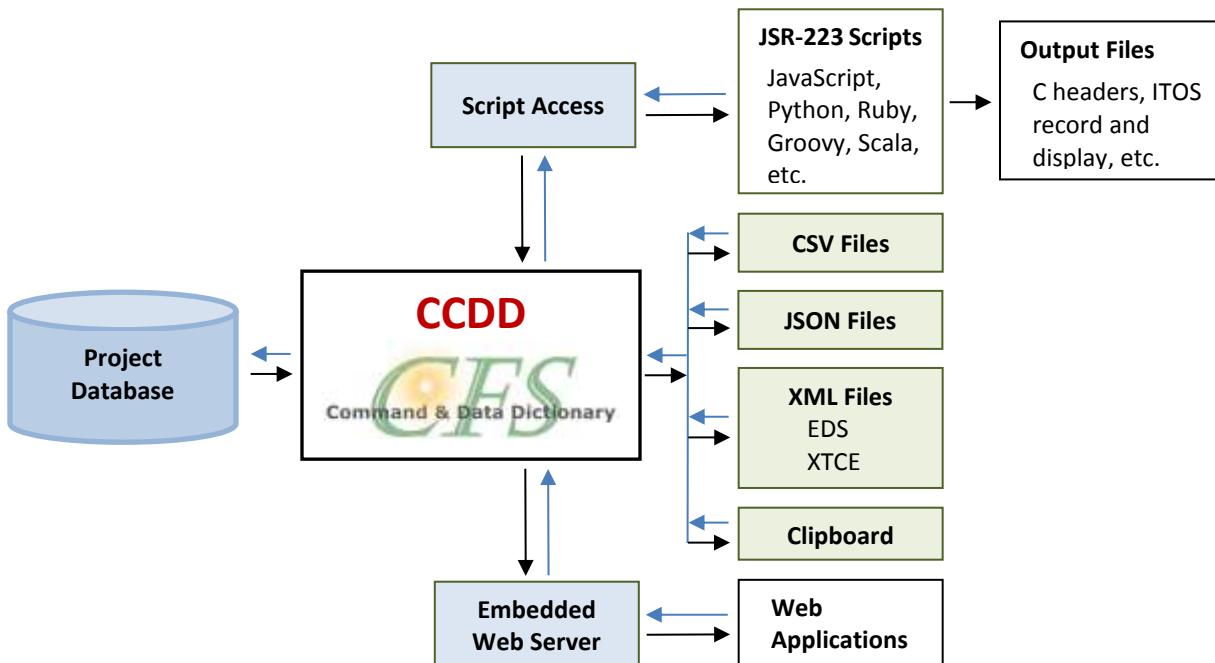


Figure 1. CCDD inputs and outputs

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

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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Version 1.2.31

Date: May 2018

Page 7 of 295

## 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 CentOS 6 Linux, Microsoft Windows 7, and Apple OS X using JavaSE 1.8
- PostgreSQL 8.4

CCDD comes with the following Java Database Connectivity (JDBC) driver, embedded Jetty web server, and JavaScript Object Notation (JSON) versions:

- JDBC driver 9.4-1207 (type 4)
- Jetty 9.2.18.v20160721
- JSON simple 1.1.1

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

- JavaScript (ECMAScript) 1.8 (Mozilla Rhino 1.7 release 3)
- JavaScript ECMA – 262 Edition 5.1 (Oracle Nashorn 1.8.0\_131)
- Python 2.7 (Jython 2.7 )
- Ruby 9.0.1.0 (JRuby 9.0.1.0)
- Groovy 2.4.4 (Groovy Scripting Engine 2.0)
- Scala 2.12.4 (Scala REPL 2.0) - *requires Java 1.8*

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`, the user's guide file `CCDD_Users_Guide.pdf`, and the library folder `CCDD_lib` and its contents to a folder. The application requires read/write access to a folder so that event log file(s) can be created; the default folder for the log files is the file from which the application is executed though this can be changed (see paragraph 4.4 for further information on event logs and paragraph 4.1 for changing the log file path).

**Java and PostgreSQL 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 `CCDD_lib` folder contains versions of the library files required by CCDD for PostgreSQL interaction, the embedded web server, and import/export format conversions. Updated versions of these libraries can be used in place of those in this folder as described in paragraph 4.1. Information on these libraries can be found at the following:

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 8 of 295
---	---	---------------------------------

- The PostgreSQL JDBC driver is located at [jdbc.postgresql.org](http://jdbc.postgresql.org). Note that version of Java installed determines which JDBC driver to use.
- Jetty must be installed in order to use the web server. Jetty is located at <http://www.eclipse.org/jetty/>. Note that version of Java installed determines which Jetty version to use.
- The JSON conversion library is located at <https://code.google.com/archive/p/json-simple>.
- Information regarding EDS is located at <http://www.ccsds.org>.
- Information regarding XTCE is located at <http://www.omg.org/space/xtce>.

CCDD supports the use of JVM-based scripting languages. At least one of these languages must be installed for the application to make use of CCDD's project-data-to-script-language interface. Only the scripting language(s) intended for use with the application need to be installed. The application was tested with five of the available languages: JavaScript, Python, Ruby, Groovy, and Scala. Details are provided in this in subsequent sections of this guide on the use of these five scripting languages; installation and use of other languages should be similar. The CCDD package provides examples of common scripts in the JavaScript, Python, Ruby, and Groovy languages.

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

- *JavaScript*® is part of the JRE download and installation from [www.java.com](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).
- *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).
- *Scala* can be downloaded from [www.scala-lang.org](http://www.scala-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 paths and .jar file names for the CCDD application (CCDD.jar) and supporting libraries, separated by colons (:) with no intervening spaces, and *args* are optional command line arguments in the form:

```
[ [<- or />] command_value [...]]
```

Versions of the JDBC driver, Jetty server, and JSON conversion libraries are included in the CCDD\_lib folder. If the default version is incompatible with the installation or an updated version is desired then the defaults can be overridden by including the library path(s) and .jar file name(s) in the *class\_paths* argument (or by overwriting the file(s) in the CCDD\_lib folder). The library files needed are shown below; <version> is the specific version number of the installed file that is part of the file's name.

```
<JDBC path>/postgresql-<version>.jar
<Jetty path>/lib/jetty-http-<version>.jar
<Jetty path>/lib/jetty-io-<version>.jar
```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
		Doc. No. JSC-37494	Version 1.2.31
Date: May 2018			Page 9 of 295

```
<Jetty path>/lib/jetty-security-<version>.jar
<Jetty path>/lib/jetty-server-<version>.jar
<Jetty path>/lib/jetty-servlet-<version>.jar
<Jetty path>/lib/jetty-util-<version>.jar
<Jetty path>/lib/servlet-api-<version>.jar
<JSON path>/json-simple-<version>.jar
```

The library class path(s) must be specified for each scripting language other than JavaScript. The library files required by Ruby, Python, Groovy, and Scala are shown below; <version> is the specific version number of the installed file that is part of the file's name. Library files for other scripting languages should follow a similar format.

```
<JRuby path>/lib/jruby.jar
```

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

```
<Jython path>jython.jar
<Groovy path>/lib/groovy-<version>.jar:<Groovy path>/lib/groovy-jsr223-
<version>.jar
<Scala path>/lib/scala-compiler.jar
```

Each command line 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 the same command is entered more than once then each instance is applied/executed in the order of appearance on the command line. The commands are not case-sensitive, so “-user” is the same as “-USER”, “-User”, etc.

<b>Command</b>	<b>Description</b>	<b>Value</b>	<b>Default Value</b>
backup	Sets the file path and name to which to automatically backup the project database once it 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>
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 is case insensitive	true
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 is case insensitive	true

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 10 of 295	

Command	Description	Value	Default Value
execute	Runs the supplied script association(s), or script(s) using the supplied table(s) and/or group(s). The application's graphical user interface (GUI) is not displayed; exits upon completion of the script(s). See paragraph 4.9.5.1 for more detail	Script association name or script file name and associated table(s). 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) and/or group name(s) are placed after the script name and a colon. Group names must be preceded by 'Group:'. If multiple tables/groups are specified the table/group names must be separated by a plus (+) character. When multiple scripts are run each definition, as described above, is separated by a semicolon (the string describing the associations must be bounded by quotes in this case; alternately, multiple <i>execute</i> commands can be issued from the command line). Bounding the value with quotes allows spaces and the semicolon to be used	<i>None</i>
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 is case insensitive	true
host	Sets the name of the PostgreSQL server's host	PostgreSQL server host name. The host name is case sensitive	<i>Previous session's PostgreSQL host (localhost for the first use)</i>
laf	Sets the application look & feel	"Look and feel" name (e.g., "Nimbus", "Windows", etc.). The names are case sensitive	<i>Previous session's L&amp;F (Metal for the first use)</i>
logPath	Sets the path to the folder where the session event log is stored	Path to the folder in which to store the session event log	<i>None (the folder in which the application is started is used)</i>

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 11 of 295	

Command	Description	Value	Default Value
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
password	Sets the user's PostgreSQL password	Password for user name for PostgreSQL. The password is case sensitive	None
port	Sets the port of the PostgreSQL server's host	PostgreSQL server port. The server port must be blank or a positive integer	Previous session's PostgreSQL port (5432 for the first use)
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 (none for the first use)
scriptOutPath	Sets the path to the folder where the script output files should be stored. The scripts can access this path and use it to set the folder for the output files	Path to the folder in which to store the script output files	None (the folder in which the application is started is used)
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 is case insensitive	true
ssl	Enables or disables a secure socket layer (SSL) connection to the PostgreSQL server. SSL must be set to "on" if the server expects an SSL connection, and "off" if it does not	"off" to disable SSL; "on" to enable SSL. The value text is case insensitive	Previous session's SSL state (off for the first use)
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 is case insensitive	true

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 12 of 295	

Command	Description	Value	Default Value
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 is case insensitive	true
user	Sets the user name to use when connecting to the PostgreSQL server	User name for PostgreSQL. The user name is case sensitive	<i>Previous session’s user name (none for the first use)</i>
version	Displays the CCDD version number and build date, then terminates the application. Any other commands are ignored		
webport	Set the port for the embedded web server. See paragraph 4.9.1.6.2 for more detail	Valid port number for the web server to listen to for queries	<i>Previous session’s web server port (7070 for the first use)</i>
webserver	Enables the embedded web server. See paragraph 4.9.1.6.1 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

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, the default libraries in the `CCDD_lib` folder are used, and the script library files are installed in the folder `/opt`. The project initially opened is “myProject” by user “userName”:

```
java -classpath ./CCDD.jar:/opt/jruby-9.0.1.0/lib/jruby.jar:/opt/jython2.7
      .0/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:/opt/scala-2.12.4/lib/scala-compiler.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/jruby-9.0.1.0/lib/jruby.jar:/o
pt/jython2.7.0/jython.jar:/opt/groovy-2.4.4/lib/groovy-2.4.4.jar:/opt/g
roovy-2.4.4/lib/groovy-jsr223-2.4.4.jar:/opt/scala-2.12.4/lib/scala-
compiler.jar CCDD.CcddMain'
```

For Microsoft Windows, the `doskey` command can be used to create an alias (the 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...]
```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b>	
	Doc. No. <i>JSC-37494</i>	<i>Version 1.2.31</i>
	Date: <i>May 2018</i>	Page 13 of 295

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:

```

usage:
java -classpath <class_paths> CCDD.CcddMain [[<- or />]<command> <value> [...]]
Command line arguments:
  Description           Command      Value
  -----
  Backup project on connecting    backup      backup file name
  Show command events          command     true or false
  Show events                  events      true or false
  Execute script(s)            execute    [script name] or
                                         [" or ']script file name[:table1 or Group:group1
                                         [+...[+tableN or Group:groupN]]][;...]" or '
  Show fail events             fail       true or false
  Set PostgreSQL server host   host       host name
  Load look & feel              laf       look & feel
  Set event log file path      logPath    file path
  Set main window size         mainSize   widthxheight
  Set user password            password   user password
  Set PostgreSQL server port   port      port number
  Select CCDD project          project    project name
  Set script output file path  scriptOutPath file path
  Show web server events       server    true or false
  Enable/disable SSL            ssl       on or off
  Show status events           status    true or false
  Show success events          success   true or false
  Set user name                user     user name
  Display CCDD version         version   -
  Set web server port          webport   port number
  Enable web server            webserver nogui or gui

```

Once the application is executed the CCDD main window appears as shown in Figure 2. If password authentication is enforced (see paragraph 4.3) and a password is not supplied on the command line then the Select User dialog appears (see paragraph 4.9.1.1), allowing the user and password to be entered. The graphical user interface (GUI) “look and feel” (L&F) can be selected by the user from a list of those 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 4.9.1.7.1 on how to alter the L&F.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 14 of 295

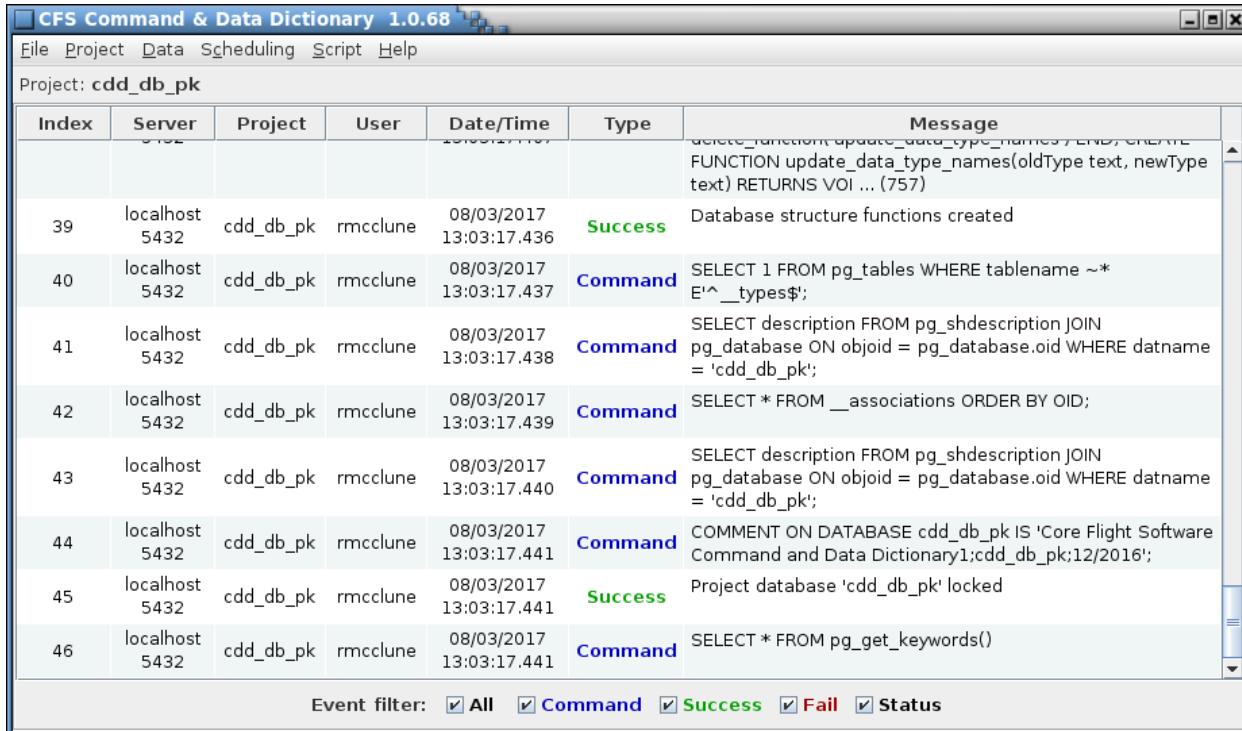


Figure 2. CCDD main window

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 4.4 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 item 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 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 **File** 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). See Appendix E.1 for a list of special keys and key sequences.

When a dialog containing a table is initially displayed it has no table 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 15 of 295

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 paste the data into another application (e.g., spreadsheet or text document) or another table use the Ctrl-V or Ctrl-I 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. If the table cell's input type (see 4.7 for information on input types) supports multiple lines then pressing Alt-Enter inserts a line break.

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

For most tables in the application, row sort order, column width, and column position are user-adjustable. The table rows can be sorted by column by positioning the mouse pointer over the column's header and pressing the left mouse button. The rows are sorted in ascending order, depending on the selected column's contents, and an icon appears beside the column name indicating the sort direction. Selecting the column again sorts in descending order (with a corresponding change in the sort direction icon), and a third selection restores the rows to their original order and removes the sort direction icon(further column header selection repeats this sequence). Only one column can be sorted at a time – selection of another column removes the sort from the first sorted column before applying the sort to the newly selected one. The column width may be resized by positioning the mouse pointer over the right border of the column header (the mouse pointer changes to indicate resizing is possible), pressing and holding a mouse button, then moving the mouse left or right; release the mouse button to exit resizing. Automatic resizing, based on the widest of the contents of the cells and header, is accomplished by double clicking a mouse button when the resize cursor appears. 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.

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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 16 of 295
---	---	----------------------------------

## 4.3 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 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 4.9.2.3; 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.

## 4.4 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 4.9.1.3. The log automatically scrolls to the latest entry when an event 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 and the port number.  |
| <b>Project</b>   | Name of the project 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:<br><br><b>Command</b> Indicates a PostgreSQL command issued to the database.                                 |

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 17 of 295

<b>Success</b>	Indicates the database transaction completed successfully.
<b>Fail</b>	Indicates the database transaction or web server command failed.
<b>Status</b>	Indicates the log entry provides application status information.
<b>Server</b>	Indicates a web server command.
<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 or web server 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). A <b>Server</b> command displays web server command information. The message length displayed is limited by default to 250 characters in order to prevent bogging 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.

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 other check boxes – unchecking it clears the other check boxes, and checking it selects the others. If none of the check boxes are selected no log entries are displayed. Note that for the single log entry viewer the filter check boxes are not displayed.

## 4.5 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.

### 4.5.1 Table types

Every data table is built from a table type definition which defines the table's columns. Think of a table type as the blueprint from which other tables are created. Two types of tables are available by default 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 4.9.3.10 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 4.10 for information regarding script access).

Every table that is created from a table type is considered a *prototype*. A 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.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 18 of 295

A prototype table that is not referenced from within another table is considered a *root* table. A table that references another table is that table's *parent*, and the referenced table is a *child* of the parent. The root table is also a parent if it contains a reference to another 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.

#### 4.5.1.1 Structure tables

Structure table rows represent C-program variables and related information. The variables can either be of a primitive data type (e.g., integer, char, 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. 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 root 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 4.7 for more information on input types). Therefore, for a table to be treated as a structure it must include at a minimum the columns with the input types "Variable name", "Primitive & Structure", "Array index", "Bit length", "Rate", and "Enumeration". 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. Table 2 shows the default structure column names and the corresponding input types.

Default Column Name	Input Type
Variable Name	Variable name <sup>1</sup>
Description	Description
Units	Units
Data Type	Primitive & Structure <sup>1</sup>
Array Size	Array index <sup>1</sup>
Bit Length	Bit length <sup>1</sup>
Rate	Rate <sup>1</sup>
Enumeration	Enumeration <sup>1</sup>

<sup>1</sup> A table must contain this input types to be identified as a structure

Table 2. Structure column names 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 data types that have a base type of "signed integer" or "unsigned integer" (see paragraph 4.5.4 for information on data types). If a non-integer data type is selected in the data type column then the bit length and enumeration cells for that row are

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 19 of 295
---	---	----------------------------------

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, each with a minimum size of 2. 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 array text is automatically formatted with spaces). 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 ways, via the **Expand arrays** command (see paragraph 4.9.3.2.3.5) 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 an expanded array definition.

array definition	Variable Name	Data Type	Array Size
	cmdCtr	uint16	3, 2
array members	cmdCtr[0][0]	uint16	3, 2
	cmdCtr[0][1]	uint16	3, 2
	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. The exception is the bit length column – an array definition or member cannot be assigned a bit length. To change the array member names or data type make the change to the array definition row; the member rows are 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 a special instance of the character base data type. If no array size is specified then the string variable is simply a single character. If an array size is supplied then the first (rightmost) array dimension 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 4.9.3.2 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 figure are replaced by the prototype and child table names respectively). Once a structure is referenced by another one it is no longer a root structure

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 20 of 295

table, and therefore can't have its own child tables, only those that are part of the hierarchy of the root structure to which the structure's prototype belongs.

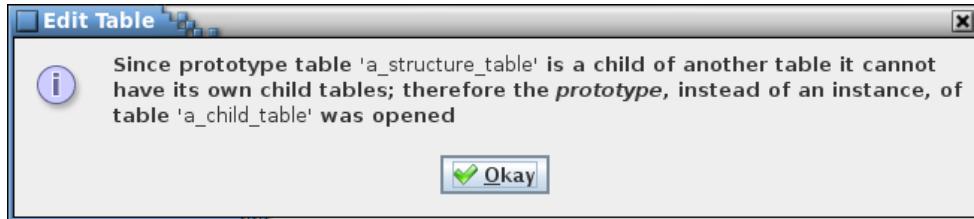


Figure 4. Example dialog

As mentioned above, a child table inherits the data values of its prototype table. If a child table's data value is edited it overrides the inherited value, producing a "custom" value for the child. Even if the child's prototype values are subsequently changed the child retains its "custom" values. Special action must be taken to remove the child's custom value and have it use the prototype value – see paragraph 4.9.3.2.2.8.2.

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.

#### 4.5.1.2 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 Name" and "Command Code". The column names can be changed if desired; it's the columns' input type that identifies the column (see paragraph 4.7 for more information on input types). Therefore, a command table must include columns with the "Command name" and "Command code" input types. Other columns ("Description", "Arg 1 Name", "Arg 1 Description", "Arg 1 Units", "Arg 1 Data Type", "Arg 1 Array Size", "Arg 1 Enumeration", "Arg 1 Minimum", and "Arg 1 Maximum") are automatically included for command tables; these additional columns can be altered, or even deleted. Only a single command name and command code column is allowed per table type definition. Columns containing other command information, such as those to describe more command arguments (name, data type, enumeration, minimum, maximum, etc.) can be added at the user's discretion. Table 3 shows the default command column names and the corresponding input types.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 21 of 295

Default Column Name	Input Type
Command Name	Command name <sup>1</sup>
Command Code	Command code <sup>1</sup>
Description	Description
Arg 1 Name	Argument name
Arg 1 Description	Description
Arg 1 Units	Units
Arg 1 Data Type	Primitive
Arg 1 Array Size	Array index
Arg 1 Enumeration	Enumeration
Arg 1 Minimum	Minimum
Arg 1 Maximum	Maximum

- 1 A table must contain this input types to be identified as a command

Table 3. Command column names and input data types

The order in which the columns are defined in the command table type definition determines which columns are associated with a particular command argument. See paragraph 4.9.3.10 for details on defining columns for a table type definition. Any column defined prior to the first column with the input type “Argument name” is assumed to apply to the command in general. All column definitions between two columns with the input type “Argument name” (or the last argument name column and the end of the table type definition) are assumed to be associated. For the default command table type definition the Description column is considered generic, and all of the columns following “Arg 1 Name” are associated with the first (and only) command argument. If a subsequent command argument is added it would begin with a column of input type “Argument name”, placed after all of the first argument’s associated columns, and followed by any column definitions associated with the new argument. 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 data types that have a base type of “signed integer” or “unsigned integer” (see paragraph 4.5.4 for information on data types). 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.

#### 4.5.2 Table groups

Data tables can be assigned to user-defined groups (see paragraph 4.9.3.9 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 4.5.3 concerning tree 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 4.9.3.9 for details on designating a group as an application, and paragraph 4.9.4.3). 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 4.6 for more details on data fields.

#### 4.5.3 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 level branches 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 second top level branch that may be displayed, *Parents & Children*, shows the root tables and, if applicable, their children as sub-branches, and the children of those tables as further sub-branches, etc. See Figure 5 and Figure 6.

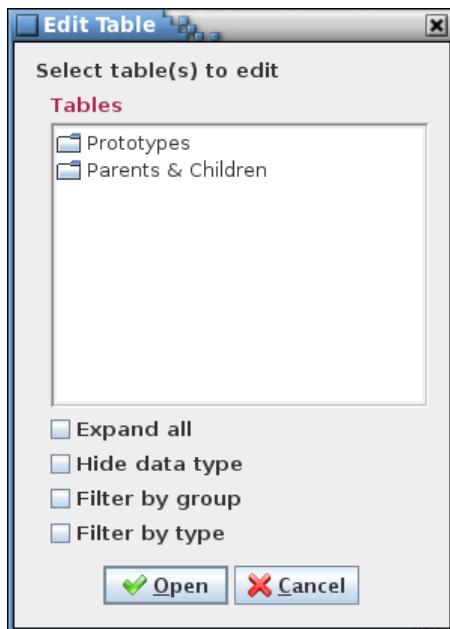


Figure 5. Table tree

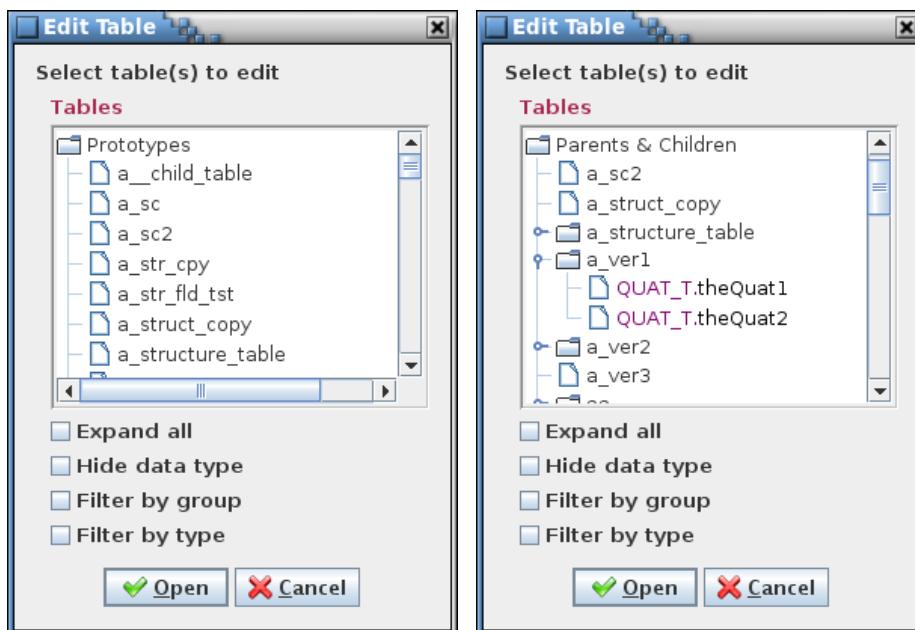


Figure 6. Table tree expansion

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 23 of 295

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; for a table this is the text from the table's description field; see Figure 34).

Below the tree are one or more check boxes. The **Expand all** check box 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 names displayed in the tree for structure and primitive variables are in the format *data type.variable name* (see paragraph 4.5.3.1). For structure variables the data type is the name of the structure defining the variable; for primitive variables (e.g., integers, floats) the data type is the name of the primitive. Color is used to highlight the data type. The **Hide data type** check box appears with table trees that display structure and primitive variables. When unchecked, the tree displays the variables in the full format as described. When checked, the data type portion of the name is not displayed.

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 *Parents & Children* branches. These sub-branches are the groups defined by the user (see paragraphs 4.5.2 and 4.9.3.9). Tables belonging to the group are displayed as sub-branches of the group branch. A special group, labeled "All tables", appears in the tree below the user-defined group sub-branches. The "All tables" group is an automatically defined group that displays all tables, including those that are not a member of a group, so that every table is still available for selection in the table tree while the group 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 *Parents & Children* branches. These sub-branches are the table types: structure, command, and any others defined by the user (see paragraphs 4.5.1 and 4.9.3.10). Tables of a given type are displayed as sub-branches of the table type branch. In other words, all of the Structure type tables appear under a *Structure* branch, all Command type tables under a *Command* branch, and so on for each defined table type. Deselecting the check box removes the type branches.

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.

#### 4.5.3.1 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 in the same manner as in the table tree, except that the variables belonging to the structure tables are also shown as branches of their parent structure. Variable trees are used where selection of variables is required; e.g., in the links manager (paragraph 4.9.4.1) and the telemetry scheduler (paragraph 4.9.4.2). 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 24 of 295	

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 4.5.5 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

#### 4.5.4 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, a reference to a structure, or a pointer.

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).

A project database is automatically provided with a number of primitive data types. These can be altered or deleted. The default primitive types 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
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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494		Version 1.2.31
	Date: May 2018		Page 25 of 295

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. The application treats a string as an array of characters. An array of data type ‘string’ is treated specially by the application. See paragraph 4.5.1.1 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 4.9.3.11) is used to create, modify, and delete the primitive data types.

#### 4.5.5 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 with bit lengths specified 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 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. In order to prevent two variables from being packed together 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.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
	<i>Version 1.2.31</i>	Page 26 of 295

## 4.5.6 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 format for an enumeration is as follows:

```
<1st enum value> <value/label separator> <1st enum label> [<value/label separator> <other 1st enum attributes>]
  [<pair separator> <2nd enum value> <value/label separator> <2nd enum label> [<value/label separator> <other 2nd enum attributes>]] [[<pair separator>...]] [<pair separator> <nth enum value>
    <value/label separator> <nth enum label> [<value/label separator> <other nth enum attributes>]]]
```

The enumeration value/label and pair separator characters are at the discretion of the user. The application detects these characters automatically.

The label can contain multiple attributes (e.g., color, limit, etc.) – use the value/label separator character to delineate each attribute. Below is an example of an enumeration with three possible values (0, 1, and 2):

```
0 | Off | red, 1 | On | green, 2 | Standby | 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. The value/label separator is the “|” character, and the pair separator is the comma (“,”). Any spaces and/or tabs bounding the separator characters are ignored.

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 parameter’s data type must be an integer 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. Data type and enumeration column associations are determined by their respective input type designation (see paragraph 4.7) and are paired based on their order in the table’s type definition. Command tables may have one or more command arguments, each with a data type and enumeration (see paragraph 4.5.1.2); these are automatically paired when both are present in a given argument. The EDS and XTCE XML conversions (see paragraphs 4.9.3.8.2 and 4.9.3.8.3) also check that the data type is valid for an enumeration, generating an error message if an enumeration is associated with a non-integer data type.

## 4.5.7 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 4.9.3.12. A macro name, when entered into a cell, must be bounded on either side by a pair of ‘#’ characters (##, with no intervening spaces) in order for the macro to be recognized. Text that’s entered into a cell bounded by the macro delimiters is only recognized as a macro if the macro is defined. If the macro isn’t recognized then the characters are treated as any other text string. If the macro is

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 27 of 295

subsequently defined the cell automatically recognizes the text string as a macro. Macros can also be entered using the **Insert macro** command in the table editor; see paragraph 4.9.3.2.2.6.

A special type of macro is the *sizeof(data type)* call. The *sizeof()* call returns the size in bytes of the specified primitive or structure data type. *sizeof()* can be used in the same locations as a macro. The macro identifier characters (##) are not used with the *sizeof()* call.

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).

Text containing a macro and/or *sizeof()* call is evaluated as a mathematical expression if the resulting expansion of the macro(s) and *sizeof()* call(s) in the text is in a valid expression format. Mathematical expressions may contain numbers (including floating point values), the operators '+' (addition or sign, based on context), '-' (subtraction or sign, based on context), '\*' (multiplication), and '/' (division), parentheses ('(' and ')') for nesting expressions, and spaces (which are ignored).

As an example assume two macros are defined, MACRO\_A with a value of "2" and MACRO\_B with a value of "3". The text "(##MACRO\_A## + 4) / ##MACRO\_B", when the macros are expanded, is "(2 + 4) / 3". Since this is a valid mathematical expression the resulting text is "2" since  $(2 + 4) / 3 = 2$ . Multiple expressions can exist in the same text if each is separated from the other by a comma (,). For example, the text "#MACRO\_A, sizeof(float) \* 3" when expanded is "2, 4 \* 3" (using the value for MACRO\_A stated previously and assuming a float is 4 bytes). Each portion of the text, when separated at the comma, is a valid mathematical expression, so the resulting text is "2, 12". This format is useful when setting the array size for a multi-dimensional array variable.

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. All of the macros in a data table can be temporarily replaced by their corresponding values using the **Show macros** command in the table editor; see paragraph 4.9.3.2.2.7.

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 4.10 and 4.9.1.6 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).

## 4.6 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 a project's data tables, groups, and to the project as a whole.

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 root 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.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 28 of 295

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 4.9.3.9.

Some data associated with a project is applicable to the entire project. Examples include the project's name and description, or the number of central processing units (CPUs) supported. For these cases the project-level data fields are appropriate. Paragraph 4.9.3.14 provides information on adding data fields at the project level.

#### 4.6.1 Data field editor

This section provides details on use of the data field editor (Figure 7). See paragraph 4.9.3.10.4.1 for information regarding adding default data fields to table types, paragraph 4.9.3.2.5.1 for information on adding fields to a particular table, and paragraph 4.9.3.14 for information on adding project-level fields. In each case the editor operation is the same.

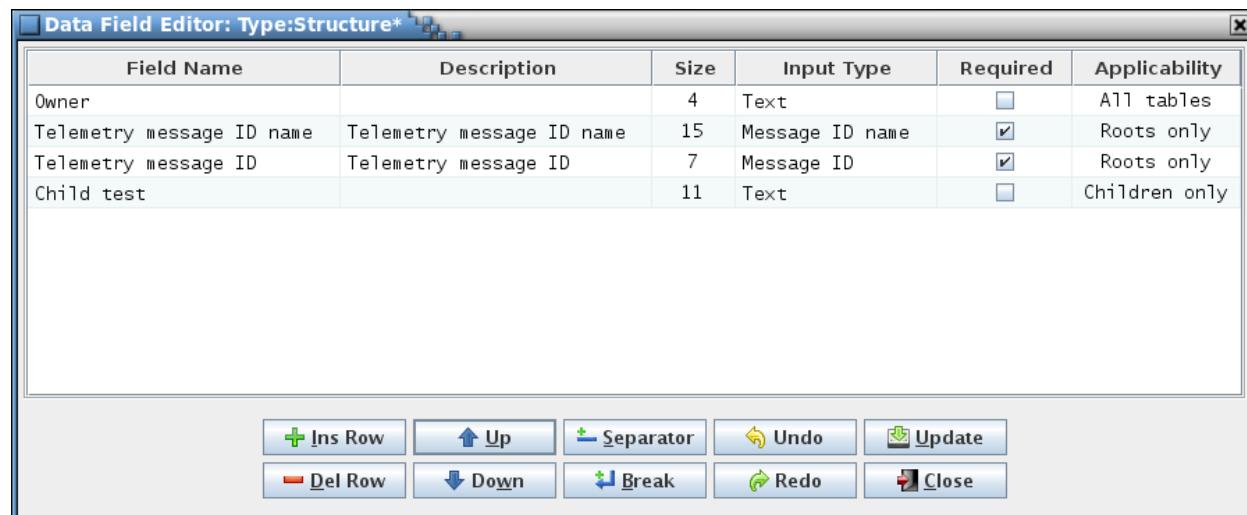


Figure 7. Data field editor

There are five or six columns in the field editor (depending on context):

**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 data field is displayed the field name is the label shown immediately to the left of the input 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 4.10). The field name is required.

**Description** The field description is used 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, group, 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 29 of 295

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 4.7 for information on the available input types. A field with a boolean input type is displayed as a check box. The input type defaults to “text.”

**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. A boolean (check box) data field does not display the highlighting.

**Applicability** This column only appears when assigning default data fields in the table type editor for a structure table type, or when editing the fields for a prototype or root structure table.

If the data field editor is invoked from the table type editor then it allows defining fields that are propagated to new and existing structure tables. Select the applicability for a field from the combo box pull down menu that appears when the applicability cell is selected. *All tables* indicates that the field will be added to all structure tables (root and child tables), *Roots only* implies that only root tables have the field added, and *Children only* means that only child tables have the field added. The default is *All tables*.

If the data field editor is invoked from a prototype or root structure table then all fields assigned to the table, regardless of the applicability setting, are displayed in the editor. For example, if the table has both *Roots only* and *Children only* fields assigned then only one of these types is displayed in the table’s editor since a table can only be a root or a child, but not both. However, a table can change from a root to a child, and vice versa; when this happens the types of fields that are displayed are changed to match. The field editor displays all of the fields, whether applicable or not, so that these can be altered by the user.

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 as described in paragraph 4.2; however, the sorted order does not dictate the field display 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, group manager, 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:

**Ins Row** The editor is initially empty unless the table editor, group manager, or table type editor from which it’s invoked has any previously defined fields. To add a field first select the **Ins Row** 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.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 30 of 295

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 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 <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>	Applies the data field definitions currently displayed in the field editor to the table editor, group manager, or table type editor from which the field editor was invoked. Any existing fields that are in the table editor, group manager, 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 <b>Store</b> button in the table editor, group manager, or table type editor is selected.
<b>Close</b>	Closes the field editor window. If any changes made have not been applied using the <b>Update</b> button then a dialog appears allowing the user to confirm discarding the updates or to cancel closing the editor.

## 4.7 Input Types

Data table columns, data fields, and macros are assigned an input type in the table type editor (see paragraph 4.9.3.10) and the data field editor (see paragraph 4.6.1) respectively. The input type constrains the type of value entered into a data table cell or data field. Leading and trailing white space characters (spaces, tabs, etc.) are ignored and eliminated when the text is formatted (except for the two Text input types that specifically preserve these characters; spaces between characters in a text string are always retained). Leading plus (+) signs and zeroes are allowed for non-negative integer and floating point values, but are ignored and eliminated. In the table type or data field 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>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>Alphanumeric (multi)</b>	Allows multiple <b>Alphanumeric</b> inputs, separated by one or more white space character(s).
<b>Argument name</b>	Special format used to designate a command table argument name column. This type has the same constraints as the <b>Alphanumeric</b> type.
<b>Array index</b>	Special format used to designate the array size column. Allows one or more integer values (each greater than 1), separated by commas. For the array

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 31 of 295

	<p>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[2][3][4]</i>).</p>
<b>Bit length</b>	Special format used to designate the structure table bit length column. This type has the same constraints as the <b>Positive integer</b> type.
<b>Boolean</b>	Only “true” or “false” (case insensitive) are allowed for the boolean type in data tables. A check box represents a data field of this type.
<b>Command code</b>	Special format used to designate the command table command code column. This type has the same constraints as the <b>Hexadecimal</b> type.
<b>Command name</b>	Special format used to designate the command table command name column. This type has the same constraints as the <b>Alphanumeric</b> type.
<b>Description</b>	Special format used to designate a description column. This type has the same constraints as the <b>Text</b> type.
<b>Enumeration</b>	Special format used to designate a column containing enumerated values. This type allows letters, numerals, and punctuation characters.
<b>Floating point</b>	This type allows floating point values to be entered; i.e., values with decimal components in the form “#.###” (e.g., 3.14).
<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>Hexadecimal range</b>	Allows one or two <b>Hexadecimal</b> values. If two values are entered they must be separated by a hyphen (-).
<b>Integer</b>	The integer data type allows input of any integer value: positive, negative, or zero.
<b>Integer &gt; 1</b>	Only integer values greater than 1 are allowed for this type.
<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>Message ID</b>	Used to designate a table column or data field as representing a telemetry or command message ID number. The constraints are identical to the <b>Hexadecimal</b> type, except that the ‘#’ character may be appended to the ID number. This character flags the message ID as being protected from overwriting by the automatic message ID assignment command. Assigning this input type to a table column or data field enables the application to pair message IDs and ID names; see paragraph 4.9.3.13.3 for more information on how this input type is used.
<b>Message ID name</b>	Used to designate a table column or data field as representing a telemetry or command message ID name. The constraints are identical to the <b>Alphanumeric</b> type. Assigning this type to a table column or data field

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 32 of 295

	enables the application to pair message ID names and IDs; see paragraph 4.9.3.13.3 for more information on how this input type is used.
<b>Message names &amp; IDs</b>	This type causes a combo box pull down menu to appear when the table cell or data field is selected. The menu contains all of the currently defined message ID names and associated message ID numbers. When a message ID name and number is selected only the ID name appears in the cell or field.
<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>Negative float</b>	Similar to the <b>Floating point</b> type, except that only negative values are allowed.
<b>Negative integer</b>	Similar to the <b>Integer</b> type, except that only negative integer values are allowed.
<b>Non-negative float</b>	Similar to the <b>Floating point</b> type, except that only zero or negative values are allowed.
<b>Non-negative integer</b>	Similar to the <b>Integer</b> type, except that only zero or negative values are allowed.
<b>Non-zero hexadecimal</b>	Similar to the <b>Hexadecimal</b> type, except that a value of zero (0x0) is excluded.
<b>Positive float</b>	Similar to the <b>Floating point</b> type, except that only positive, non-zero values are allowed.
<b>Positive integer</b>	Similar to the <b>Integer</b> type, except that only positive, non-zero integers are allowed.
<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 <b>Data Type</b> column in structure tables but can be used elsewhere. This selection is not available in the data field editor.
<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.
<b>Text</b>	This type allows letters, numerals, and punctuation characters. <b>Text</b> is the default data type.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
	<i>Version 1.2.31</i>	Page 33 of 295

<b>Text (multi-line)</b>	<p>This type allows letters, numerals, punctuation, and new line characters, the latter allowing for multiple lines of text in a data field. The field initially displays as a single row, but the number of rows increases as new line characters are entered. This type behaves identically to <b>Text</b> if used as a table column's input type.</p>
<b>Text (spaces)</b>	<p>Identical to the <b>Text</b> type except that leading and trailing white space characters are preserved.</p>
<b>Text (multi-line, spaces)</b>	<p>Identical to the <b>Text (multi-line)</b> type except that leading and trailing white space characters are preserved. This type behaves identically to <b>Text (spaces)</b> if used as a table column's input type.</p>
<b>Units</b>	<p>Special format used to designate a column containing units (e.g., degrees F, rpm, m/s). This type has the same constraints as the <b>Text</b> type.</p>
<b>Variable name</b>	<p>Special format used to designate a variable name column. This type has the same constraints as the <b>Alphanumeric</b> type.</p>
<b>Variable path</b>	<p>This is a special format valid only in a structure table type definition. If a column in the table is assigned this type then the table's path is combined with the variable name and data type to produce the variable's full path. The column can be edited, in which case the manually entered value overrides the automatically generated path. The characters that separate each structure reference in the path, and the data types and variable names, and whether or not the data type is displayed, are set using the <b>Show variables</b> command menu command; see paragraph 4.9.3.16.</p>
<b>XML: Application ID</b>	<p>Name of the variable in the telemetry and command header structure tables that contains the application ID. This type has the same constraints as the <b>Alphanumeric</b> type. Used during XML import and export to determine the location from which to retrieve or where to store the application ID.</p>
<b>XML: Command Header</b>	<p>Name of the structure table representing the command header. This type has the same constraints as the <b>Alphanumeric</b> type. Used during XML import and export to indicate the command header table.</p>
<b>XML: Function Code</b>	<p>Name of the variable in the command header structure table that contains the command function code. This type has the same constraints as the <b>Alphanumeric</b> type. Used during XML import and export to determine the location from which to retrieve or where to store the command function code.</p>
<b>XML: Telemetry Header</b>	<p>Name of the structure table representing the telemetry header. This type has the same constraints as the <b>Alphanumeric</b> type. Used during XML import and export to indicate the telemetry header table.</p>

## 4.8 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, radio, etc. Each data stream has its own set of rate parameters (see paragraph 4.9.4.4). Addition of a stream is accomplished by adding a new sample rate column to a structure table definition (see paragraph 4.9.3.10 for information on altering a table type). A rate column is designated by assigning the column an input type of 'Rate' (see paragraph 4.7 for information on input types). A telemetry parameter can be assigned a rate in each of the defined

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 34 of 295

data streams. The link manager (paragraph 4.9.4.1) allows linking telemetry parameters for allocation in the downlink messages. These linkages are specific to a data stream. The data stream can be assigned a name different from its associated rate column name in the rate parameter dialog (paragraph 4.9.4.4).

## 4.9 Command Menu

The following paragraphs describe the main application window's menu commands.

### 4.9.1 File menu

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

#### 4.9.1.1 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 to discard the unsaved changes and continue with the user change, 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 8, 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 similar to that in Figure 9. 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 database is closed. Select the **Cancel** button to exit the dialog without changing the user.



Figure 8. Select User dialog (no server connection)

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 35 of 295
---	---	----------------------------------

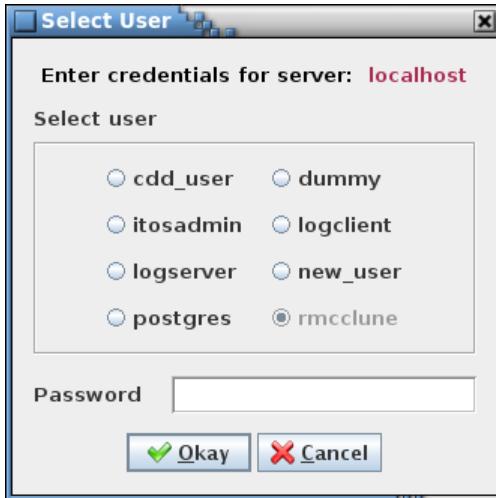


Figure 9. Select User dialog (server connected)

#### 4.9.1.2 Database server

The **Database server** command is used to set the PostgreSQL database connection properties. When the 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 to discard the unsaved changes and continue with the server change, or to cancel the server 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 10). The dialog allows entering the PostgreSQL server host name and server port number, and to enable or disable secure sockets layer (SSL) authentication.



Figure 10. Database server dialog

The dialog allows selection of a PostgreSQL server on the local or a remote host. Enter the server host name and the server port number, set the **Enable SSL** check box appropriate to the target connection's encryption requirements, and then select the **Okay** button. The host text field uses auto-completion to fill in the host string. The previous 30 host names are remembered by default, including those from previous sessions (the number of remembered host names can be changed via the Preferences dialog). 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 to the server on the new host as the current user. Select the **Cancel** button to exit the dialog without changing the database server properties.

The SSL selection must match that of the target or else the connection will not be established. *Note: When SSL is enabled no certificate validation is performed by the JDBC driver.*

#### 4.9.1.3 Read log

The **Read log** command causes the **Open Event Log** file selection dialog to be displayed. Navigate to the location of the desired CCDD event log file, highlight it using the mouse or keyboard controls, and select

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 36 of 295

**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 without opening a log file. See paragraph 4.4 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 4.9.1.5); 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.

#### 4.9.1.4 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.

#### 4.9.1.5 Search log

The **Search log** dialog provides a means of searching the current session's event log for a specified text string (see Figure 11). Case sensitivity for the search is governed by the **Ignore text case** check box.

The **Allow regular expression** check box, when checked, allows the use of a regular expression to define the search pattern in the search text field. A regular expression can be formulated to find multiple matching conditions (for example, the search for **a.c** would match any string that has a single character between the characters 'a' and 'c'). Information regarding the use of regular expressions is beyond the scope of this document; however, resources and tutorials can be found online. When unchecked, the search text is matched as typed in the search text field.

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-F key sequence while the main application window has the focus also displays the event log search dialog.

The search text field uses auto-completion to fill in the search string. The previous search strings (those for the event log, tables, and scripts) are remembered, including those from previous sessions. The number of remembered search strings can be changed via the Preferences dialog, and defaults to 30. Case sensitivity for auto-completion is based on the **Ignore text case** check box selection state.

The input text can be changed and the **Search** button pressed again to initiate another search of the log. 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.

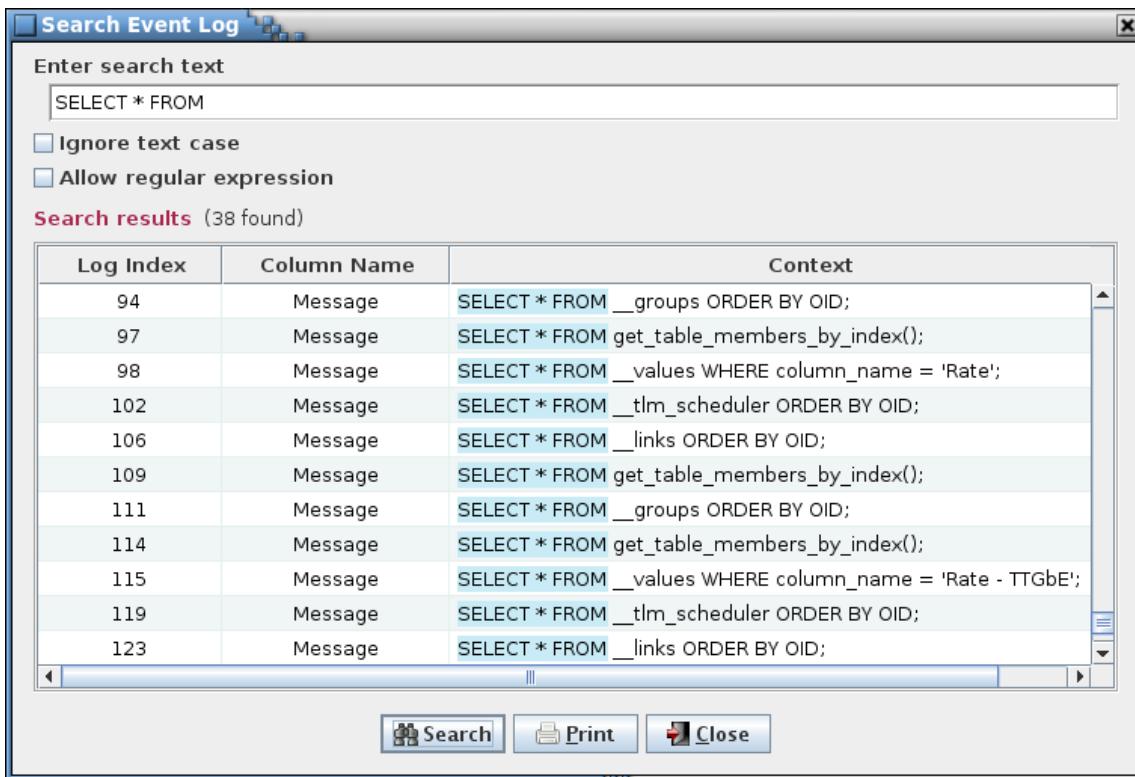


Figure 11. Search event log dialog

#### 4.9.1.6 Web server

The embedded 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 4.9.1.6.1) is used to start and stop the server. The server is disabled by default. 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. Lists containing the names of all tables, groups, or applications can be requested. Table 6 contains the recognized *component*, *attribute*, and *name* combinations.

The data is returned to the requesting application in JSON "key":"value" pair 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 read access to the project open in the CCDD application hosting the web server.

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 4.5.7 and 4.9.3.12 for more information relating to macros). Requests can be made to return the table data with the macro names

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 38 of 295

displayed in place of the macro values by appending ;**macro** (or ;**macros** – either is case insensitive) at the end of the *name* portion of the request, or at the end of the *attribute* portion if there is no *name* portion. For example, the request *table?\_=;macro* returns the table data for all tables with the macro names displayed, and the request *telemetry?\_=;macros* returns the telemetry information for all tables with the macro names displayed in those JSON “key”:“value” pairs that are taken directly from table cells.

Variable paths, formatted as single strings, can be requested as part of a structure table’s data (see paragraph 4.9.3.16 for more information on variable paths). The variable paths are attached to the table data as an additional column named “Variable Path”. The format for returning the paths is similar to the **variable** command (see Table 6):

*;path<s><,<variable path member separator character(s)>,<hide data types flag (true or false)>,<data type and variable name separator character(s)>>*

If the variable path member separator character(s), hide data type flag, and data type and variable name separator character(s) are not provided then the default values – “\_”, false, and “\_”, respectively – are used. The command is case insensitive and must be placed at the end of the *name* portion of the request, or at the end of the *attribute* portion if there is no *name* portion. It may be used in conjunction with the **macro** command described above. For example, the request:

*table?data=;path,\_,"false",."*

would convert the variable path *rootStruct,parentStruct.structVar,int.primVar* to *rootStruct\_parentStruct.structVar\_int.primVar*, and the request:

*table?data=;path,\_,"true,\_"*

would convert the variable path *rootStruct,parentStruct.structVar,int.primVar* to *rootStruct\_structVar\_primVar*.

Request			Returned Information	Output <sup>1,2</sup>
Component	Attribute	Name		
table	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), or for all tables if <i>table name</i> is omitted	{"Table Name":" <i>table name</i> ","Table Type":" <i>type</i> ","Table Description":" <i>description</i> ","Table Size": <i>size</i> ,"Table Data":[{"first row column name":" <i>first row column value</i> ,<,"first row next column name":" <i>first row next column value</i> {“second row column name”：“ <i>second row column value</i> ”,<,”second row next column name”：“ <i>second row next column value</i> ”<,...>}<,...>],"Data Field":[{"Field Name":" <i>field name</i> ","Description":" <i>field description</i> ","Size":" <i>field character length</i> ","Input Type":" <i>input data type</i> ","Required":"true or false","Applicability":" <i>field applicability</i> ","Value":" <i>field value</i> "}<,next field's data...>]}
	data	<table name>	Data for the specified data table ( <i>table name</i> is case insensitive), or for all tables if <i>table name</i> is omitted	{"Table Name":" <i>table name</i> ","Table Data":[{"first row column name":" <i>first row column value</i> ,<,"first row next column name":" <i>first row next column value</i> {“second row column name”：“ <i>second row column value</i> ”,<,”second row next column name”：“ <i>second row next column value</i> ”<,...>}<,...>}]}
	description	<table name>	Description for the specified table ( <i>table name</i> is case insensitive), or for all tables if <i>table name</i> is omitted	{"Table Name":" <i>table name</i> ","Table Description":" <i>description</i> "}

Request			Returned Information	Output <sup>1,2</sup>
Component	Attribute	Name		
	fields	<table name>	Data field information for the specified table ( <i>table name</i> is case insensitive), or for all tables if <i>table name</i> is omitted	{"Table Name":" <i>table name</i> ","Data Field": [{"Field Name":" <i>field name</i> ","Description":" <i>field description</i> ","Size":" <i>field character length</i> ","Input Type":" <i>input data type</i> ","Required":"true or false","Applicability":" <i>field applicability</i> ","Value":" <i>field value</i> "}<,next field's data...>]}
	names	<table type>	Names of all tables of the specified table type ( <i>table name</i> is case insensitive), or for all table types if <i>table type</i> is omitted	{"Table Type":" <i>table type</i> ","Table Names":[" <i>table name</i> "<,"next table name"<,...>]}
	size	<table name>	Size (in bytes) for the specified structure table ( <i>table name</i> is case insensitive), or for all tables if <i>table name</i> is omitted	{"Table Name":" <i>table name</i> ","Byte Size": <i>size</i> }
	search	<search text>,<ignore case (true or false)>,<allow regular expression (true or false)>,<data table cells only (true or false)><,search table column names>	Search for the specified text in the project database tables. Set the <i>ignore case</i> flag to true to ignore case when matching the text (defaults to false if not provided). Set the <i>allow regular expression</i> flag to allow a regular expression in the search text. Set the <i>data table cells only</i> flag to return only those matches within the data table cells (otherwise matches in the internal tables are returned as well; defaults to false if not provided). Optionally constrain the search to specific data table columns by appending a comma-separated list of column names (the names are case sensitive)	[{"Table / Object":" <i>table name</i> ","Location":" <i>location within table</i> ","Context":" <i>table cell contents containing the matching search text</i> "}<,...>]

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 41 of 295	

Request			Returned Information	Output <sup>1,2</sup>
Component	Attribute	Name		
proto_table		<i>Same requests as for table above, except only table information for prototype tables is returned</i>		
root_table		<i>Same requests as for table above, except only table information for root tables is returned</i>		
instance_table		<i>Same requests as for table above, except only table information for instance tables is returned</i>		
group	all (or blank)	<group name>	Description, associated table(s), and data field(s) for the specified group, or for all groups if <i>group name</i> is omitted	{"Group Name":" <i>group name</i> ","Group Description":" <i>description</i> ","Group Table":[" <i>table name</i> <,"next table name<,...>"],"Group Data Field": [{"Field Name":" <i>field name</i> ","Description":" <i>field description</i> ","Size":" <i>field character length</i> ","Input Type":" <i>input data type</i> ","Required":"true or false","Applicability":" <i>field applicability</i> ","Value":" <i>field value</i> "}<,next field's data...>]}
	tables	<group name>	Table(s) associated with the specified group ( <i>group name</i> is case sensitive), or for all groups if <i>group name</i> is omitted	{"Group Name":" <i>group name</i> ","Group Table":[" <i>table name</i> <,"next table name<,...>"]}
	description	<group name>	Description for the specified group ( <i>group name</i> is case sensitive), or for all groups if <i>group name</i> is omitted	{"Group Name":" <i>group name</i> ","Group Description":" <i>group description</i> "}
	fields	<group name>	Data field information for the specified group ( <i>group name</i> is case sensitive), or for all groups if <i>group name</i> is omitted	{"Group Name":" <i>group name</i> ","Group Data Field": [{"Field Name":" <i>field name</i> ","Description":" <i>field description</i> ","Size":" <i>field character length</i> ","Input Type":" <i>input data type</i> ","Required":"true or false","Applicability":" <i>field applicability</i> ","Value":" <i>field value</i> "}<,next field's data...>]}
	names		Names of all groups	{"Group Names":[" <i>first group name</i> <," <i>second group name</i> <,...>"]}

Request			Returned Information	Output <sup>1,2</sup>
Component	Attribute	Name		
application	<i>Same requests as for group above, except only group information for groups representing a CFS application is returned</i>			
scheduler	telemetry	<data stream name>, <header size (bytes)>, <message ID name data field name>, <optimize flag (true or false)>	Telemetry scheduler's copy table entries for the specified data stream	{"Stream Name":"stream name","Header Size":"size","Optimized":"true or false","Copy Table":[{"Input Message ID":"input ID","Input Offset":"input byte offset","Output Message ID":"output ID","Output Offset":"output byte offset","Number of Bytes":"output size in bytes","Root Table":"root table name","Variable Path":"variable path"}<,next row<,...>>]}
	application		Application scheduler's schedule table entries	<i>This command is currently ignored</i>
variable	<variable path + name> or blank	<variable path member separator character(s)>, <hide data types flag (true or false)>, <data type and variable name separator character(s)>	The variable path and name in the format used by the application paired with the converted path and name (dependent on the input parameters). If no variable path and name is supplied then every variable in the project along with its converted form is returned	{"variable path and name (application format)":"variable path and name (user-specified format)"[,...]}

Request			Returned Information	Output <sup>1,2</sup>
Component	Attribute	Name		
telemetry	<>group name>, <data stream name>, <sample rate>>		Structure table name, variable name, data type, data stream information, enumeration information, and all other data associated with the variable for the telemetered variables in the structure tables belonging to the specified group (or application), or for all structure tables if <i>group name</i> is omitted. Filters can be specified for the data stream name and/or the sample rate so that only those telemetered variables matching the filter(s) are returned	{"Structure Table Name":"table name from which the variable is taken","Variable Name":"variable name","Data Type":"variable's data type","Data Streams":[{"Stream Name":"first stream's name","Sample Rate":"variable's sample rate in this stream"}], "next stream's data<,...>>,"Enumerations":[{"Enumeration Name":"first enumeration's name","Enumeration Value":"first enumeration's value"}], "next enumeration's data<,...>>]}
command	all (or blank)	<group name>	Command table name, command name, command code, command description, and for each of the command's arguments the argument name, data type, enumeration, minimum value, maximum value, and any other argument data, for the commands in the command tables belonging to the specified group (or application), or for all command tables if <i>group name</i> is omitted	{"Command Table Name":"table name from which the command is taken","command name column name":"command name", "command code column name":"command code"}, "command description column name":"command description">,"Arguments": [{"first argument's name column name":"first argument's name", "first argument's data type column name":"first argument's data type", "first argument's enumeration column name":"first argument's enumeration", "first argument's minimum value column name":"first argument's minimum value", "first argument's maximum value column name":"first argument's maximum value"}], "next argument's data<,...>>]}

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 44 of 295	

Request			Returned Information	Output <sup>1,2</sup>
Component	Attribute	Name		
table_type			Table type definitions	{"Table Type Definition": [{"Column Name": "type name", "Description": "type description", "Input Type": "input data type", "Unique": "true or false", "Required": "true or false", "Enable if Structure": "true or false", "Enable if Pointer": "true or false"}]}
data_type			Data type definitions	{"Data Type Definition": [{"Type Name": "user-defined name", "C Name": "C-language name", "Size": "size in bytes", "Base Type": "base data type"}], "next data type definition": ...}
macro			Macro definitions	{"Macro Definition": [{"Macro Name": "macro name", "Value": "macro value"}], "next macro definition": ...}
message_id			Message ID owners, names, and values	{"Message ID Owner, Name, and Value": [{"Owner": "message ID owner", "Message ID Name": "message ID name", "Message ID": "message ID value"}], "next message ID": ...}
project_info			Get the active project's information (name, database name, description, lock status, current user name, project owner name, and project data fields)	{"Project": "project name", "Database": "database name", "Description": "project description", "Status": "locked or unlocked", "User": "user name", "Owner": "owner name", "Project Data Field": [{"Field Name": "field name", "Description": "field description", "Size": "field character length", "Input Type": "input data type", "Required": "true or false", "Applicability": "field applicability", "Value": "field value"}], "next field's data": ...}

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 45 of 295

Request			Returned Information	Output <sup>1,2</sup>
Component	Attribute	Name		
authenticate	<i>user name</i>	<i>user password</i>	Determine if the specified user credentials are valid for the currently open project database	true <i>if the credentials are valid; false otherwise</i>
shutdown			Close the web server and project database, and exit the CCDD application	

- 1 *Only those JSON “key”：“value” pairs that are members of a JSON array (i.e., that are enclosed by brackets ([])) in the output have their original order preserved; other pairs may appear in any order.*
- 2 *If Name is omitted in a table, proto\_table, root\_table, instance\_table, group, or application request then a JSON array is returned in the format [<first output><,second output<,...>], with each array member representing a table, group, or application.*

Table 6. Web data access commands

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 46 of 295

#### 4.9.1.6.1 Enable server

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

#### 4.9.1.6.2 Select port

The **Select port** command displays a dialog (Figure 12) that allows selection of a port number for the embedded 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 12. Web Server dialog

#### 4.9.1.7 Preferences

The **Preferences** command displays the **Preferences** dialog (Figure 13), which allows altering the application's display characteristics. This includes choosing the "look and feel" (L&F), fonts, colors, sizes, and spacing values applied to the program's GUI components. The dialog has a separate tab for each of the selection types. Once updates to the preferences are completed press the **Close** button to exit the dialog. The preference changes are saved so that the updated characteristics are used when the application is opened again. See Appendix E.2 for details on the stored preferences.

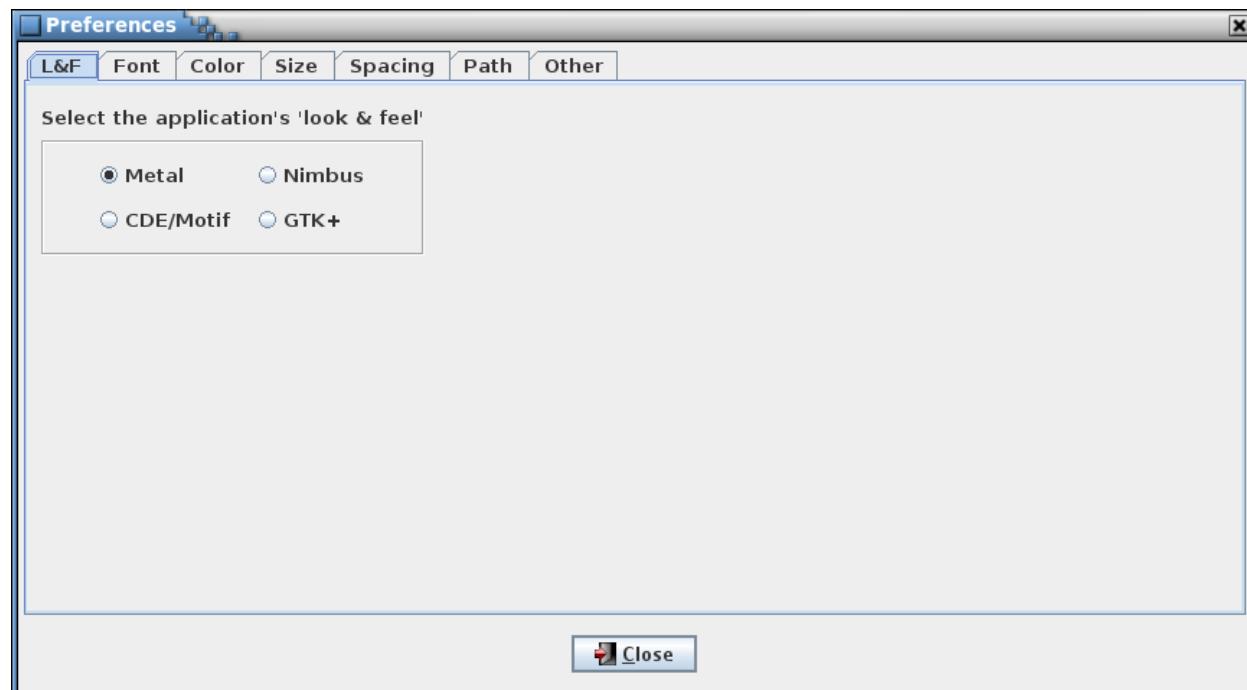


Figure 13. Preferences dialog; look and feel preferences

##### 4.9.1.7.1 L&F

The L&F can be changed via the **L&F** tab (Figure 13). L&Fs change the shape and color scheme of the graphical components (see Figure 14), though the basic layout remains the same. The default is

“Metal”, the standard L&F provided with Java. The 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 **Preferences** dialog, main application window, and any other open CCDD windows are immediately redrawn to reflect the L&F chosen.

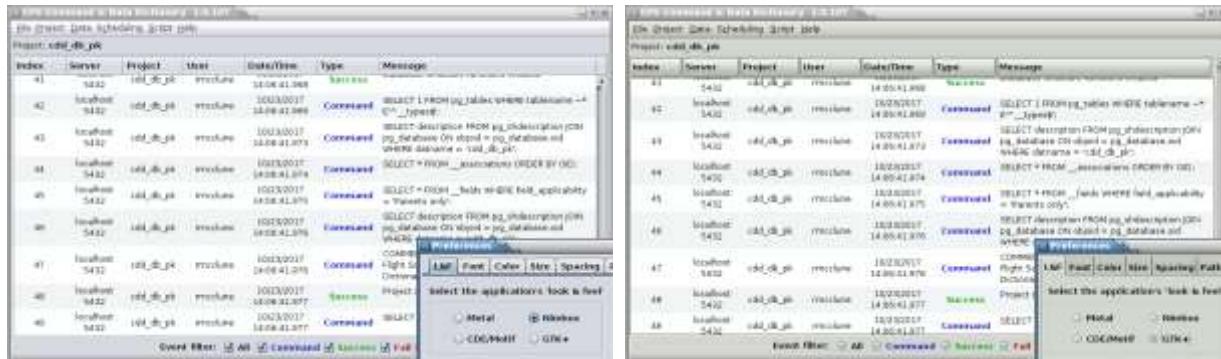


Figure 14. Example look and feel differences

#### 4.9.1.7.2 Font

The **Font** tab displays the selection controls for the various fonts used in the application (Figure 15).

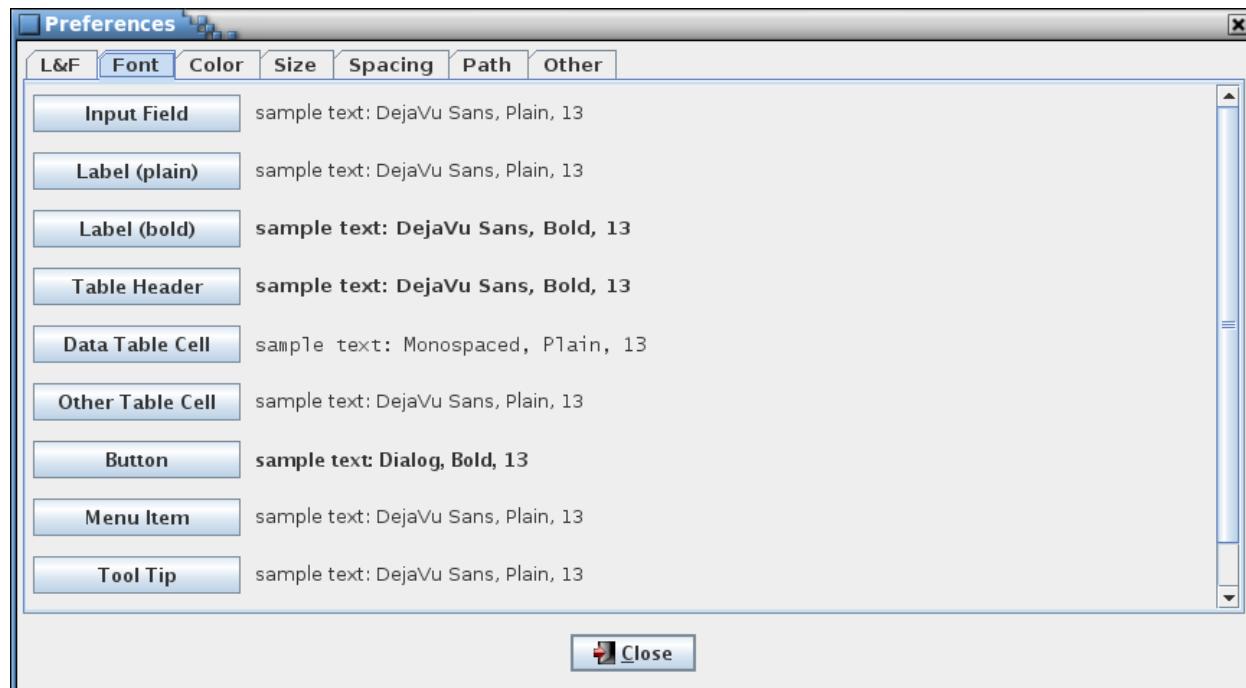


Figure 15. Font preferences

Each font is represented by a button with a sample of the font’s text beside it. Hovering the mouse pointer over the font button displays a tool tip describing the font’s use in more detail. To alter a font press its associated button – a font selection dialog appears (Figure 16). The font, style, and size are selected from the lists; the Sample field displays a sample of text using the chosen font settings. Select **Update** to change the selected font. The font change is implemented immediately in all of the application’s open windows. Select **Default** to use the font’s default settings. Press the **Close** button to exit the font selection dialog.

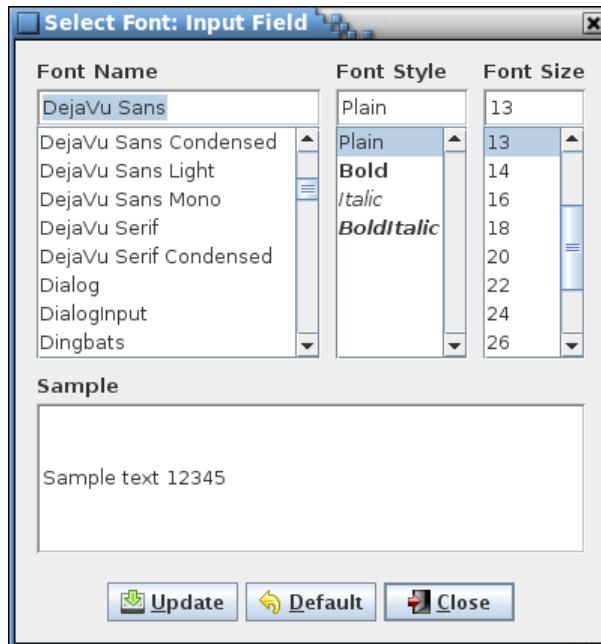


Figure 16. Font selection dialog

#### 4.9.1.7.3 Color

The **Color** tab displays the selection controls for the various colors used in the application (Figure 17).

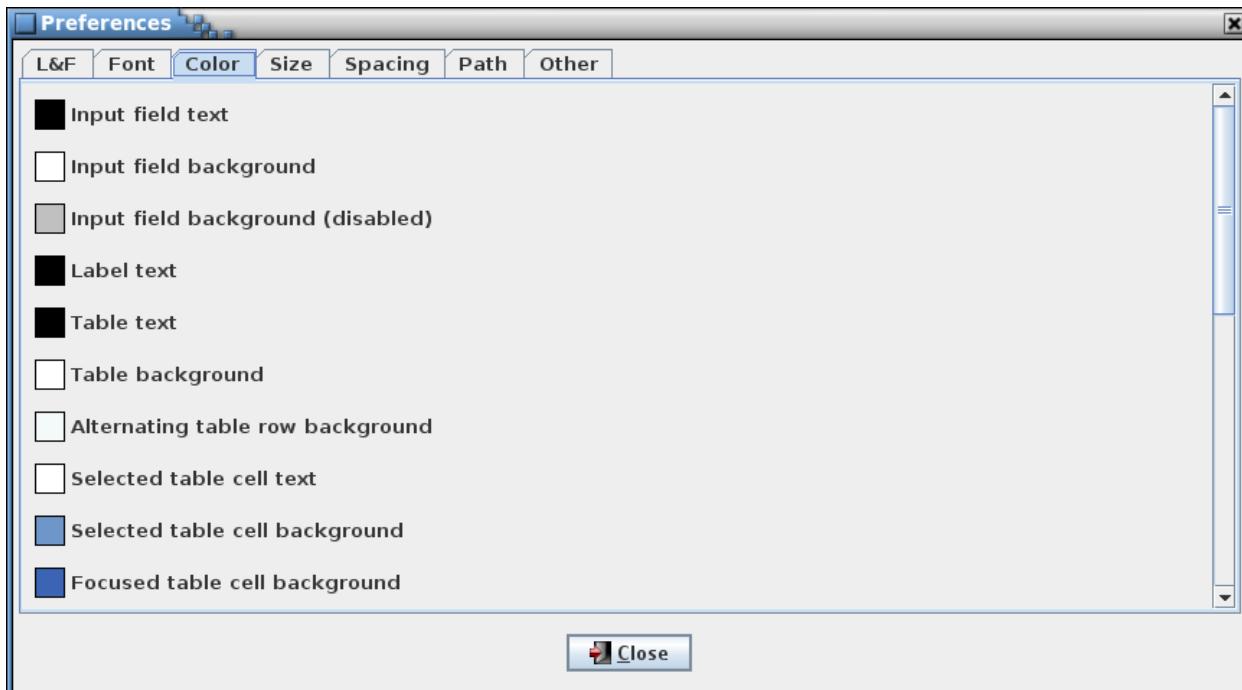


Figure 17. Color preferences

Each color is represented by a color box with a description beside it. Hovering the mouse pointer over the color box or description displays a tool tip describing the color's use in more detail. To alter a color select its associated box – a color selection dialog appears (Figure 18). The actual appearance of the color selection dialog varies based on the chosen L&F. However, the basic color selection for each is the same – use the mouse to operation alter the color parameters via the dialog's buttons, sliders, wheels, input fields, etc. Select **Update** to change the selected color. The color change is implemented

immediately in all of the application's open windows. Select **Default** to use the color's default settings. Press the **Close** button to exit the color selection dialog.

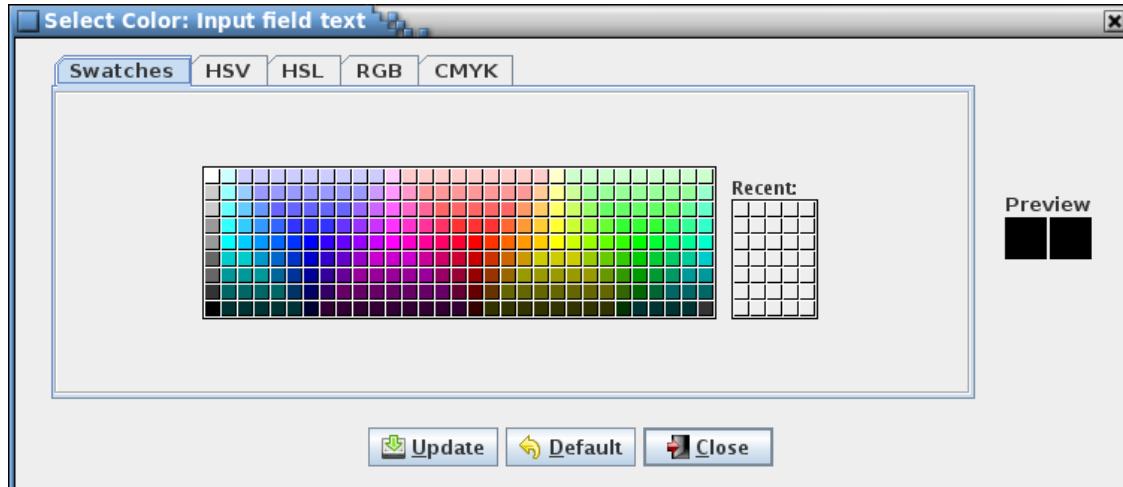


Figure 18. Color selection dialog

#### 4.9.1.7.4 Size

The **Size** tab displays the selection controls for the various size values used in the application (Figure 19).

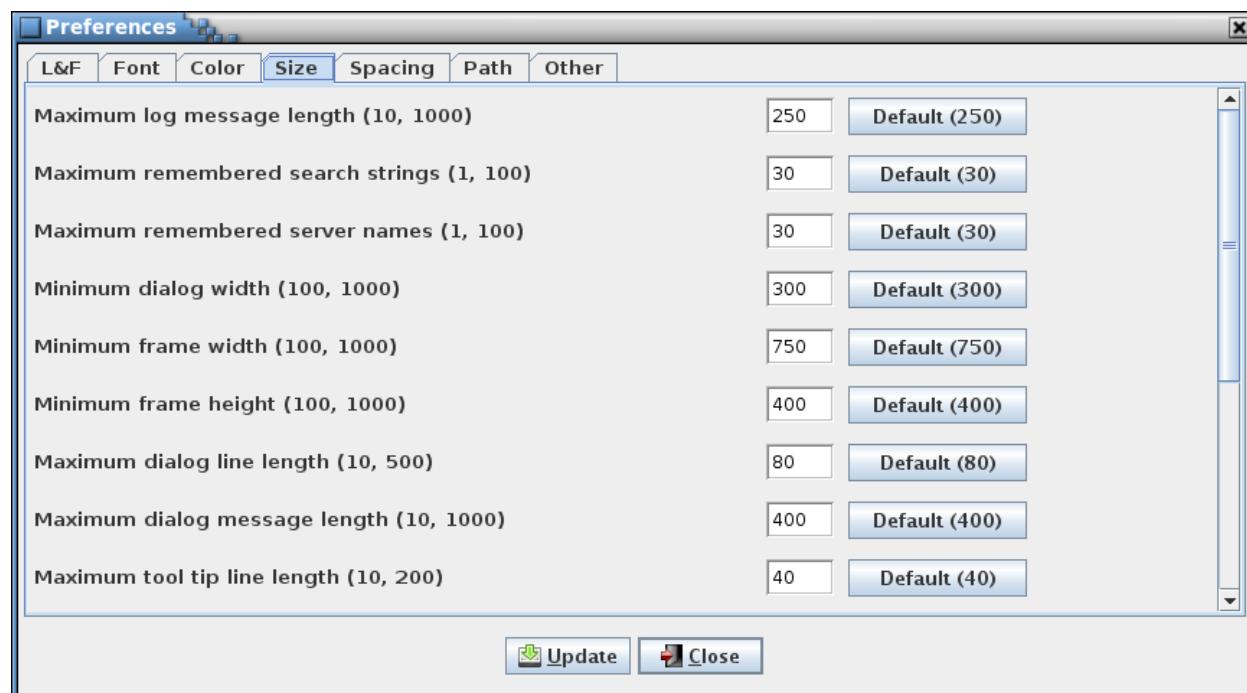


Figure 19. Size preferences

Each size value is represented by an input field with a description and default value button on either side of it. Hovering the mouse pointer over the description or input field displays a tool tip describing the size value's use in more detail. The description includes the minimum and maximum values allowed. To alter a size value enter a valid value into its associated input field. Selecting the size value's associated **Default** button (which displays the default value) enters the size value's default value into the input field. Select **Update** to implement the altered size value(s). Although size changes are implemented immediately, these are not apparent until the size value is used (e.g., the log message length applies to event messages subsequently logged).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 50 of 295

#### 4.9.1.7.5 Spacing

The **Spacing** tab displays the selection controls for the various spacing values used in the application (Figure 21).

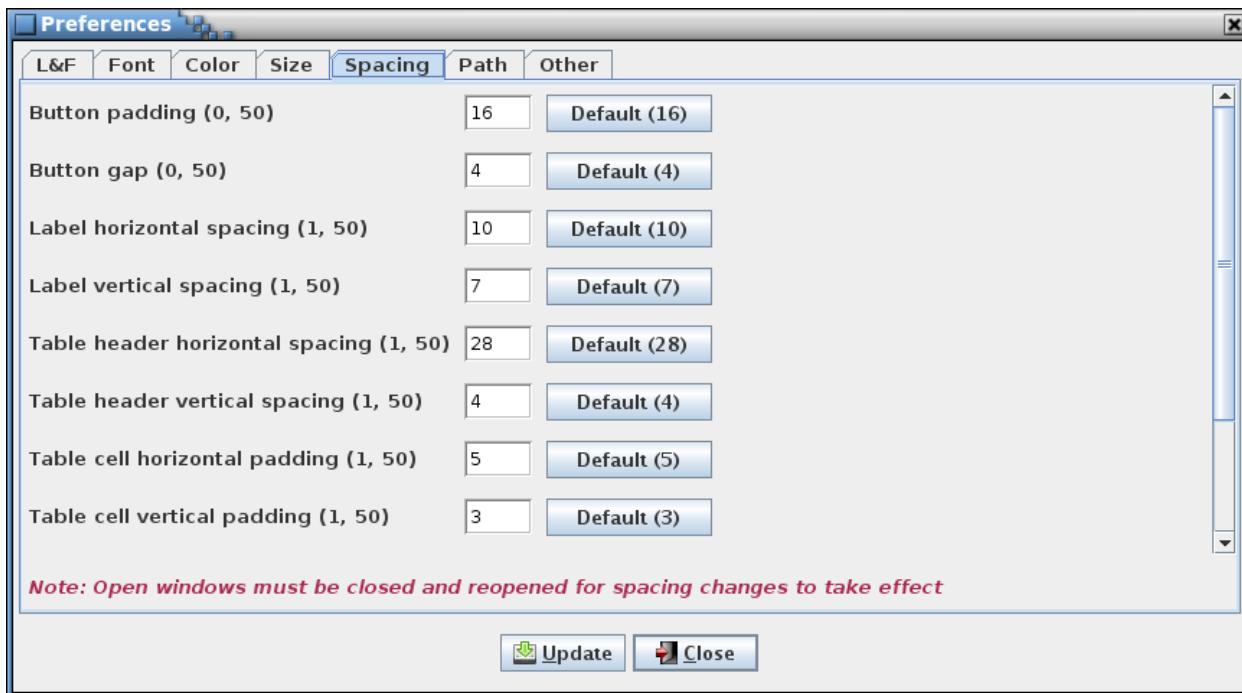


Figure 20. Spacing preferences

Each spacing value is represented by an input field with a description and default value button on either side of it. Hovering the mouse pointer over the description or input field displays a tool tip describing the spacing value's use in more detail. The description includes the minimum and maximum values allowed. To alter a spacing value enter a valid value into its associated input field. Selecting the spacing value's associated **Default** button (which displays the default value) enters the spacing value's default value into the input field. Select **Update** to implement the altered spacing value(s). As indicated by the note at the bottom of the dialog changes to spacing values are not applied to the application's open windows. These must be closed and reopened to use the updated values. Since this includes the main window, the application must be restarted in order for the spacing values to affect it.

#### 4.9.1.7.6 Path

The **Path** tab displays the selection controls for the various folder paths used in the application (Figure 21).

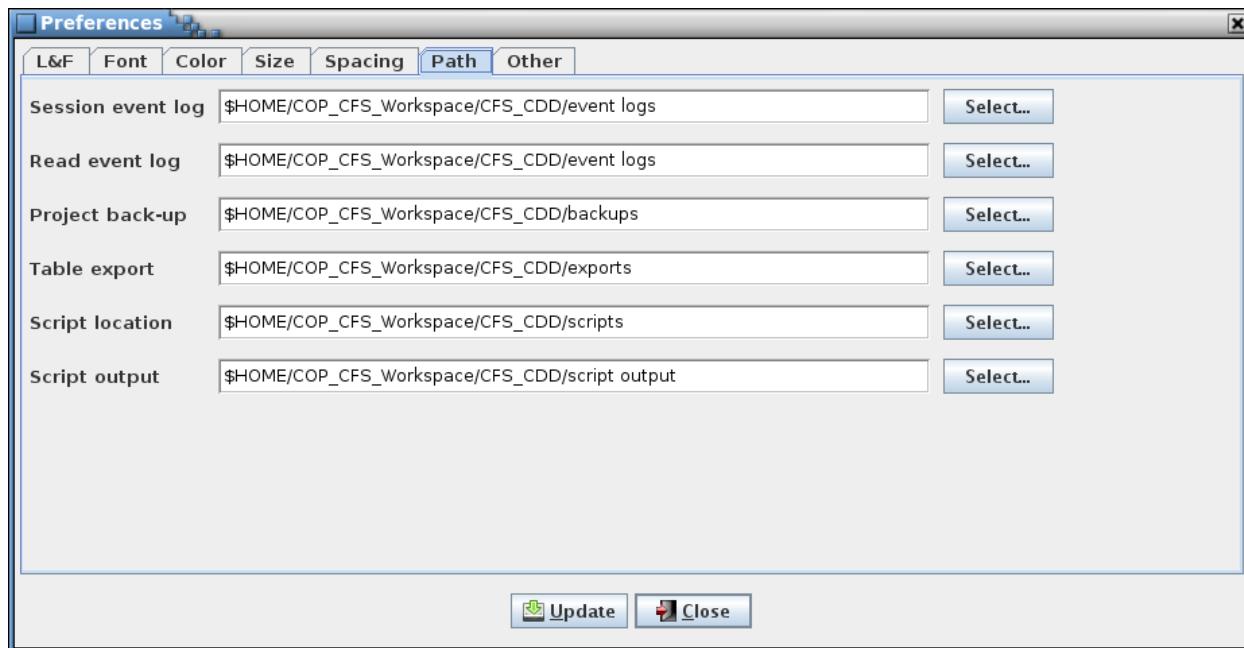


Figure 21. Path preferences

Each path is represented by an input field with a description and path selection button on either side of it. Hovering the mouse pointer over the description or input field displays a tool tip describing the path's use in more detail. To alter a path either enter it into its associated input field or press its **Select...** button to display a folder chooser dialog; once selected in the dialog the path appears in the input field. Select **Update** to implement the altered path(s). Paths can be absolute or relative. Relative paths are based on the folder in which the application is executed. Note that path selection in other dialogs (for example, the **Script Manager** or **Open Event Log** dialogs) updates the respective path if changed in the dialog. Path selection via command line argument also updates the path preference.

The **Script output** path is a special case. It does not directly affect the output location of files generated by scripts. Instead, a script may obtain the **Script output** path via an access method (see paragraph 4.10.7) and use it to set the location of any output file(s).

#### 4.9.1.7.7 Other

The **Other** tab displays program preferences that don't conform to those in the other tabs (Figure 22).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Date: May 2018

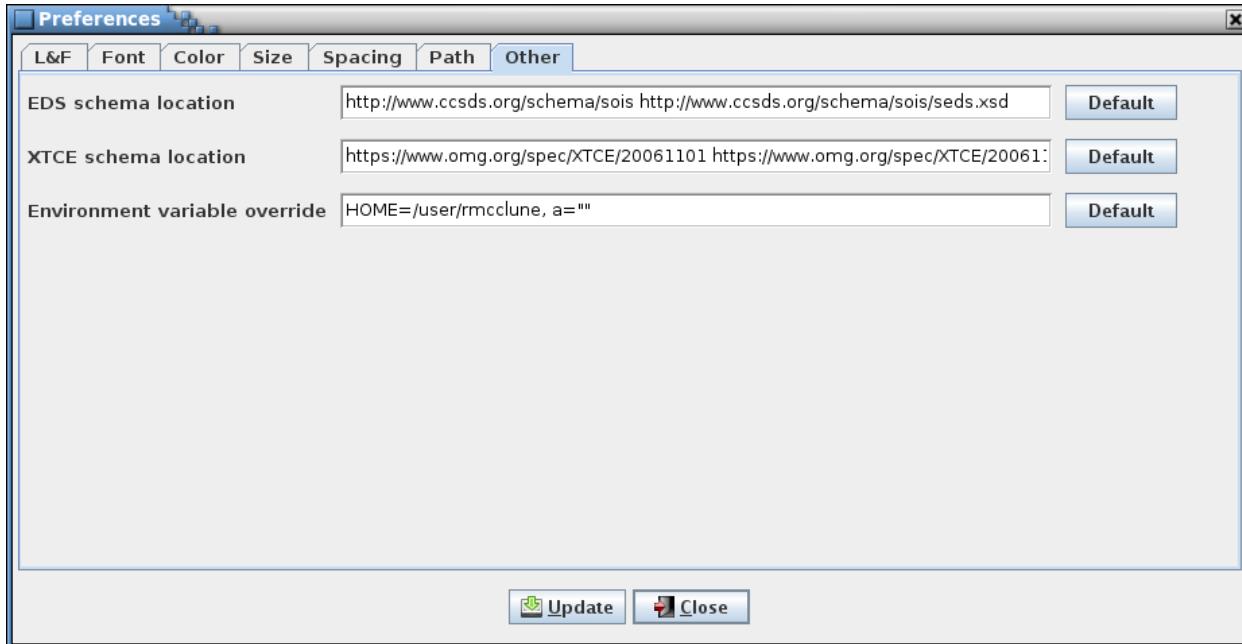


Figure 22. Other settings

Each setting is represented by an input field with a description and default selection button on either side of it. Hovering the mouse pointer over the description or input field displays a tool tip describing the setting's use in more detail. To alter a setting value enter a valid value into its associated input field. Selecting the setting's associated **Default** button enters the setting's default value into the input field. Select **Update** to store the altered setting value(s).

#### 4.9.1.8 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.

#### 4.9.2 Project menu

The **Project** menu contains commands for interacting with the project databases.

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 4.9.2.9) 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.*

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 53 of 295

#### 4.9.2.1 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 23). 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 associated radio button disabled. Select a project's radio button 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 to continue with the project change, discarding the unsaved changes, or to cancel the project change. Select **Cancel** to allow the currently opened project to remain open.

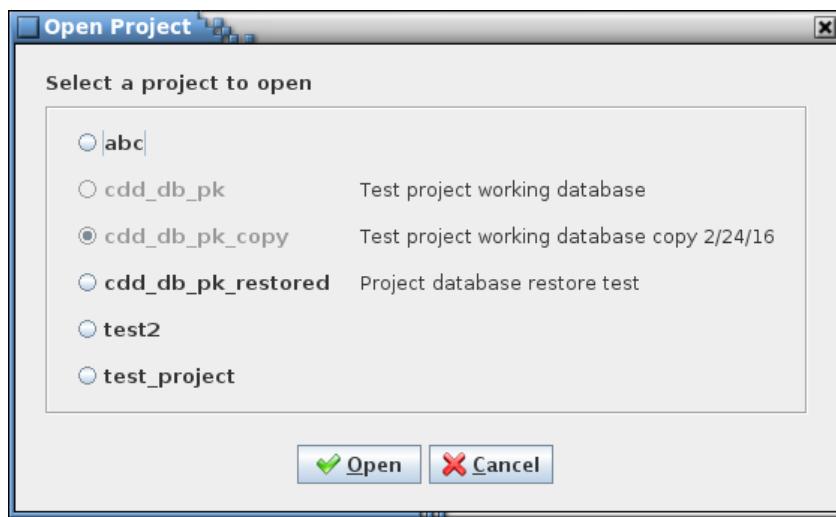


Figure 23. Select Project dialog

#### 4.9.2.2 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 to discard the unsaved changes and continue with closing the project, or to cancel closing the project.

#### 4.9.2.3 New

The **Create Project** dialog (see Figure 24) 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 that 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 super user status can access the project's database regardless of the owner. See paragraph 4.3 for further information regarding setup of the PostgreSQL server.

The project name can contain any character except a semi-colon (;). The project name is used to create the project's PostgreSQL database name. To meet the PostgreSQL naming constraints all letters are changed to lower case, any non-alphanumeric characters are converted to underscores, and an underscore is prepended to the resulting name if it begins with a numeral. If the resulting database

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
Doc. No. JSC-37494	<i>Version 1.2.31</i>	
Date: May 2018	Page 54 of 295	

name length exceeds 63 characters (the maximum allowed by PostgreSQL) then the name is truncated to the maximum length. The project and derived database names must be unique; these may not be the same as another project/database name existing in the server. The description can contain any character and, optionally, can be formatted using HTML tags.

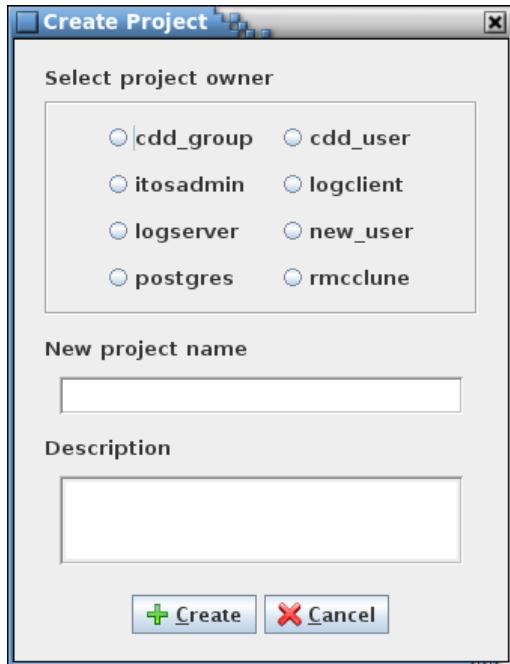


Figure 24. Create Project dialog

#### 4.9.2.4 Rename

When the **Rename** command is selected the **Rename Project** dialog is displayed (see Figure 25). 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 4.9.2.3 for constraints on the project name and description. When the **Rename** button is selected the project and description are updated. If the currently open project is renamed any open table editors are closed; if an editor has unstored changes then a dialog first appears allowing the user to confirm discarding the changes or canceling the rename operation. Note that this dialog can, if desired, be used to alter only the project's description (the description can also be edited using the **Manage project fields** command in the **Data** menu; see paragraph 4.9.3.14). Selecting the **Cancel** button closes the dialog without making any alterations.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 55 of 295

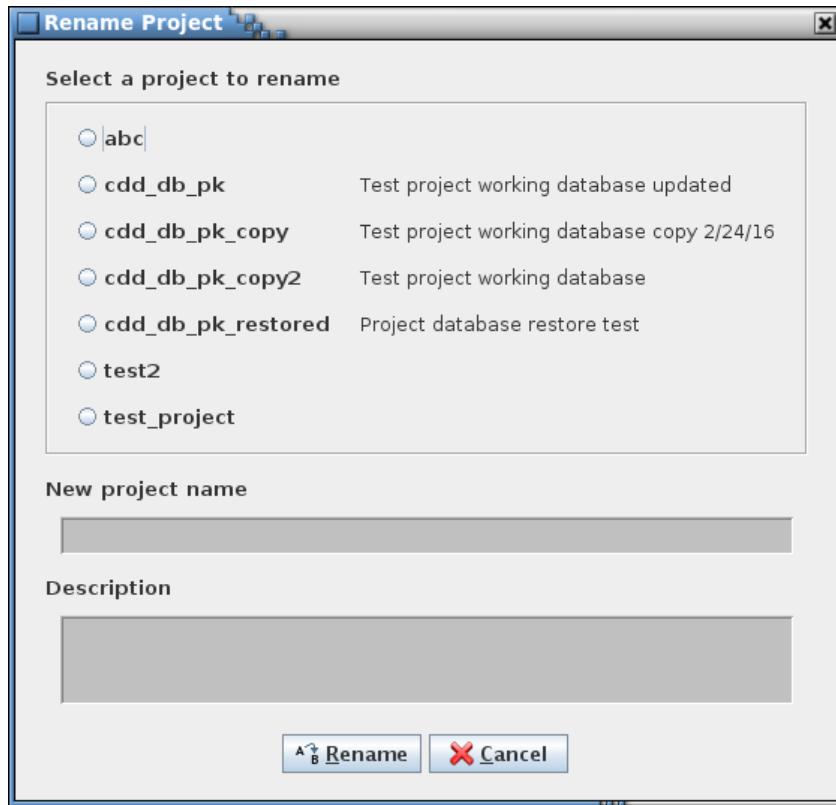


Figure 25. Rename Project dialog

#### 4.9.2.5 Copy

When the **Copy** command selected the **Copy Project** dialog is displayed (see Figure 26). 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 4.9.2.3 for constraints on the project name and description. The **Append date and time to project name** check box, if selected, replaces the text “\_copy” with the current date (year, month, and day) and time (hours, minutes, and seconds) stamp. Deselecting the check box removes the data and time stamp and appends the text “\_copy” again. 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 Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Version 1.2.31

Date: May 2018

Page 56 of 295

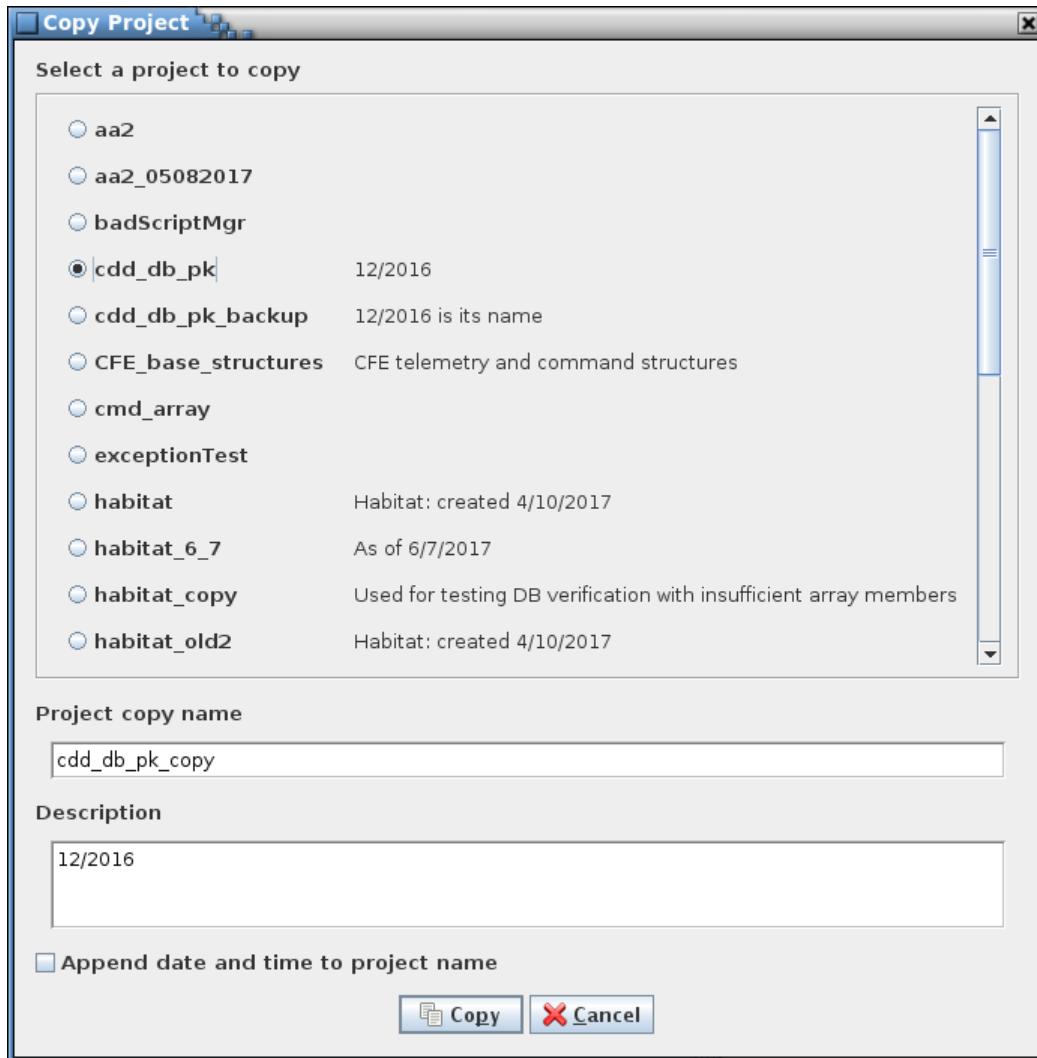


Figure 26. Copy Project dialog

#### 4.9.2.6 Delete

The **Delete** command allows one or more project databases to be deleted. The **Delete Project(s)** dialog (Figure 27) 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 associated radio button disabled. After selecting a project (or projects) to delete, selecting the **Delete** button removes the project database(s) from the server. 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.

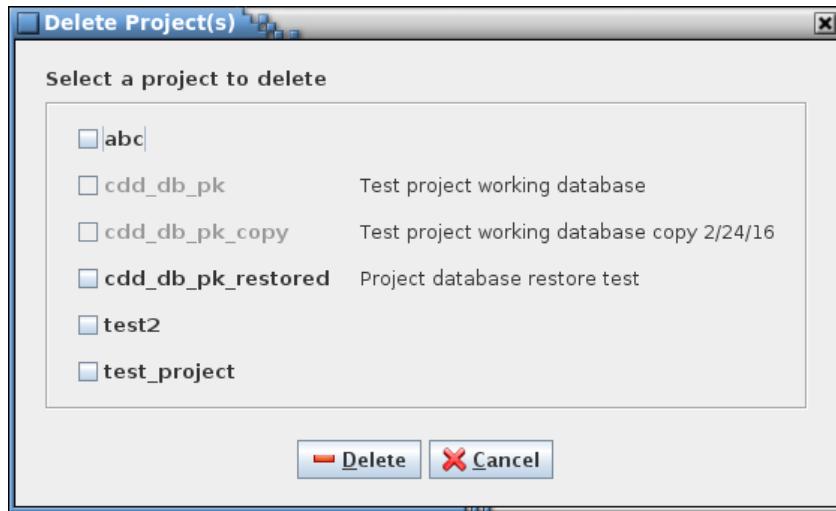


Figure 27. Delete Project dialog

#### 4.9.2.7 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 28). The backup file extension is '.dbu'. The **Append date and time to file name** check box, if selected, appends the current date (year, month, and day) and time (hours, minutes, and seconds) stamp to the file name. Deselecting the check box removes the data and time stamp. 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 4.9.2.8, 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.

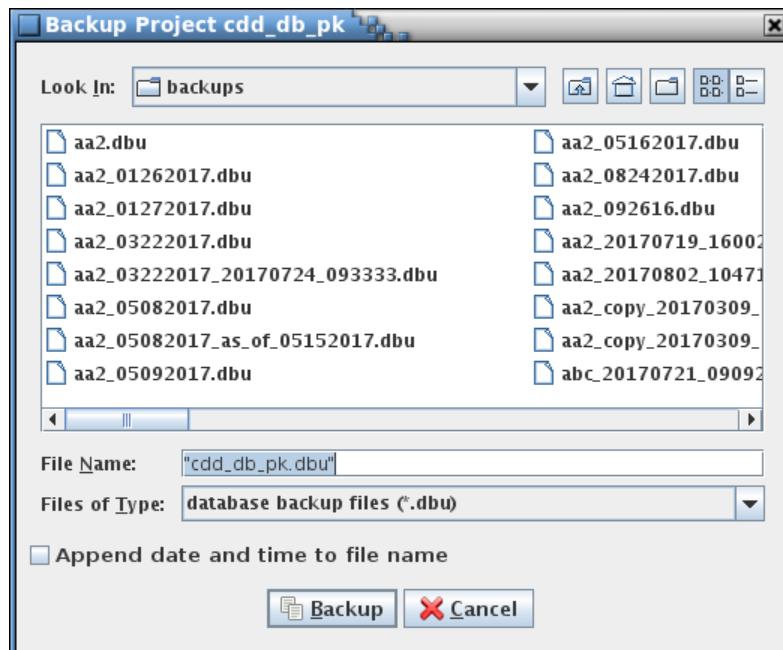


Figure 28. Backup Project dialog

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 58 of 295

#### 4.9.2.8 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 4.9.2.7). 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.

#### 4.9.2.9 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 29) appears when the command is issued. Though all projects are displayed, only those that are locked have their associated check box enabled.

The locked/unlocked status is displayed beside the project database name along with the name(s) of the user(s) 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 29, 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 *rmccclune*, 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 the multiple instances make updates to the update.*

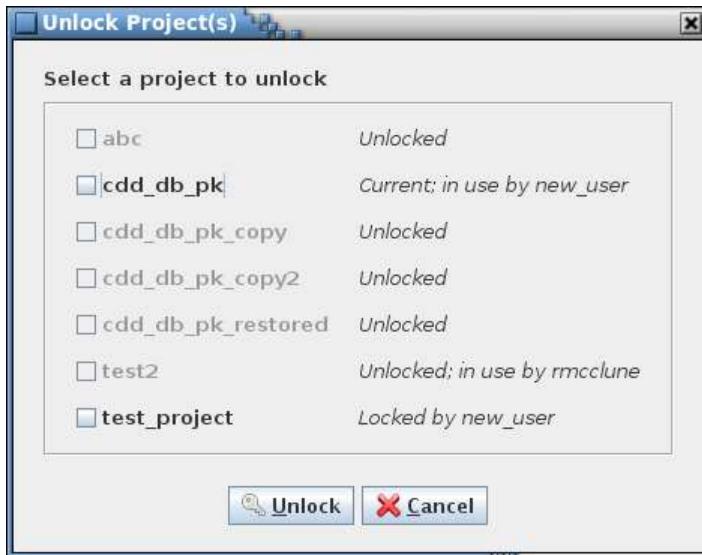


Figure 29. Unlock Project(s) dialog

#### 4.9.2.10 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; these tables can be created by the application's automatic update feature and can be ignored.

The *path reference check* verifies that the table and variable references in the internal tables are valid. Many of the internal tables store references to tables, and in the case of structure tables the table reference can include one or more child tables and associated variable names. When changes made to a table the references in the internal tables are updated as well. If any table or variable references no longer exist these references can be removed upon confirmation by the user.

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 Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 60 of 295

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 (these indices 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., an array member without a corresponding array definition) and missing 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 (an error dialog is displayed) and checked until the missing column is added 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 updated the subsequent missing definition warnings won't occur.

While the check is being conducted a dialog appears (Figure 30) showing the verification progress. This dialog allows halting the verification by pressing the **Halt** button. No changes are made to the project database if the **Halt** button is pressed.

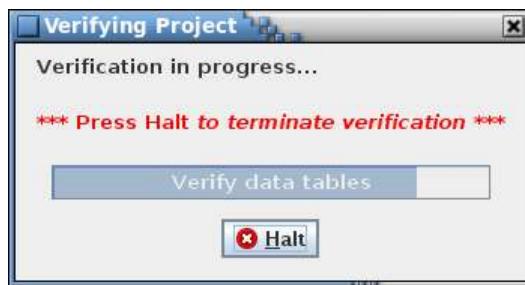


Figure 30. Verification and termination dialog

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 31). After selecting the check box(es) in the **Corrective 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 Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 61 of 295

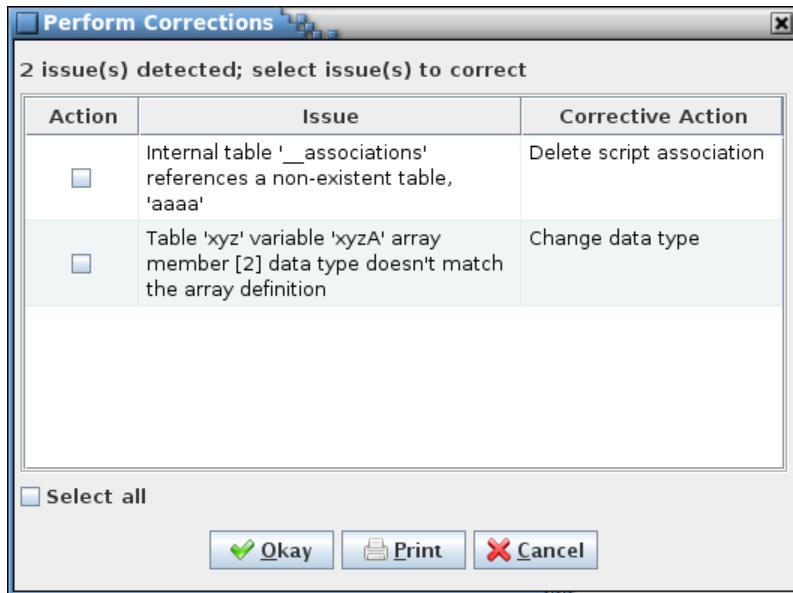


Figure 31. Example Perform Corrections dialog

#### 4.9.2.11 Recently opened project(s)

The remaining commands in the Project menu are the names of the most recently opened projects. Selecting one of these items opens the specified project. Note that the attempt to open the project is made in the currently attached PostgreSQL server.

### 4.9.3 Data menu

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

#### 4.9.3.1 New table(s)

The **New Table** command allows creation of a new data table. The **New Table** dialog (Figure 32) displays the defined table types and input fields for the table name and its description.

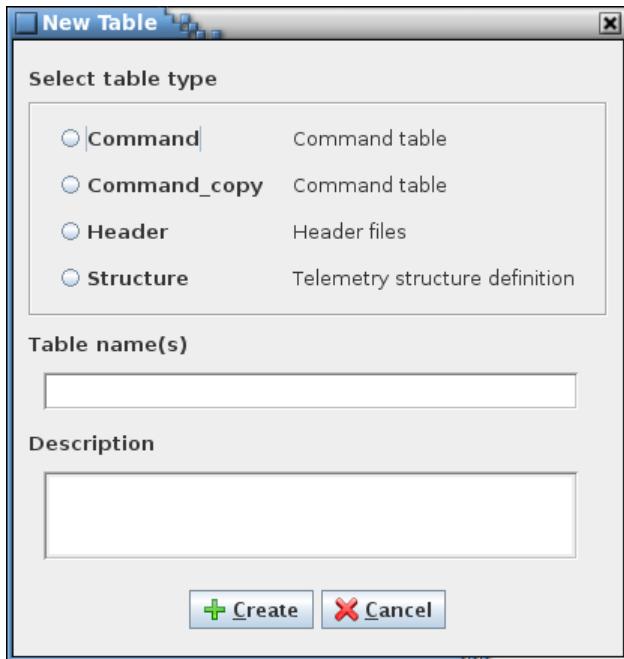


Figure 32. New Table dialog

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 begin with a pair of underscores (this is used to designate internal tables created by the CCDD application). A warning dialog appears if any constraint is violated.

Multiple tables of the same type may be created 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 4.9.3.2).

#### 4.9.3.2 Edit table(s)

The **Edit table(s)** command displays the data table selection dialog (Figure 33).

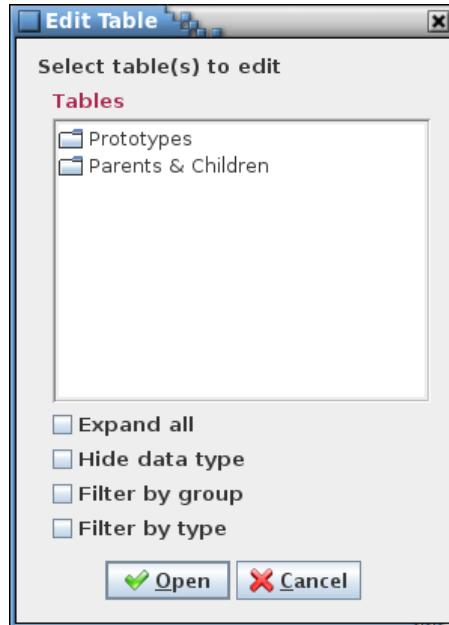


Figure 33. Edit Table dialog

The selection dialog has a table tree (see paragraph 4.5.3) 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 34 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.

A screenshot of the 'ahrs\_OutDataPacket\_T' table editor. The window has a menu bar with 'File', 'Edit', 'Row', 'Column', and 'Field'. A 'Table tab(s)' dropdown shows 'ahrs\_OutDataPacket\_T'. The main area is a grid table with columns: Variable Name, Description, Units, Data Type, Array Size, Bit Length, Enumeration, and Rate. The grid contains the following data:

Variable Name	Description	Units	Data Type	Array Size	Bit Length	Enumeration	Rate
CCSDS_HEADER	ccds header		CCSDS_T1sPkt_t				2
PKT_Time_Seconds	first portion of time stamp		int32				
PKT_Time_Subseconds	seconds portion of time stamp		int16				
ucCsdCounter			uint8				
ucErrCounter			uint8				
FdirFlags			AHRS_M_FDIR_T				
ucSpare1			uint8				
usExecCount			uint16				
padding			uint8	6			
padding[0]			uint8	6			
padding[1]			uint8	6			
padding[2]			uint8	6			
padding[3]			uint8	6			
padding[4]			uint8	6			
padding[5]			uint8	6			
ahrs_M_OutData			ahrs_M_OutData_T				

Below the table are sections for 'Description' (containing 'Attitude and heading reference system packet structure'), 'Message ID Name' (empty), 'Message ID' (0x1070), and a set of edit buttons: 'Ins Row', 'Up', 'Left', 'Delete', 'Store', 'Del Row', 'Down', 'Right', 'Redo', and 'Close'.

Figure 34. Example table editor

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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 64 of 295
---	---	----------------------------------

- Menu bar** The first section is the menu bar, which contains the commands, described in the following paragraphs, for manipulating the table contents.
- Table tab(s)** 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 root table shows only the prototype/root name. For a structure table's child table the tab displays a name in the format *root : structure.variable* where *root* is this table's root 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 pop-up tool tip showing the table's type, root table, and complete structure and variable path.
- Table data** The columns displayed in the tabbed pane's table are determined by the table type of the table being edited. The table columns can be sorted and repositioned as described in paragraph 4.2; however, the sorted order does not dictate the actual table data row order. See paragraph 4.9.3.2.4 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** This section contains the table description. The description is initially empty (unless set when the table was first created). The text entered here is used as a tool tip when the mouse pointer hovers over the table's 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 field(s)** This section displays any data field(s) assigned to the table. See paragraph 4.6 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 4.9.3.2.3.1.
  - Del Row** Deletes the selected row(s) from the table. See paragraph 4.9.3.2.3.2.
  - Up** Moves the selected row(s) up one row. See paragraph 4.9.3.2.3.3
  - Down** Moves the selected row(s) down one row. See paragraph 4.9.3.2.3.4.
  - Left** Moves the selected column(s) left one column. See paragraph 4.9.3.2.4.1
  - Right** Moves the selected column(s) right one column. See paragraph 4.9.3.2.4.2.
  - Undo** Undoes the last action performed (typing, paste, insert, delete, redo, etc.).
  - 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 4.9.3.2.1.3.
  - Close** Closes the currently displayed table's editor. See paragraph 4.9.3.2.1.8.

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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 65 of 295

#### 4.9.3.2.1 File menu

##### 4.9.3.2.1.1 *Edit table(s)*

The **Edit table(s)** command displays the Edit Table dialog (Figure 33). The table(s) opened from this dialog appear in the current table editor under their own tabs.

##### 4.9.3.2.1.2 *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 current table editor under its own tab.

##### 4.9.3.2.1.3 *Store current*

The **Store current** command stores the currently displayed table's contents, including the table's cell data, description, data fields, row order, and 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 and canceling it.

##### 4.9.3.2.1.4 *Store all*

Selecting the **Store all** command is similar to the **Store current** command (paragraph 4.9.3.2.1.3) 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 and canceling it.

##### 4.9.3.2.1.5 *Import data*

The **Import data** command provides a means of inserting data and/or data fields from a CSV, EDS XML, JSON, or XTCE XML formatted file into the table currently displayed in the table editor. The import file may contain information for multiple tables, but only the data (and fields for CSV and JSON imports) for the first table defined in the file are imported. A dialog appears (Figure 35) allowing the user to choose the import file based on file format. The **Export table** and **Export table(s)** commands produce files compatible with the import command; see paragraphs 4.9.3.2.1.6 and 4.9.3.8; Appendix C contains details on the file formats.

The **Overwrite existing cells** check box, if selected, causes the imported data to overwrite data in the existing cells, beginning at the currently selected row. If unchecked, 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.

The **Use existing field if duplicate** check box, if selected (the default), causes any imported data fields to be ignored if the table already has a field of the same name. If unchecked, then the existing data field is replaced by the imported one.

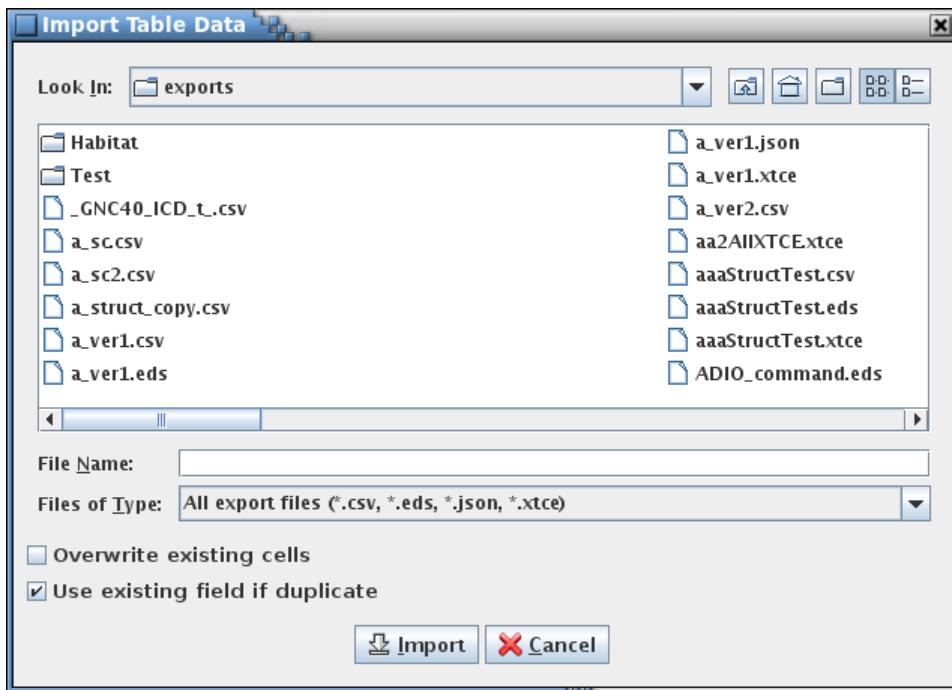


Figure 35. Import data and fields into an existing table dialog

For CSV and JSON imports the data type and macro definitions in the import file, if present, are ignored. The table type definitions are evaluated in the event the table from which the data is imported is otherwise undefined. If a column name is missing in the table type definition a dialog appears alerting the user to the error. The user can elect to ignore this table type definition and continue with the import, ignore any remaining table type errors in this import file and continue, or cancel the import. Following the import any table type additions are removed.

For CSV and JSON imports the column names in the import file associated with the row data are compared to those in the table's table type definition columns in order to insert the cell values into the proper column. If the column name associated with a cell value isn't recognized as one belonging to the table's table type definition then that cell value is ignored and a dialog appears alerting the user to the discrepancy. The user can elect to ignore the unrecognized column name and continue with the import, ignore any remaining unrecognized column names in this import file and continue, or cancel the import. For EDS AND XTCE imports the data is placed in the structure or command table based on the table's table type definition column input types (for example, a variable name is placed in the column with the **Variable name** input type, regardless of its position in the table).

#### 4.9.3.2.1.6 Export table

The **Export table** command provides a means of outputting the current table's definition to a file in CSV, EDS, JSON, or XTCE format. This is equivalent to the **Export table(s)** command in the main window's **Data** menu - see paragraph 4.9.3.8 for details. The dialog that appears when is similar to that for the **Export table(s)** command, except there is no table tree.

#### 4.9.3.2.1.7 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.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 67 of 295

#### 4.9.3.2.1.8 Search

Selecting the **Search** command causes a search dialog to be displayed (see Figure 36). This dialog allows searching for matching text in the currently active table editor. The search dialog remains on top of the table editor dialog. If the editor dialog has multiple tabs, then when another tab is selected the search criteria are applied to the new active table editor. If a window other than that for the table editor dialog or search dialog is selected the search dialog is hidden; the dialog becomes visible again when the table editor dialog is selected. Pressing the Ctrl-F key sequence while the table editor dialog window has the focus also displays the search dialog.

Case sensitivity for the search is governed by the **Ignore text case** check box.

The **Allow regular expression** check box, when checked, allows the use of a regular expression to define the search pattern in the search text field. A regular expression can be formulated to find multiple matching conditions (for example, the search for **a.c** would match any string that has a single character between the characters 'a' and 'c'). Information regarding the use of regular expressions is beyond the scope of this document; however, resources and tutorials can be found online. When unchecked, the search text is matched as typed in the search text field.

Enter the search text in the input field and select the **Search** button. Any matching text in the table is highlighted and the total number of matches, including multiple matches in a single cell, is displayed beside the search text field's label. If the search text field is empty and the **Search** button is pressed, or if the search dialog is closed, then any search match highlighting is removed.

The search text field uses auto-completion to fill in the search string. The previous search strings (those for the event log, table, and script) are remembered, including those from previous sessions. The number of remembered search strings can be changed via the Preferences dialog, and defaults to 30. Case sensitivity for auto-completion is based on the **Ignore text case** check box selection state.

If the table contains one or more instances of the search text then the search dialog's **Previous** and **Next** buttons are enabled. These buttons cause the table cell selection to move from the currently selected cell to the previous or next cell containing a search match, wrapping to the end or beginning of the table if needed. The table scrolls, and for a structure table array members are expanded, if necessary for the selected cell's row to be visible.

The input text can be changed and the **Search** button pressed again to initiate another search of the table. To exit the search dialog select the **Close** button.

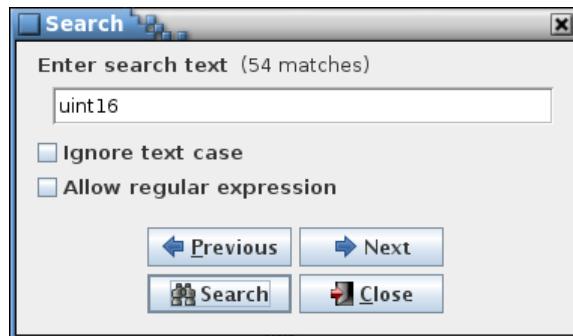


Figure 36. Data table search dialog

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 68 of 295

#### 4.9.3.2.1.9 *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.

#### 4.9.3.2.1.10 *Close all*

The **Close all** command performs a similar operation to the **Close current** command (paragraph 4.9.3.2.1.8) 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.

### 4.9.3.2.2 **Edit menu**

#### 4.9.3.2.2.1 *Copy*

The **Copy** command places the contents of the highlighted cell(s) into the operating system's clipboard. This information can then be pasted into another table cell or input field within the application, or into applications other than CCDD. The Ctrl-C keys perform the same operation.

#### 4.9.3.2.2.2 *Paste*

The **Paste** command places the contents of the operating system's clipboard into one or more table cells, if a cell is selected, or into an input field if a field is selected. The paste location within the table 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 then placed into the editor beginning at this location and extending down and to the right, overwriting the existing data in the cells. If insufficient columns exist for the pasted data then the excess column data is ignored. 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 4.9.3.2.2.3 on inserting copied data without overwriting the existing cell contents. If data from multiple cells is pasted into an input field then content from each cell is concatenated, separated by a space, and the result is pasted into the field. The Ctrl-V keys perform the same operation.

#### 4.9.3.2.2.3 *Insert*

The **Insert** command behaves similarly to the **Paste** command (paragraph 4.9.3.2.2.2) 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 perform the same operation.

#### 4.9.3.2.2.4 *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 table editor. The Ctrl-Z keys perform the same operation.

#### 4.9.3.2.2.5 *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 table editor. The Ctrl-Y keys perform the same operation.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494		Version 1.2.31
	Date: May 2018		

#### 4.9.3.2.2.6 Insert macro

The **Insert macro** command provides a convenient means to insert a macro into a table cell. Editing must first be initiated 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, then select the **Insert Macro** command. A pop-up list appears displaying all macros with values that are consistent with the input type of the cell being edited (see Figure 37; see paragraph 4.7 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 37). Press the Escape key to remove the macro pop up dialog without inserting a macro.

The Ctrl-M keys can be used in place of the **Insert macro** command. While editing a cell, position the text cursor or highlight one or more characters to be replaced, then press Ctrl-M. The pop-up list appears as described above.

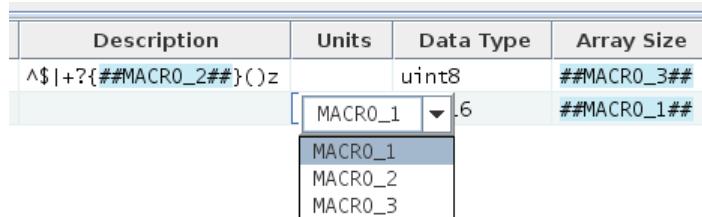


Figure 37. Example of macro name display and pop-up dialog in a data table

#### 4.9.3.2.2.7 Show macros

The **Show macros** command, when checked, temporarily replaces every macro with its corresponding value in each table of the editor dialog. No editing of the table may take place while the command is selected (all tables cells are grayed out and most of the editor menu commands and buttons are disabled). Deselecting the **Show macros** command restores the macros and enables normal editing and commands.

Pressing Ctrl-Shift-M produces a similar effect, causing the currently selected table to display all cells containing a macro to replace the macro with its value; releasing the Ctrl-Shift-M keys restores the macro names in the cells.

#### 4.9.3.2.2.8 Clear selected

The **Clear selected** sub-menu contains commands for replacing the contents of the selected cell(s) in the currently displayed table.

##### 4.9.3.2.2.8.1 With blanks

The **With blanks** command replaces the values in the selected cell(s) with a blank. Pressing the Delete key performs the same action.

##### 4.9.3.2.2.8.2 With prototype

The **With prototype** command is only available if the table editor is displaying a child table. This command replaces the value in selected cell(s) with the contents of the corresponding cell in the table's prototype table. Pressing the Shift-Delete keys performs the same action. A special indicator flag ( $\emptyset$ ) is prepended to the cell value and highlighted (see Figure 38 for an example). If the table changes are stored or the cell is subsequently edited this indicator is removed.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494		Version 1.2.31
	Date: May 2018		

n	Rate	Limit Sets	Po
5	test		

Figure 38. Special indicator flag example

#### 4.9.3.2.3 Row menu

##### 4.9.3.2.3.1 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 cells in multiple rows are selected then the new row is inserted below the lowest one. If no cell is selected then the new row is inserted at the end of the table. The Insert key performs the same operation.

A row may not be inserted within an array. If an array definition or member cell is selected then the new row is inserted below the last array member.

##### 4.9.3.2.3.2 Delete row(s)

The **Delete row(s)** command performs the same action as the **Del Row** button. This command deletes the row associated with each currently selected cell. If cells in multiple rows are selected then each of the rows is deleted. If no row is selected then this command has no effect. The Delete key performs the same operation.

A row may not be removed from within an array. If an array definition or member cell is selected then the entire array – definition and members – is deleted.

##### 4.9.3.2.3.3 Move up

The **Move 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 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).

Arrays cannot be split by use of the **Move up** command. If an array definition or member cell is selected then the array moves up as a unit.

##### 4.9.3.2.3.4 Move down

The **Move 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 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).

Arrays cannot be split by use of the **Move down** command. If an array definition or member cell is selected then the array moves down as a unit.

##### 4.9.3.2.3.5 Expand arrays

The **Expand arrays** command is only available for those tables containing a column with an input type of "Array index" (e.g., the **Array Size** column in the default **Structure** table type). This command toggles display of array members in the table. 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 header) and double right-clicking.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 71 of 295

#### 4.9.3.2.3.6 *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.

#### 4.9.3.2.4 **Column menu**

##### 4.9.3.2.4.1 *Move left*

Issuing the **Move left** command moves the 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).

##### 4.9.3.2.4.2 *Move right*

Issuing the **Move right** command moves the column(s) of the selected cell(s) to the right 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).

##### 4.9.3.2.4.3 *Reset order*

The **Reset order** command restores the column order for the currently displayed table to the default order. The default order is established by the order of the column definitions in the table type editor for the table's type.

#### 4.9.3.2.5 **Field menu**

##### 4.9.3.2.5.1 *Manage fields*

The **Manage fields** command allows the user to create, alter, and delete data fields for the table represented by the active table editor tab. See paragraph 4.6 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 table.

##### 4.9.3.2.5.2 *Clear values*

The **Clear values** command clears the contents of all of the currently displayed table'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 affecting the data field values.

#### 4.9.3.3 **Recent**

This sub-menu contains the names (with paths) of the most recently opened tables. Selecting one of these items opens the specified table. Note that the attempt to open the table is made in the currently open project.

#### 4.9.3.4 **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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 72 of 295

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 39) appears, displaying a table tree showing the prototype tables and input fields for providing the selected table's new name and description. See paragraph 4.5.3 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 4.9.3.1 for details on table name constraints. The new name and description are 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 a change to a table name.

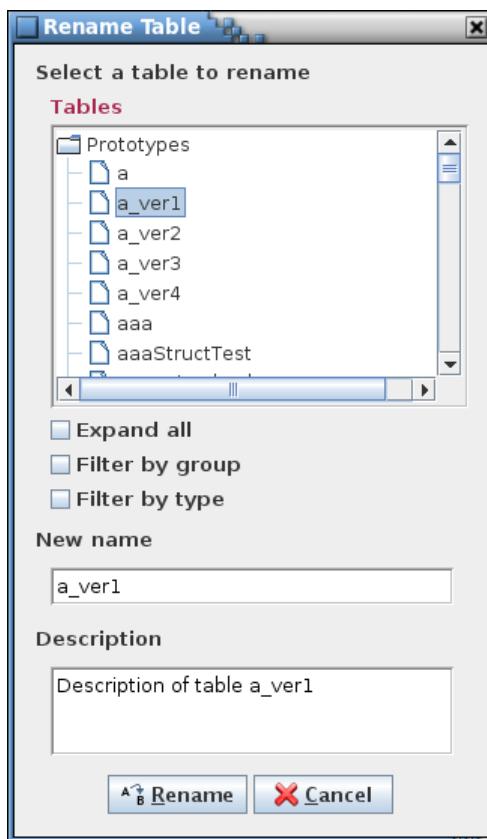


Figure 39. Rename Table dialog

#### 4.9.3.5 Copy table

The **Copy table** command allows a prototype data table to be copied. The **Copy Table** dialog (Figure 40) appears, displaying a table tree showing the prototype tables and input fields for providing the name and description of the selected table's copy. See paragraph 4.5.3 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 4.9.3.1 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 copying a table.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 73 of 295
---	---	----------------------------------

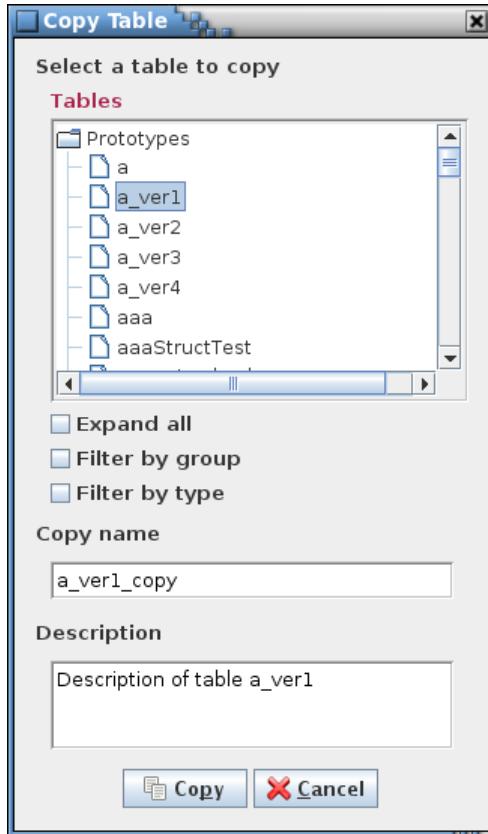


Figure 40. Copy Table dialog

#### 4.9.3.6 Delete table(s)

The **Delete table** command allows one or more prototype tables to be deleted. The **Delete Table** dialog (Figure 41) appears, displaying a table tree showing the prototype tables. See paragraph 4.5.3 for details on the table tree. After one or more tables is selected from the tree the **Delete** button is selected to delete the table(s). A confirmation dialog appears and if **Okay** is selected all instances of the deleted table, both parent and child tables, are deleted from the project database, including those appearing in open table editors. Select the **Cancel** button to exit the dialog without deleting a table.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 74 of 295

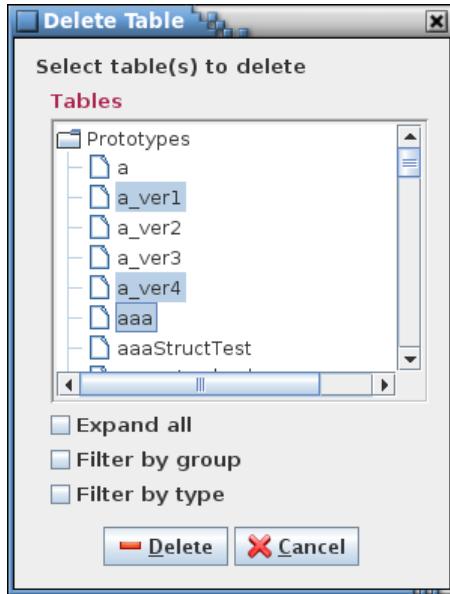


Figure 41. Delete Table dialog

#### 4.9.3.7 Import table(s)

The **Import table(s)** command allows importing information from CSV, EDS, JSON, and XTCE formatted files in order to create tables in the project database. CSV and JSON import files can contain information for tables of any type (structure, command, and other user-defined types), table type definition(s), data type definition(s), macro definition(s), reserved message IDs, and variable paths. EDS and XTCE XML import files are restricted in the information contained, so only basic information for structure and command tables can be imported from files in these formats. When the command is selected a dialog appears allowing the user to choose the location of the import file(s) (see Figure 42).

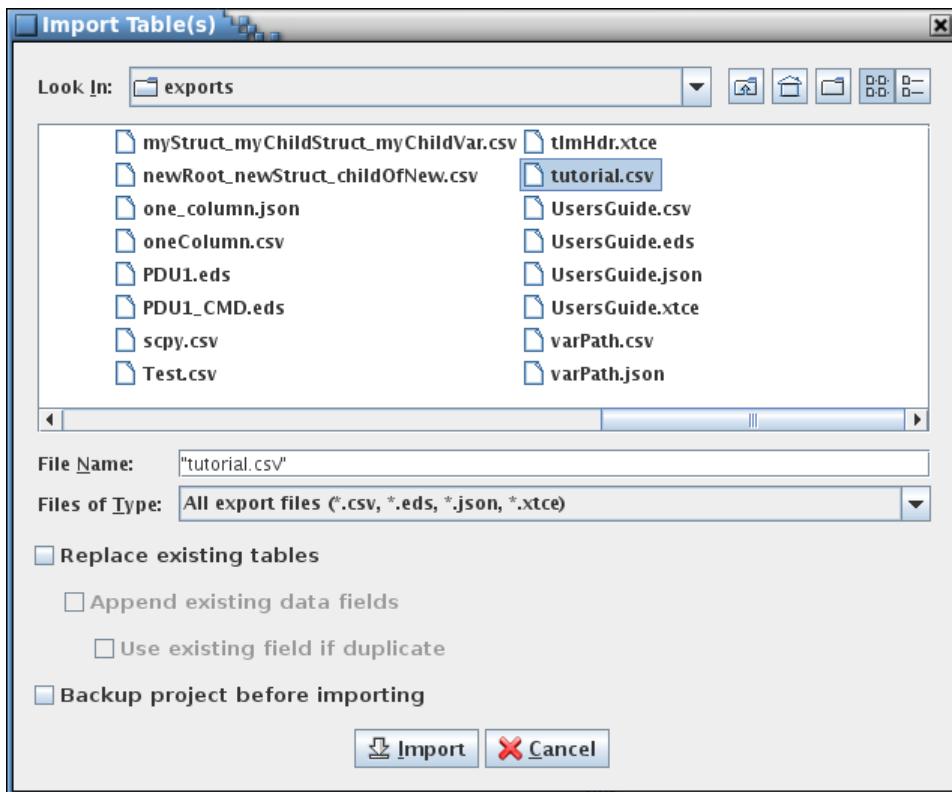


Figure 42. Import table(s) dialog

The **Replace existing tables** check box allows the user to choose whether to replace any existing tables with the same name with the import file data. If selected, the **Append existing data fields** check box is enabled, which indicates if replaced tables that have data fields have the existing fields appended to those imported. If this check box is selected then the **Use existing field if duplicate** check box is enabled, which 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, **Backup project before importing**, determines if a backup of the project database is created prior to continuing with the import operation.

More information regarding the permissible formats for the import file are described in Appendix C.

When a structure table is imported, if it has a variable with a data type that is a reference to another structure and the prototype for this child structure doesn't already exist in the project and isn't defined in the import file(s), then the prototype table is created. This auto-generated prototype is a copy of the child structure; i.e., it's created using the same structure table type definition as the child structure and contains all of the information in the child.

#### 4.9.3.7.1 Importing table types

When importing from EDS or XTCE XML formatted files a single structure and/or a single command table type is created on which all of the structure and command tables are based. If more than one command table definition is present in the import file then the one with the largest number of arguments determines the command table type definition (these argument entries are empty for command tables requiring fewer arguments). The table type definition names are 'Structure' and 'Command' with with 'EDS\_' or 'XTCE\_' prepended and are guaranteed to be unique (a numeral is appended to the table type definition name as needed to differentiate the new type from an existing one).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 76 of 295
---	---	----------------------------------

CVS and JSON formatted files must either use a table type definition already defined in the target project database, or contain a definition for each type of table imported. If a table type present in the import file has the same name as one already present in the project then the column definitions must match (other than for the column descriptions), otherwise an error dialog appears; if the error is ignored then this table type definition is skipped. Note that any tables in the import file using this type won't successfully be imported. Three options are available to overcome the table type collision: (1) change the existing type's name, (2) change the type's name in the import file (including any references in any table definitions using it), or (3) alter the existing table type to match the imported type (adding, removing, or altering the column definitions as needed). Once the changes are made the import can be performed again.

When importing a table type column definition the inputs are expected in the same order as they appear in the table type editor (see Figure 49). If the number of inputs is fewer or more than those expected, or the column name is blank, then an error dialog appears; if the error is ignored then this table type definition is skipped. If the input type is blank then it defaults to 'Text', but if the input type is present and isn't one recognized by the application (see paragraph 4.7) a warning error dialog appears; if the error is ignored then the input type 'Text' is used. The inputs representing the check box columns evaluate to 'true' if the input is 'true' (case insensitive); otherwise 'false' is used, regardless of the actual value supplied.

#### 4.9.3.7.2 Importing data fields

When importing from EDS and XTCE XML formatted files certain data fields are created if the information is present in the import file. These fields are the application (or message) ID for root structure tables and command tables, and the system path for each table (root or instance).

When importing a data field definition from a CSV or JSON formatted file the inputs are expected in the same order as they appear in the data field editor (see Figure 7). This includes an input for the "Applicability Type", even if the data field isn't associated with a structure table type definition. An additional input containing the field's value is expected as the last input. If the number of inputs is fewer or more than those expected, or the field name is blank, then an error dialog appears; if the error is ignored then this data field is skipped. If the input type is blank then it defaults to 'Text', but if the input type is present and isn't one of the recognized ones a warning dialog appears; if the error is ignored then the input type 'Text' is used. The input representing the check box column evaluates to 'true' if the input is 'true' (case insensitive); otherwise 'false' is used, regardless of the actual value supplied. If the applicability type is blank then it defaults to 'All tables', but if the applicability type is present and isn't one of the recognized ones a warning dialog appears; if the error is ignored then the applicability type 'All tables' is used.

#### 4.9.3.8 Export table(s)

The **Export table(s)** commands allow outputting a project's data table information in one of four formats: CSV, JSON, EDS XML, and XTCE XML. The CSV and JSON formats include information for the table type definition(s), data type definition(s), macro definition(s), reserved message IDs, variable paths, and table definition(s). This allows these formats to exchange detailed project information with another CCDD project. The XML formats only export a subset of the information that appears in the project's structure and command data tables.

Each export dialog, shown in the following sections, differs in options and inputs. The common features are as follows. At the top of the dialog is a table tree for selecting the tables to export; below the tree are the various tree filters – see paragraph 4.5.3 for more details. The dialog includes an input field for specifying the export path or file. The CSV and JSON formats allow for storing the output either combined into a single file or with each table in its own file based on the selection of the radio buttons (**Store in Multiple files** or **Single file**). The EDS and XTCE output is to a single file only. The output file

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 77 of 295

name or path must be entered in the export file/path field, or selected from the file chooser dialog that appears when the **Select...** button is pressed. For multiple tables with output to individual files the file names are automatically assigned based on the table names. If an output file already exists then the table isn't exported unless the **Overwrite existing file(s)** check box is selected or the user confirms overwriting each file (via a dialog that appears during the export operation).

The dialogs for the EDS and XTCE exports contain a pair of radio buttons, **Endianess Big** and **Little**, that are used to determine the flags set in the export file that indicate if the data is in big or little endian format. If little endianness is selected then the check box **Headers are big endian** is enabled. If selected, the telemetry and command header table variables are flagged as big endian (as an example, the CFS CCSDS headers are always big endian). This check box is selected by default.

The following inputs allow for modifying what other information is included in the export file and only apply to CSV and JSON exports. When the **Substitute macro values for macro names** check box is selected the table cells containing macro references have the references replaced with the corresponding macro value. The macro definitions are not stored in the export file(s) in this case. The **Include reserved message IDs** check box, when checked, causes the message IDs in the reserved message ID editor (see paragraph 4.9.3.13.2) to be exported. If the **Include project data fields** check box is selected then all of the project-level data field definitions are included in the output. If the **Include variable paths** check box is selected then the variable path, in both application and user-defined formats, for each variable is included in the output. The remaining inputs become active when this check box is selected. These inputs behave identically to those described in paragraph 4.9.3.17.

The column data in an exported table file may be imported into an existing table using the **Import data** command (paragraph 4.9.3.2.1.5). The **Import table(s)** command (paragraph 4.9.3.7) imports the entire contents of the export file into the current project, creating tables, table type definitions, and data field definitions (and data type definitions, macros, reserved message IDs, and variable paths in the case of CSV and JSON formatted files) as described in the file. Details on the allowable formats of the export file are described in Appendix C.

#### 4.9.3.8.1 CSV

The **Export table(s) - CSV** command allows exporting one or more selected tables in CSV format. See Appendix C.1 for details on the file format.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Date: May 2018

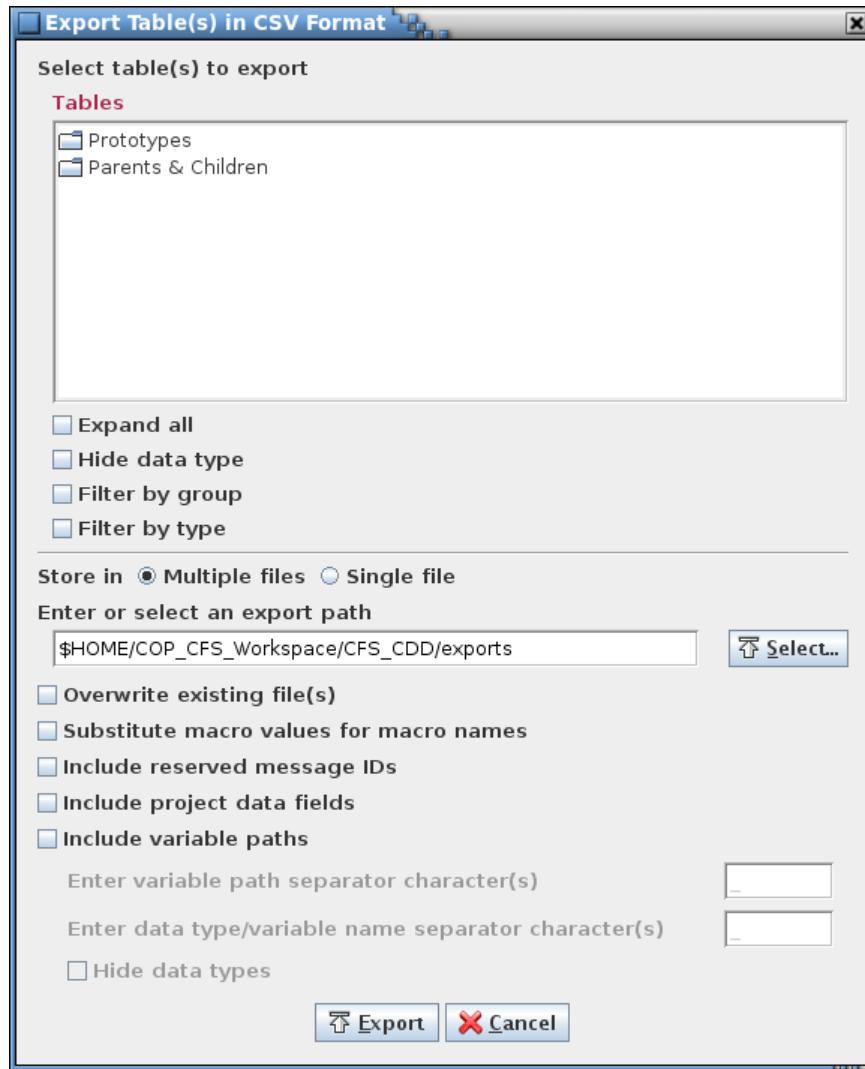


Figure 43. CSV export dialog

#### 4.9.3.8.2 EDS

The **Export table(s) - EDS** command allows outputting one or more selected tables in EDS XML format. Only certain data for structure and command tables is exported; other table types are ignored. Any macros are expanded in the exported data. See Appendix C.2 for details on the file format.

Four special project-level data fields are utilized by the EDS format (see paragraph 4.9.3.14 for information on assigning project-level fields). These fields are described in Table 7. The names of the fields can be whatever the user desires; the input types determine how the fields are used during the export process. The telemetry and command header table names indicate which tables represent the telemetry and command header structures, respectively. All root structure tables reference a single telemetry header in the export file, and all command tables reference a single command header in the export file. If a field is missing or its contents is empty the export is still performed, but certain data may be missing. For example, without the header fields or the application ID and function code fields the export operation can't determine where to place the ID and function code values in the export file, so these values are ignored. The contents of the four data fields are stored as device metadata in the export file. Application ID and command function code use default values if not specified in the data fields.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Date: May 2018

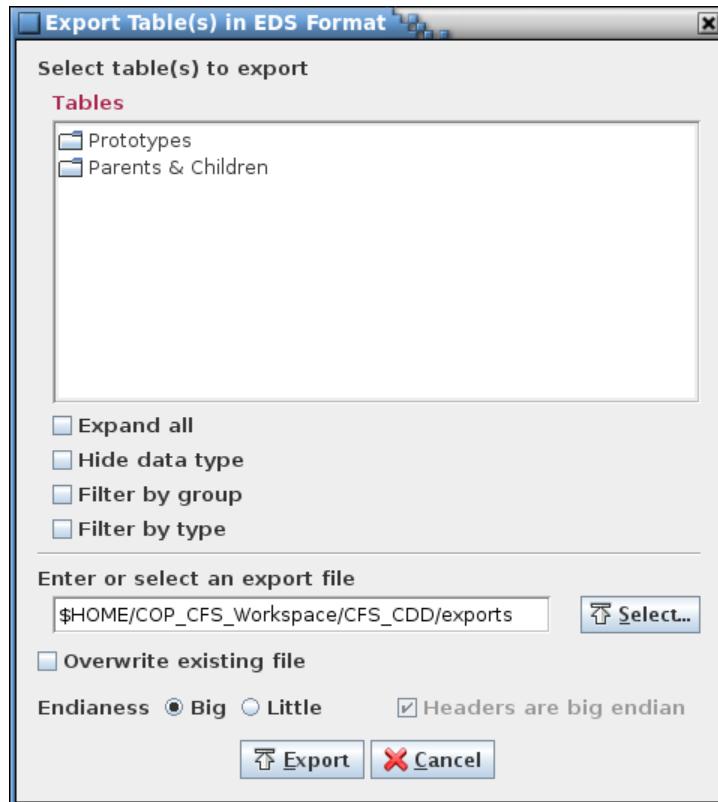


Figure 44. EDS export dialog

Input Type	Description
XML: Application ID	Name of the variable containing the application ID in the structure tables representing the telemetry and command headers
XML: Command Header	Name of the structure table representing the command header
XML: Function Code	Name of the variable containing the command function code in the structure table representing the command header
XML: Telemetry Header	Name of the structure table representing the telemetry header

Table 7. XML special data fields

#### 4.9.3.8.3 JSON

The **Export table(s) - JSON** command allows outputting one or more selected tables in JSON format. See Appendix C.3 for details on the file format.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 80 of 295

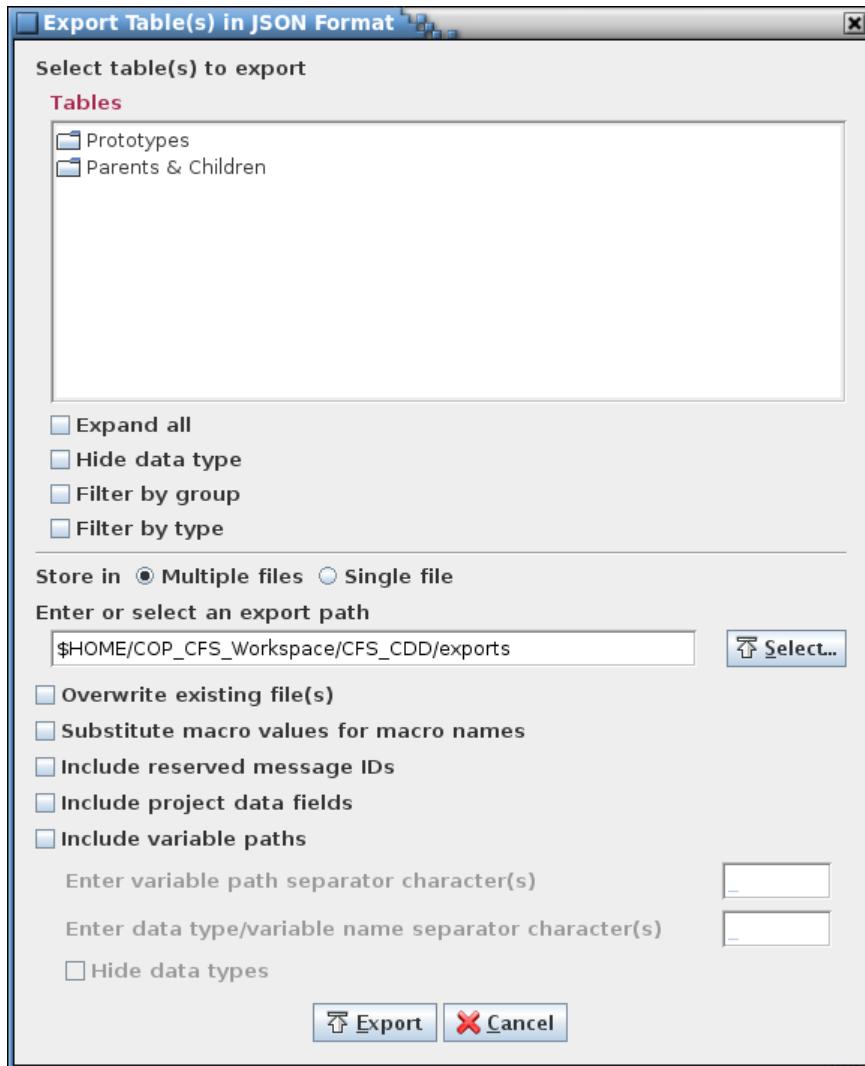


Figure 45. JSON export dialog

#### 4.9.3.8.4 XTCE

The **Export table(s) - XTCE** command allows outputting one or more selected tables in XTCE XML format. Only certain data for structure and command tables is exported; other table types are ignored. Any macros are expanded in the exported data.

The **Use external methods** check box, when checked, indicates that the script, named in the **Enter script file name** input field, is used to perform some or all of the XTCE conversion. The process is identical to performing the export operation via script association (see paragraph 4.10.7 for details on modifying the XTCE conversion via a script). However, not all of the script data access methods are available when using the external methods feature from the export dialog – any method referencing the table names, rows, table data, etc. is not accessible.

The XTCE dialog has additional inputs for defining certain XTCE attributes and classification levels that are used when constructing the output file. Default values are provided, but can be changed as desired. The attributes are used to construct the XML **Header** elements. See Appendix C.4 for details on the file format.

Four special project-level data fields are utilized by the XTCE format (see paragraph 4.9.3.14 for information on assigning project-level fields). These fields are described in Table 7. The names of the

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 81 of 295

fields can be whatever the user desires; the input types determine how the fields are used during the export process. The telemetry and command header table names indicate which tables represent the telemetry and command header structures, respectively. All root structure tables reference a single telemetry header in the export file, and all command tables reference a single command header in the export file. The command tables all reference a single command header in the export file. If a field is missing or its contents is empty the export is still performed, but certain data may be missing. For example, without the header fields or the application ID and function code fields the export operation can't determine where to place the ID and function code values in the export file, so these values are ignored. The contents of the four data fields are stored as ancillary data in the root system in the export file. Application ID and command function code use default values if not specified in the data fields.

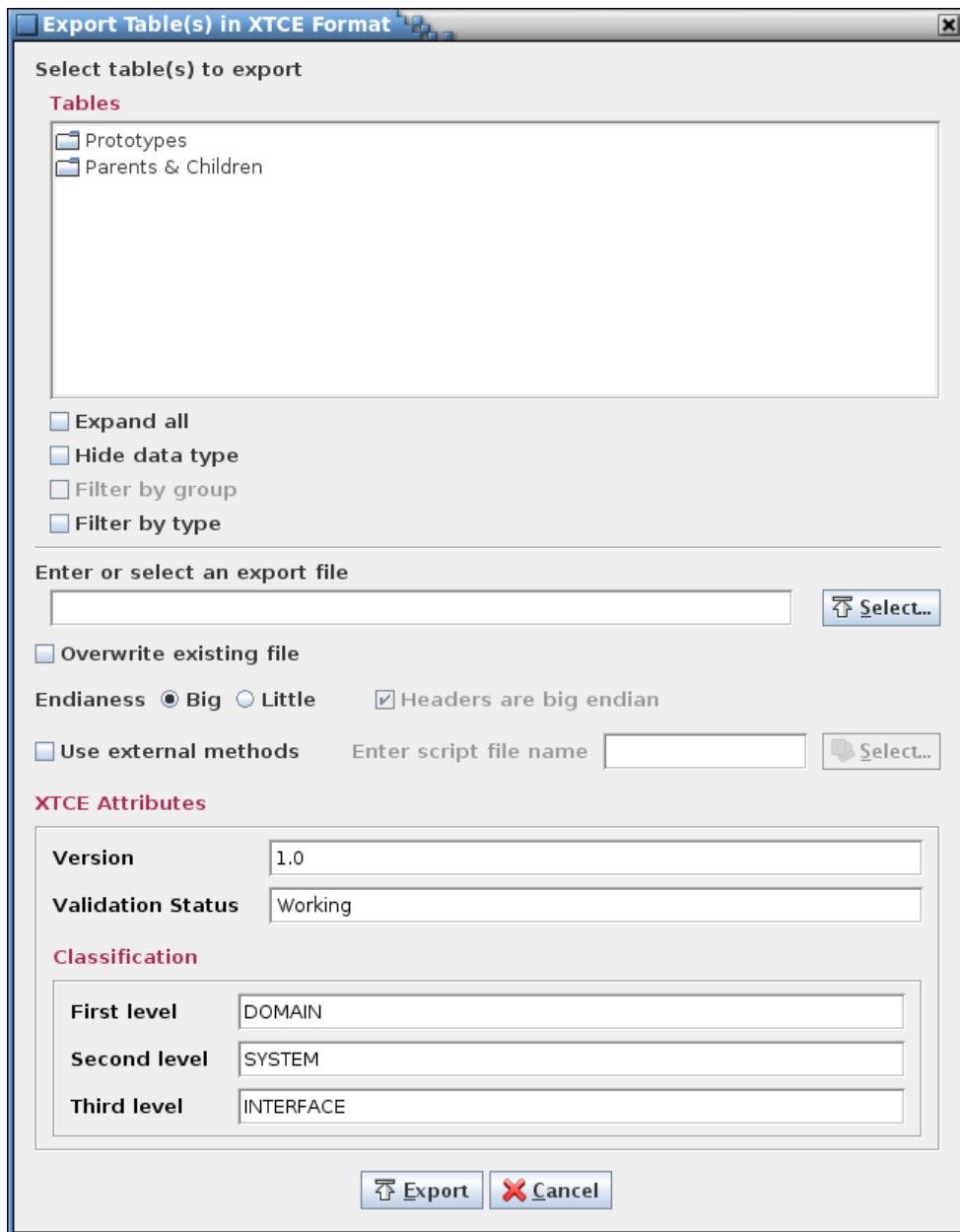


Figure 46. XTCE export dialog

#### 4.9.3.9 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 47 appears.

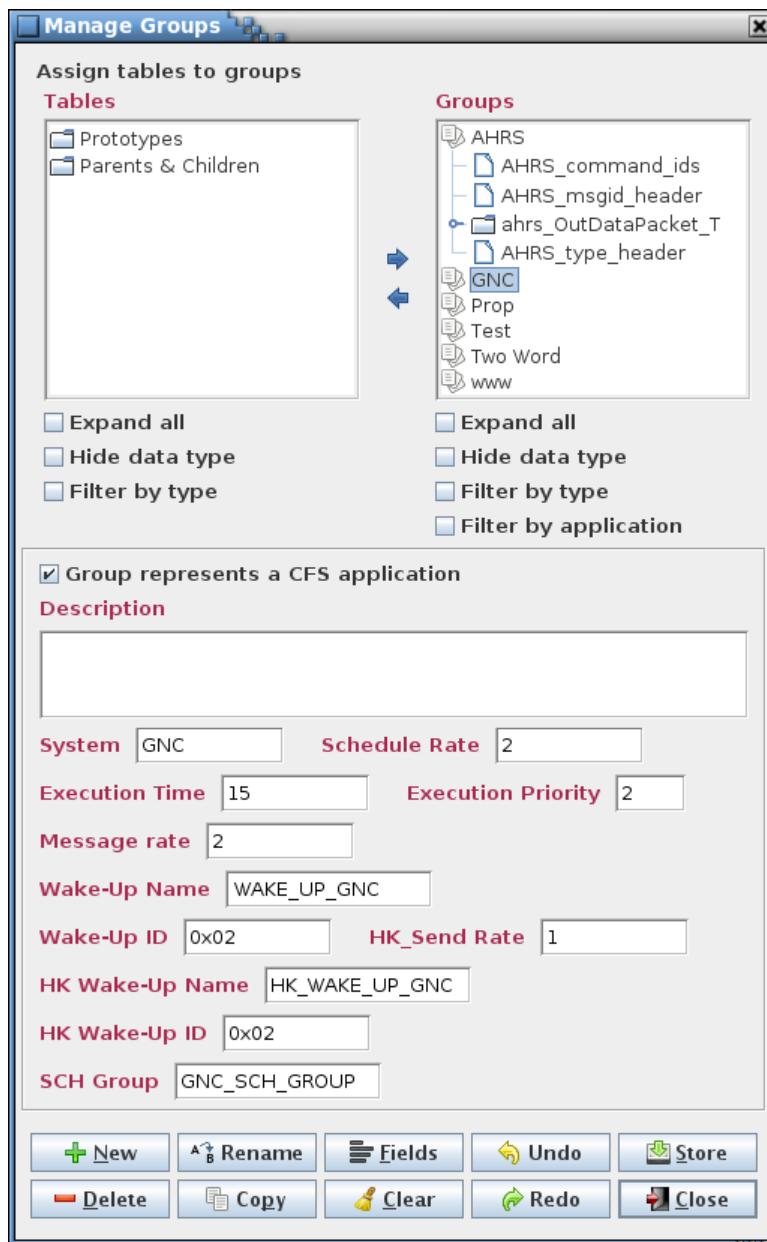


Figure 47. Manage Groups dialog

The upper left of the dialog contains a table tree (under the **Tables** heading). The upper right shows the groups and their trees (under the **Groups** heading). Both trees have the **Expand all** and **Filter by type** check boxes (see paragraph 4.5.3 for further details). The **Groups** tree has an additional filter, **Filter by application**, which causes the groups to be divided into two sub-trees, **Application** and **Other**. The **Application** sub-tree displays only those groups that represent a CFS application (see below for more details), and the **Other** sub-tree shows the remaining groups. In between the tree panes are arrows for moving tables in and out of the group(s). Each tree also has one or more check boxes, to

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 83 of 295
---	---	----------------------------------

expand/collapse the tree and to filter the tree information. 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 in the table trees whenever the mouse pointer hovers over a group name.

The space separating the table and group trees delineates a split pane control that is used to resize these panels relative to one another. Position the mouse pointer between the two tree panels and when the pointer changes to a double-headed arrow press and hold the left mouse button. Space permitting, the adjoining panes can be resized by moving the mouse pointer left or right. Release the mouse button to exit resizing.

If a variable is selected in the **Tables** tree then every group to which it belongs is selected automatically in the **Groups** tree. If multiple tables are selected in the **Tables** tree then any selected groups are deselected. Selecting a non-grouped table deselects any highlighted group in the **Groups** tree.

To create a group select the **New** button and provide a group name and description in the input dialog that appears (Figure 48). The group name may not be blank, nor is the name allowed to match that of an existing group (including the automatically supplied group's name, 'All tables', which does not appear in the **Groups** tree). The group name may contain alphanumeric, spaces, and punctuation characters; there is no length constraint. If the **Group represents a CFS application** check box is selected then the group is automatically assigned a number of data fields appropriate for a CFS 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 populating the scheduler table created with the application scheduler (paragraph 4.9.4.3). If **Okay** is selected the new group's name appears in the group tree.



Figure 48. 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 structure 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 root table, 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.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 84 of 295

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 **Groups** 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 **Groups** 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 copy has all of the tables assigned to it that are assigned to the original. 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 4.6) 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 (see paragraph 4.6.1 for details on its use) for the currently selected group. Once a field is created it can have a value assigned, and the values for existing fields may be altered if desired. The data field values for the currently selected group can be cleared by selecting the **Clear** button.

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.

The group manager button commands are summarized below:

- |               |   |
|---------------|---|
| <b>New</b>    | Create a new group.   |
| <b>Delete</b> | Delete the selected group(s).   |
| <b>Rename</b> | Rename the selected group.  |
| <b>Copy</b>   | Create a copy of the selected group, including its member tables and data fields.                       |
| <b>Fields</b> | Invokes the data field editor in order to create, alter, and delete data fields for the selected group. |

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Version 1.2.31

Date: May 2018

Page 85 of 295

- Clear** Replace the values in all data fields with blanks and deselect any check box data field for the selected group.
- Undo** Undoes the last action performed (table assignment, typing, paste, insert, delete, redo, etc.) on the selected group.
- Redo** Reverses the last action undone (table assignment, typing, paste, insert, delete, undo, etc.) on the selected group.
- Store** Stores the changes made to the groups in the group manager into the project database.
- Close** Closes the group manager 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 dialog.

#### 4.9.3.10 Manage table types

The **Manage table types** command opens the table type editor (Figure 49). The editor window can be broken down into the following sections: the menu bar, table type tab(s), column definitions, type description, and buttons. An additional section, showing data fields, is visible below the description if data fields are created.

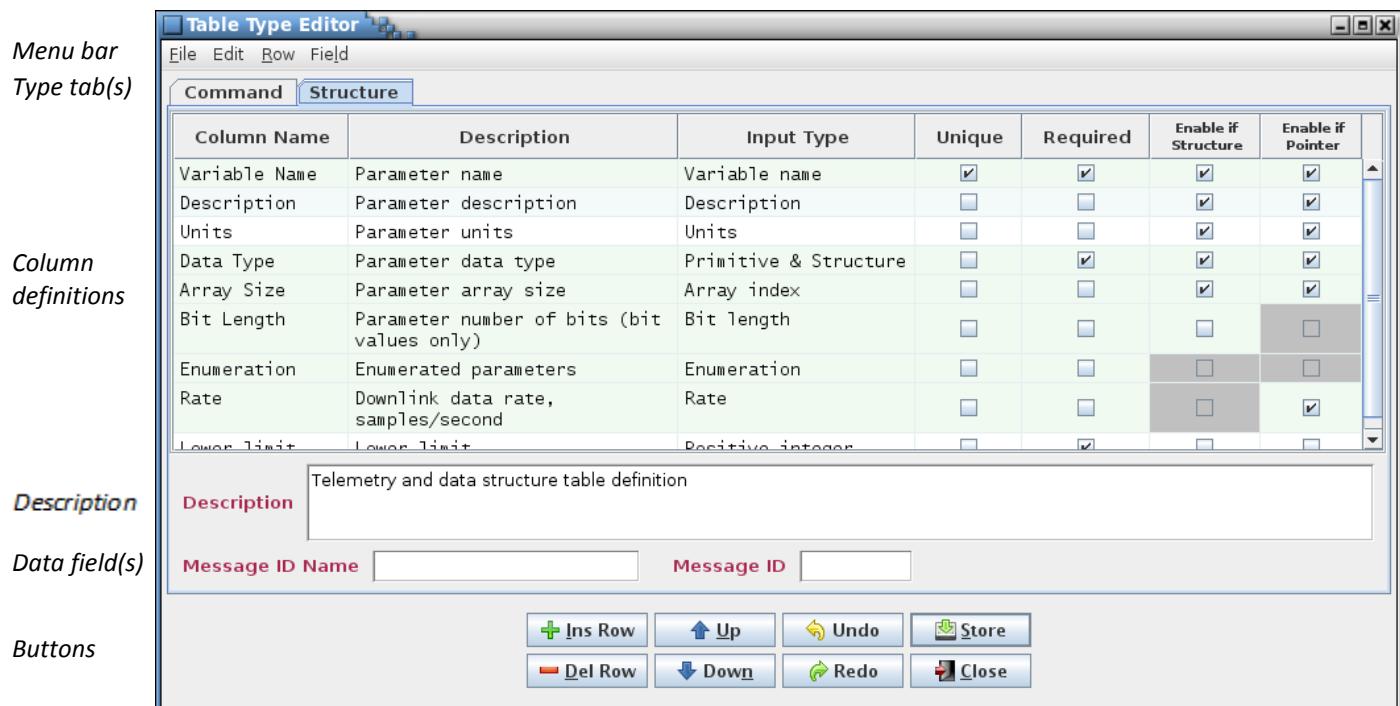


Figure 49. Table type editor

The menu bar has commands associated with the table 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. As a default this includes the Structure and Command types. Any types created by the user also appear. The tabs are arranged in alphabetical order. Selecting the tab causes the editor to display the information for the selected table type.

Each row in the editor is a definition of a column that appears in each table of this type. The order of the column definitions determines the initial column order when the table is first displayed. The editor

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 86 of 295

columns can be sorted and repositioned as described in paragraph 4.2. Note that sorting the columns using the header does not affect the column order of tables created using the type. The displayed column order can be changed for each individual table in the table editor (see paragraph 4.9.3.2.4).

Editor cells that are grayed out cannot be changed. Selections in a columns definition can preclude other selections. For example, since only integer data types are valid for an enumeration, columns with an enumeration input type can't have a data type that's a structure and the **Enable if Structure** check box is disabled.

Column definition rows are highlighted when the all of the column definitions necessary to define a structure or command table are present and the column definition is one of these rows. In Figure 49 the column definitions for the variable name, data type, array size, bit length, enumeration, and rate are highlighted since these are the ones required to define a table representing a structure. If any one of these column definitions is removed then this highlighting is removed for all of the rows. The highlighting serves to indicate that the minimum column definitions are present to define the table 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 header for the column defined on this row of the editor.
<b>Description</b>	This text is displayed as a tool tip whenever the mouse pointer hovers over this column's name in the table's header.
<b>Input Type</b>	The input type constrains the type of value entered into the table cells for this column. If the value entered into the cell doesn't conform to the 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 <b>Input Type</b> column is selected. See paragraph 4.7 for information on the available input types.
<b>Unique</b>	This check box, if selected, indicates that each cell's data value must be unique within this column. If a duplicate value is entered into a 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 the cell in this column requires a value. This causes the cell to be highlighted in yellow 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 that the information in this cell is considered important (for example, necessary to a script).
<b>Enable if Structure</b>	This check box, if selected, indicates that the cells in this column allow input when the same row's data type column contains a structure reference. If the check box isn't selected then whenever a structure is selected as the data type the table's cell is grayed out and its value is blanked. <i>Note: This editor column only appears when the table type contains all column definitions necessary to represent a structure.</i>
<b>Enable if Pointer</b>	This check box, if selected, indicates that the cells in this column allows input when the same row's data type column contains a pointer reference. If the check box isn't selected then whenever a pointer is selected as the data type the table's cell is grayed out and its value is blanked. <i>Note: This editor column only appears when the table type contains all column definitions necessary to represent a structure.</i>

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:

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 87 of 295

- 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.
- Up** 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 type. The displayed column order can be changed for each individual table in the table editor (see paragraph 4.9.3.2.4).
- 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 4.9.3.2.4).
- 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 table type editor (not those in the other tabs) in the database. See paragraph 4.9.3.10.1.5 for further details.
- Close** Closes the table 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.

#### 4.9.3.10.1 File menu

##### 4.9.3.10.1.1 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 (Figure 50). The three radio buttons allow automatically populating the new type with those columns that are required for a structure or command table type; selecting **None** results in a table type with no columns defined. Select **Create** to create the new type, which is opened in the table type editor. The editor can then be used to populate the table type with column definitions and edit any that were automatically added, and afterwards new tables of this type may be created and edited. Select **Cancel** to exit the dialog without creating a new table type.

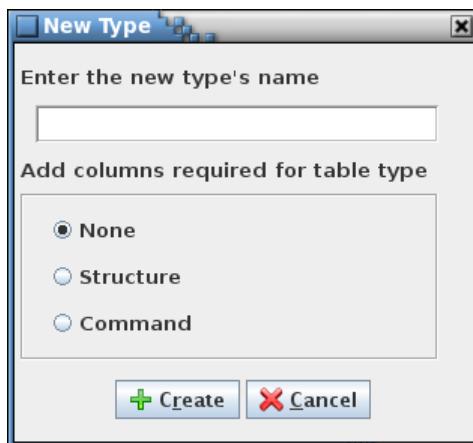


Figure 50. New table type dialog

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 88 of 295

#### 4.9.3.10.1.2 *Copy type*

The **Copy type** command is used to create a new table type from an existing one, including all of its column definitions and default data fields. The active tab in the table 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 table 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.

#### 4.9.3.10.1.3 *Rename type*

The **Rename type** command is used to rename an existing table type. The active tab in the table 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 reference the new table type name. Select **Cancel** to exit the dialog without renaming the table type.

#### 4.9.3.10.1.4 *Delete type*

The **Delete type** command deletes an existing table type. The active tab in the table 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 deleted type from the project database*. Select **Cancel** to exit the dialog without deleting the table type or any tables.

#### 4.9.3.10.1.5 *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 table type editor (not those in the other tabs) in the project database. Afterwards, any table created using this table type inherits the type's columns and data fields.

All existing tables of this type, including those in any open table editors, are updated immediately with the column and data field 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 4.9.3.10.4.3.

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

#### 4.9.3.10.1.6 *Store all*

The **Store all** command is similar to the **Store current** command described above, except that it stores the changes made to all the table type editor tabs in the project 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 and canceling it.

#### 4.9.3.10.1.7 *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), to be output to the selected printer (or file). Selecting **Cancel** removes the print dialog without printing the table type editor contents.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 89 of 295

#### 4.9.3.10.1.8 Search

Selecting the **Search** command causes a search dialog to be displayed. The behavior of this dialog is identical to that for the table editor dialog except that the search is confined to the currently displayed table type editor. See paragraph 4.9.3.2.1.8 for details on use of the search dialog.

#### 4.9.3.10.1.9 Close

The **Close** command performs the identical action to the **Close** button. The command closes the table 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.

### 4.9.3.10.2 Edit menu

#### 4.9.3.10.2.1 Copy

The **Copy** command places the contents of the highlighted cell(s) into the operating system's clipboard. This information can then be pasted into another cell or input field in the application, or into applications other than CCDD. The Ctrl-C keys perform the same operation.

#### 4.9.3.10.2.2 Paste

The **Paste** command places the contents of the operating system's 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, overwriting the existing data in the cells. If insufficient columns exist for the pasted data then the excess column information is ignored. 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 4.9.3.10.2.3 on inserting copied data without overwriting the existing cell contents. The Ctrl-V keys perform the same operation.

#### 4.9.3.10.2.3 Insert

The **Insert** command behaves similarly to the **Paste** command (paragraph 4.9.3.10.2.2) 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 perform the same operation.

#### 4.9.3.10.2.4 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 table type editor. The Ctrl-Z keys perform the same operation.

#### 4.9.3.10.2.5 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 table type editor. The Ctrl-Y keys perform the same operation.

#### 4.9.3.10.2.6 Clear data

The **Clear data** command empties all of the currently displayed editor's cells.

#### 4.9.3.10.2.7 Add command arguments

The **Add command arguments** command appends a default set of command argument column definitions to the currently displayed table type. The command is only enabled if the table type represents a command. The index for the added arguments is determined by scanning the existing argument column names for the pattern "Arg # ", where '#' is the argument index (initialized to 1). If a

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 90 of 295

column is found matching the pattern then the index is incremented and the process is repeated. This continues until no match is found; the new arguments are named using the index value.

#### 4.9.3.10.3 Row menu

##### 4.9.3.10.3.1 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 cells in multiple rows are selected then the new row is inserted below the lowest one. If no cell is selected then the new row is inserted at the end of the table. The Insert key performs the same operation.

##### 4.9.3.10.3.2 Delete row(s)

The **Delete row(s)** command performs the same action as the **Del Row** button. This command deletes the row associated with each currently selected cell. If cells in multiple rows are selected then each of the rows is deleted. If no row is selected then this command has no effect. The Delete key performs the same operation.

##### 4.9.3.10.3.3 Move up

The **Move 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 relative to the remaining rows. Since each row is a column definition, 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 4.9.3.2 and 4.9.3.2.4).

##### 4.9.3.10.3.4 Move down

The **Move 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 relative to the remaining rows. Since each row is a column definition, 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 4.9.3.2 and 4.9.3.2.4).

#### 4.9.3.10.4 Field menu

##### 4.9.3.10.4.1 Manage fields

The **Manage fields** command allows the user to create, alter, and delete default data fields for the type represented by the active table type editor tab. See paragraph 4.6 for information regarding data fields and use of the data field editor.

The fields manipulated by the field editor are displayed below the table type editor table and description when the **Update** button is pressed. The table 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 assign 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, and value (depending on the state of the **Overwrite values** check box; see paragraph 4.9.3.10.4.3) are updated in the table to match the default.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 91 of 295

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.

#### 4.9.3.10.4.2 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 affecting the data field values.

#### 4.9.3.10.4.3 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.

### 4.9.3.11 Manage data types

The Data Type Editor (Figure 51) provides a means of creating, modifying, and deleting primitive data type definitions (see paragraph 4.5.4 for more information on data types). When a project database is first created the primitive data types default to those shown in Table 5.

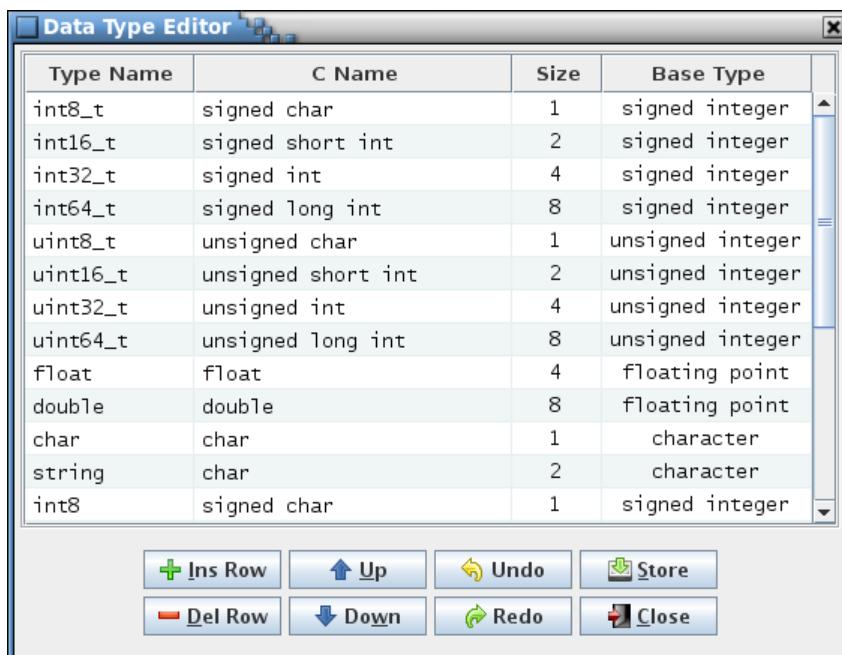


Figure 51. Data Type Editor dialog

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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 92 of 295

data type names are case insensitive and must be unique. If the type name is left blank then the text in the C Name column determines the data type name displayed in the data table cell.

<b>C Name</b>	The C name 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 <code>typedef</code> statements using the type name and C name combinations). One or more trailing asterisks are allowed if the corresponding base type is 'pointer'. The C name is used as the data type in a data table cell if the corresponding type name is blank.
<b>Size</b>	Size, in bytes, occupied by this data type. The size must be an integer greater than 0.
<b>Base Type</b>	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 Name** 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. Valid values must be entered in the **Size** and **Base Type** columns for every definition. The rows can be sorted by selecting the column headers, as with other table editors in the application.

If a pointer base type is selected then an asterisk (\*) is automatically appended to the C 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 name is 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**:



Figure 52. 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 and 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 53). 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	signed integer		
int32	signed integer		
int64	signed integer		
uint8	unsigned integer		
uint16	unsigned integer		
uint32	unsigned integer		
uint64	unsigned integer		
float	floating point		
double	floating point		

Figure 53. 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 cells in multiple rows are selected then the new row is inserted below the lowest one. 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 cells in multiple rows are selected then each of the rows is deleted. 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 relative to the remaining rows. 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 relative to the remaining rows. 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 project database. All tables are updated with the changes, including any tables currently open in a table editor.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018		
			Version 1.2.31 Page 94 of 295

- 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.

#### 4.9.3.12 Manage macros

The Macro Editor (Figure 54) provides a means of creating, modifying, and deleting macro definitions (see paragraph 4.5.7 for more information on macros).

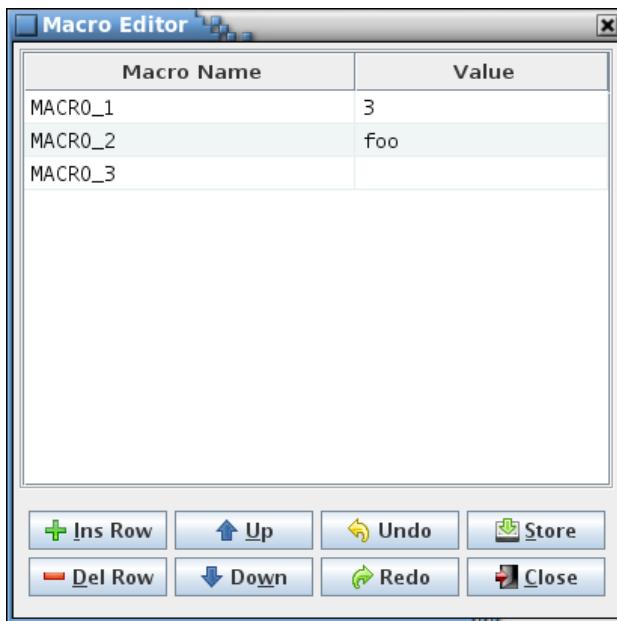


Figure 54. Macro Editor dialog

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 or macro editor cell (the macro name is delimited by pairs of '#' characters 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. The rows can be sorted by selecting the column headers, as with other table editors in the application. 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 evaluate to 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 editor cell reverts to its previous value.

Macro values can reference other macros. Any macros referenced in a macro value are highlighted. Each macro in the macro's value, when the macro is expanded, is replaced by its value. Circular macro references are not allowed (i.e., a macro references itself in its value); if detected a warning dialog is displayed and the cell contents reverts to its previous value.

Macro values can also use the `sizeof(data type)` call. Any `sizeof()` calls in the macro value are highlighted similar to macro references. When expanded the `sizeof()` call is replaced by the size in bytes of the specified primitive or structure data type.

While editing a macro value cell, the Ctrl-M keys can be used to display a pop-up list of macros for insertion into the value. Position the text cursor or highlight one or more characters to be replaced, then press Ctrl-M. The pop-up list appears as shown in Figure 55.

Macro Name	Value
MACRO_1	3
newMacro	sizeof(int32_t)
MACRO_2	eee MACRO_1 ▾
MAC1	##MACRO_1 newMacro## + 1
MAC3	##newMacro MAC1 if = MAC4
MAC4	##MACRO_2 ##
MAC5	##MAC1 MAC3 MAC4 MAC5

Figure 55. Example of macro name selection pop-up dialog in a macro value cell

Pressing Ctrl-Shift-M temporarily replaces every macro with its corresponding value in the macro editor **Values** column. Releasing the Ctrl-Shift-M keys restores the macro names in the cells.

The button commands are described below:

- |                |   |
|----------------|---|
| <b>Ins Row</b> | Inserts an empty row below the currently selected cell's row. If cells in multiple rows are selected then the new row is inserted below the lowest one. 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 cells in multiple rows are selected then each of the rows is deleted. 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. |
| <b>Up</b>      | Move the row(s) of the currently selected cell(s) up one row relative to the remaining rows. 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.   |
| <b>Down</b>    | Move the row(s) of the currently selected cell(s) down one row relative to the remaining rows. 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.   |
| <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, undo, etc.).  |
| <b>Store</b>   | 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.   |
| <b>Close</b>   | 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.  |

#### 4.9.3.13 Message IDs

**Message IDs** is a sub-menu of commands relating to message ID names and numbers, described in the following paragraphs.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Version 1.2.31

Date: May 2018

Page 96 of 295

#### 4.9.3.13.1 Assign IDs

The **Assign IDs** command provides a method for automatically assigning a unique message ID number to table cells and data fields having an input type of 'Message ID' (see paragraph 4.7). Telemetry message IDs are assigned in the telemetry scheduler (see paragraph 4.9.4.2). Automatic assignment can be limited to structure tables, command tables, other type tables, and groups as desired. A dialog appears (Figure 56) containing a tab for structure tables, command tables, other table types, and groups.



Figure 56. Assign Message IDs dialog

For each tab there are two check boxes and two input fields. The first check box is used to determine if the message IDs associated with the selected tab are to be updated. For the table type tabs this affects both table cells and data fields, and for the Group tab this affects only group data fields. If this check box is unchecked the remaining fields are disabled and ignored. The first input field is the starting ID number, in hexadecimal (the '0x' preceding the number is optional). The second 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 final check box determines whether or not existing table/group message ID numbers are updated or left as is. When checked, existing message IDs are replaced unless the ID is flagged as protected. Message IDs are protected if a '#' character is appended to the ID value (example: ox1234#).

When **Okay** is selected then the tables representing a structure, command, or other type, and group data fields (depending on whether or not the associated check box is selected) are checked to determine if they have a column(s) or data field(s) with the input type of 'Message ID.' If so, then the column/field value is assigned a message ID number. IDs are assigned beginning with the starting ID number, and with each subsequent ID number equal to the previous number plus the interval value. Table column message IDs are assigned (for all tables) before message ID data fields. Macros are allowed in the table columns representing message IDs; however, the auto-assignment process will overwrite the macros with message ID values if the overwrite check box is selected for the table's type. An ID is skipped if it is listed in the reserved message ID list (see 4.9.3.13.2), assigned to table columns and/or data fields for a table type that doesn't have the overwrite check box selected, assigned to a group data field (unless the Group overwrite check box is selected), or assigned to telemetry messages in the telemetry scheduler (see paragraph 4.9.4.2). If the overwrite check box isn't selected then the values for any existing structure (command, other, group) message ID data fields are also skipped in order to avoid duplicate ID values. This action also updates the project database and the message ID number columns and/or data fields for any open table editors. Select **Cancel** to exit the dialog without altering the message ID values.

#### 4.9.3.13.2 Reserve IDs

The editor dialog shown in Figure 57 to appears when the **Reserve message IDs** command is selected. This dialog allows message IDs, either singly or as ranges, to be flagged as reserved. This means that when automatic assignment of message IDs is performed (see paragraphs 4.9.3.13.1 and 4.9.4.2) the message IDs in the reserved table are skipped.

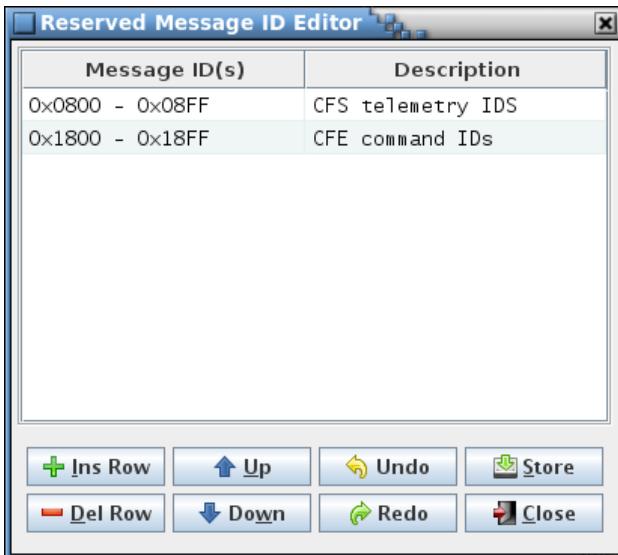


Figure 57. Reserved Message ID Editor dialog

The editor column descriptions are as follows:

**Message ID(s)** This column can contain a single hexadecimal number, optionally prepended with “0x”, or a range of IDs consisting of two hexadecimal numbers separated by a hyphen (-). If a range is entered the second number must be greater than the first. Message IDs are not allowed to be duplicated in the table. This includes IDs falling within an existing range or overlapping of two ranges.

**Description** This column can be used to describe the ID or ID range. It may remain empty.

Each row in the table is a reserved message ID definition. Every definition requires a message ID or ID range, but the description may be blank. If an invalid or duplicate value is entered a dialog appears indicating the problem, and the editor cell reverts to its previous value. The rows can be sorted by selecting the column headers, as with other table editors in the application.

The button commands are described below:

**Ins Row** Inserts an empty row below the currently selected cell’s row. If cells in multiple rows are selected then the new row is inserted below the lowest one. 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 cells in multiple rows are selected then each of the rows is deleted. If no row is selected then this has no effect.

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

**Down** Move the row(s) of the currently selected cell(s) down one row relative to the remaining rows. The order of the reserved message ID definitions in the editor has no effect on ID usage. The capability to arrange the rows is solely for the user to group the IDs 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.).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018		
	Version 1.2.31 Page 98 of 295		

- Store** Stores the changes made to reserved message IDs or description in the reserved message ID editor into the project database.
- Close** Closes the reserved message ID 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.

#### 4.9.3.13.3 Show all IDs

The **Show all IDs** command displays a table showing all of the message IDs in a project along with their corresponding message ID names and the entity (table or group) in which the ID and name are found. Figure 58 is an example of the table produced by the command. The pairing of message IDs and ID names depends on assigning specific input types to the table columns and data fields. For a message ID to be recognized as such it must have the input type **Message ID**, and a message ID name column or field must have the input type **Message ID name** (see paragraph 4.7 for more information on input types).

The pairings are made in the order the IDs and names appear, and the paired values must have the same owner (this prevents pairing an ID from one table with an ID name from another one for example). If the number of IDs/names is a table or group of data fields isn't the same then the excess IDs/names are assigned blank names/IDs in the table. Lack of a corresponding ID or name can lead to erroneous pairings, so care should be taken, if this feature is used, to always create ID and name columns and fields together, and ensure the input types are correct.

A script data access method, `getMessageIDOwnersIDsAndNames`, is provided that returns the same information as is produced by this command; see the reference in Table 9 for details.

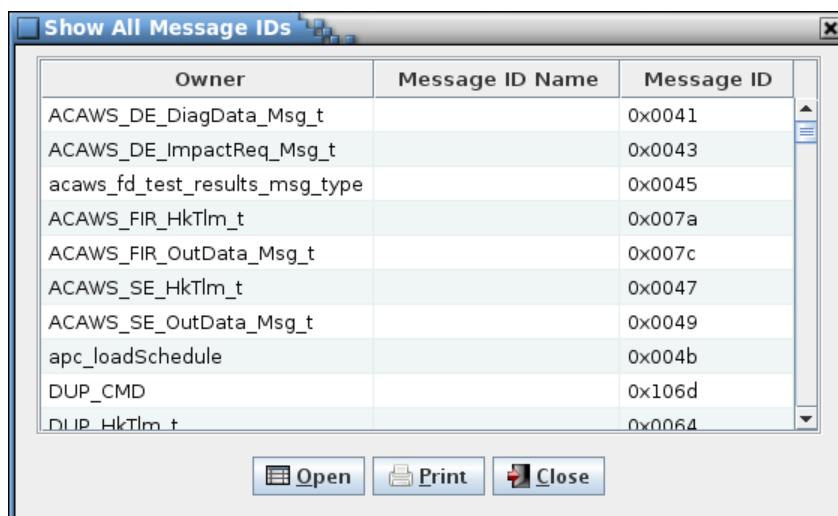


Figure 58. Example Show all IDs dialog

If one or more cells is selected and the **Open** button is pressed then the table(s) associated with the selected cell(s) are opened in a table editor. A row is ignored if it contains a message ID belonging to a group or telemetry message.

Selecting the **Print** button opens a printer selection dialog in order to print a copy of the table to the selected printer or file. Selecting **Close** closes the message ID dialog.

#### 4.9.3.13.4 Find duplicates

The **Find duplicates** command searches the project database table cells, data fields, and telemetry messages for message ID values used more than once, and displays a dialog showing these IDs and

where they are located (see the example dialog in Figure 59). If no duplicate message ID exists the dialog's table is empty.

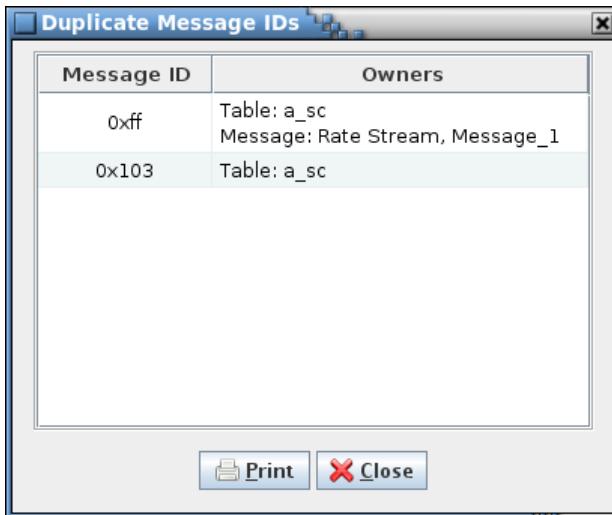


Figure 59. Example Duplicate Message IDs dialog

The **Message ID** column shows the duplicated ID as a hexadecimal value; the table is sorted based on the ID value. The **Owners** column displays the location(s) where the message ID is referenced – this can be a table cell or data field (table path and name is preceded by “Table:”), or telemetry message (message data stream and message name is preceded by “Message:”). The rows can be sorted by selecting the column headers, as with other tables in the application. Column order can be changed by dragging a column to a new position.

Selecting **Print** causes the dialog contents to be output to the selected printer (or file). Selecting **Close** closes the duplicate message IDs dialog.

#### 4.9.3.14 Manage project fields

The **Manage project fields** command causes the project data field manager dialog, shown in Figure 60, to appear. Data fields (see paragraph 4.6) may be associated with a project, similar to how they can be associated with specific data tables and groups. The project's description can also be edited via this dialog (the description can also be edited using the **Rename** command in the **Project** menu; see paragraph 4.9.2.4).

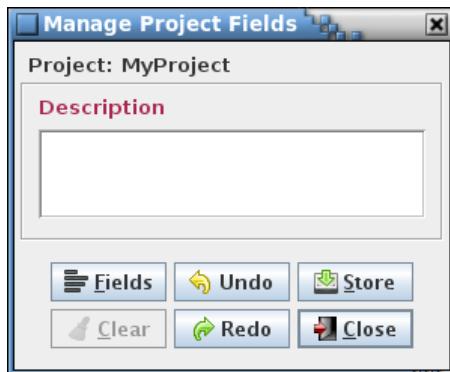


Figure 60. Project data field management dialog

The **Description** field appears by default and cannot be deleted. The **Fields** button causes the data field editor (see paragraph 4.6.1 for details on its use) to appear; data fields can be added, modified, or removed via this editor.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 100 of 295
---	---	-----------------------------------

The project data field manager button commands are summarized below:

- |               |  |
|---------------|--|
| <b>Fields</b> | Invokes the data field editor in order to create, alter, and delete data fields for the project.   |
| <b>Clear</b>  | Replace the values in all data fields with blanks and deselect any check box data field for the project.   |
| <b>Undo</b>   | Undoes the last action performed (typing, paste, insert, delete, redo, etc.) in the manager dialog.  |
| <b>Redo</b>   | Reverses the last action undone (typing, paste, insert, delete, undo, etc.) in the manager dialog.   |
| <b>Store</b>  | Stores the changes made to the project description and data fields in the manager dialog into the project database.  |
| <b>Close</b>  | Closes the project data field manager 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 dialog. |

#### 4.9.3.15 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 via 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 61 for an example; if no tables exist or no data fields are currently assigned then a warning dialog appears instead indicating there is nothing to display.

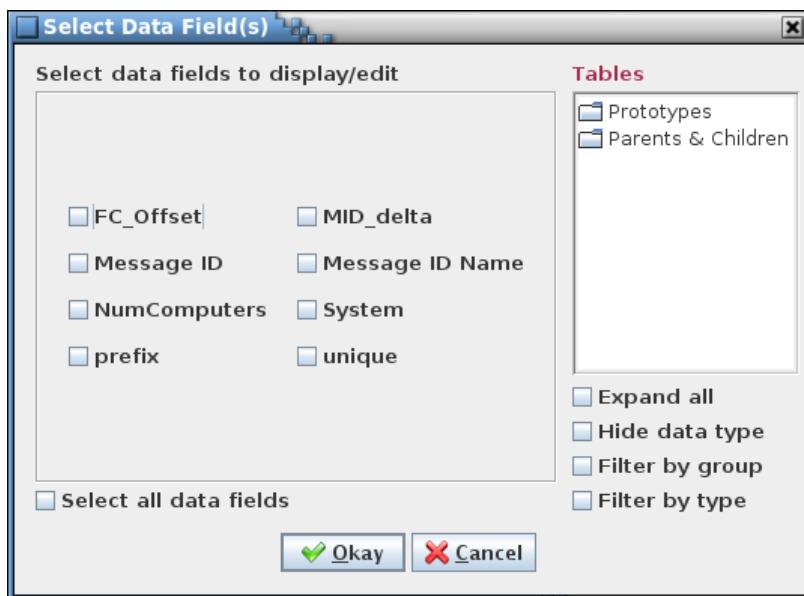


Figure 61. Example Select Data Field(s) dialog

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. Selection of a header node in the table tree (e.g., ‘Parents & Children’, or a group name if group filtering is enabled) selects all tables under that header. 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. An example of the editor dialog that appears is shown in Figure 62.

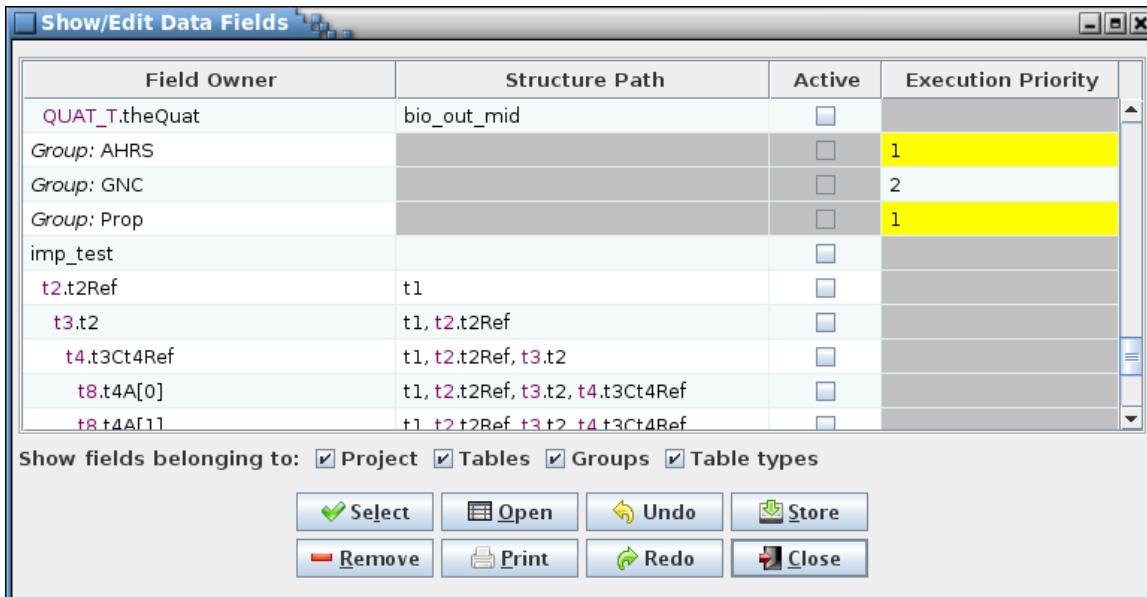


Figure 62. Example Show/Edit Data Fields dialog

The first column, **Field Owner**, 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 root structure. Project-level fields display “*Project:*” as the field owner. Fields that belong to a group display the group's name with “*Group:*” prepended, and default data fields (those belonging to a table type definition) display the table type name prepended with “*Type:*”.

The second column, **Structure Path**, displays the structure path for child tables, listing each prototype and instance pair in the child structure's path leading back to its root structure. The root structure is shown first, then each subsequent child prototype and instance in the path. If the field owner is not a child structure then the structure path is this row has a gray background. The column is not displayed if there is no child structure table field owner in any row of the editor. For example, note the row in Figure 62 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 root 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, non-blank value. 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.

Below the editor table are a number of filter check boxes that determine the type(s) of data fields displayed in the table. All of the filters are initially selected. The filters are as follows:

**Project**      Display the project-level data fields (see paragraph 4.9.3.14).

**Tables**      Display data fields belonging to the data tables (see paragraph 4.9.3.2.5.1).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 102 of 295

**Groups** Display group data fields (see paragraph 4.9.3.9).

**Table types** Display the default data fields; those belonging to a table type definition (see paragraph 4.9.3.10.4.1).

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 operation and discarding the changes, or canceling it and retaining the current selection with its unstored changes. The current selection state of the data field type filter check boxes is retained when the newly selected fields are displayed.

Data field values can be altered or the entire field removed via the editor. To change a field's value 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 cell(s) are opened in a table editor. A row is ignored if it contains a data field belonging to a group or table type.

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

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 and canceling it.

Selecting **Close** closes the data field editor dialog. If a change has been made to a data field that hasn't been stored in the project 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.

#### 4.9.3.16 Padding

The **Padding** sub-menu displays several commands for adjusting the byte alignment used when padding variables are automatically inserted and for adding or removing the padding variables from the structure tables.

##### 4.9.3.16.1 Set alignment

When selected, a dialog appears allowing setting of the word byte size value (Figure 63). A 4-byte word size (the default) corresponds to a 32-bit architecture. The selected alignment value is retained between sessions. The variable padding isn't updated until the **Add/update** command is issued (paragraph 4.9.3.16.2)

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 103 of 295

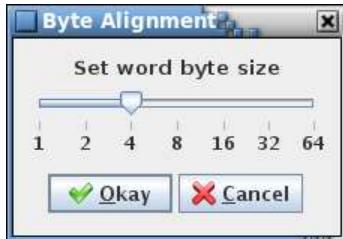


Figure 63. Byte alignment selection dialog

While the padding is being added/updated or removed a dialog appears (Figure 64) showing the adjustment progress. This dialog allows halting padding adjustment by pressing the **Halt** button. Padding may be altered in the project database for some of the structures depending on when in the adjustment process the **Halt** button is pressed.



Figure 64. Example padding adjustment progress/cancellation dialog

#### 4.9.3.16.2 Add/update

The **Add/update** command inserts padding variables into the structure tables as needed to align the variables based on the selected word byte size value (see paragraph 4.9.3.16.1). The padding variables are of data type *char* and are represented as a single variable or an array of variables (if needed for multiple, consecutive padding). The exception for the data type is that padding variables added to “fill out” the unused bits for a bit-wise variable or series of bit-packed variables has the same data type as the bit-wise variable(s). The padding variable names are in the format *\_pad#* (where # is one or more numerals). Padding variables are highlighted in the structure tables; see Figure 65. These variable rows may be manually edited as with any other row – added, altered, or removed – but are only recognized as the automatically inserted variety if they conform to the aforementioned name and data type constraints. If any data table has unstored changes a confirmation dialog appears allowing the user to choose between continuing with the padding add/update operation (losing the unstored changes) and canceling it (allowing the changes to be stored).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 104 of 295
---	---	-----------------------------------

The screenshot shows a software window titled "primonly" with a menu bar: File, Edit, Row, Column, Field. Below the menu is a toolbar with buttons for Ins Row, Up, Left, Undo, Store, Del Row, Down, Right, Redo, and Close. The main area is a table with columns: Variable Name, Description, Units, Data Type, Array Size, Bit Length, Enumeration, and Rate. The table contains the following data:

Variable Name	Description	Units	Data Type	Array Size	Bit Length	Enumeration	Rate
a			char				
_pad0			char				
b			int16_t		3		
c			int16_t		1		
_pad1			int16_t		12		
d			uint8_t	5			
_pad2			char				
e			uint16_t				
f			uint16_t				
g			char				
_pad3			char				
h			int64_t				
i			int8_t				
_pad4			char	3			

Figure 65. Structure table showing highlighted padding variables

#### 4.9.3.16.3 Remove

Execution of the **Remove** command causes all padding variables to be removed from the structure tables. Only variables conforming to the name and data type constraints outlined in paragraph 4.9.3.16.2 are recognized as padding variables. If any data table has unstored changes a confirmation dialog appears allowing the user to choose between continuing with the padding removal operation (losing the unstored changes) and canceling it (allowing the changes to be stored).

#### 4.9.3.17 Show variables

The **Show variables** command causes a dialog to appear that displays the variable paths and names, in alphanumeric order, for tables representing structures in the project database (see Figure 66 for an example). The variables displayed can be constrained by selecting one or more tables in the table tree. Only variables for the selected tables are displayed. Selection of a header node in the table tree (e.g., 'Parents & Children', or a group name if group filtering is enabled) selects all tables under that header. If no table or header node is selected in the tree then all variables in the project are displayed. The number of variables matching the constraints is displayed beside the variable table's label.

The paths and names are shown in two formats. The **Application Format** column shows the format used within the application:

```
rootTable
[ ,structureDataType1.structureVariableName1
 [ ,structureDataType2.structureVariableName2
 [ ... ] ],
 primitiveDataType.variableName[[arrayIndex]]
```

The variable path and name is a combination of the structure's root table, ancestor structure(s) (if any), and the variable data type and name. This combination is unique for each variable defined in the

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 105 of 295

project database. Note that any macro embedded in a variable name is replaced by its corresponding value before being displayed in the dialog.

The **User Format** column shows the path and name based on the user inputs. The **Enter variable path separator character(s)** text field allows entering the character(s) that are used to replace the commas that separate each variable in the variable path. The **Enter data type/variable name separator character(s)** text field allows entering the character(s) that are used to replace the periods that separate the data types and variable names. The **Hide data types** check box, if selected, causes the data types (structure and primitive) to be removed from the path, along with the periods that separate the data types from the variable names. The data type/variable name separator field is disabled and ignored for this case. Array member indices are altered by replacing the left bracket ([]) with an underscore (\_) and removing the right bracket (]). The **User Format** column values are identical to that returned by the script data access method call (see Table 9):

```
getFullVariableName("variable path + name", path_separator,  
hide_data_types, type_and_name_separator)
```

To perform a conversion enter the separator character(s) and set the data type check box, then press the **Show** button. The **User Format** column updates to display the variables in the new format. The **Print** button outputs the table to a user-specified printer or file. The **Store** button stores the separators and show/hide data type state as a program preference so that these are restored if the dialog is closed and reopened. Additionally, this sets the separators used in displaying the variable path column; any open tables displaying a column with the variable path input data type are automatically updated. Select the **Close** button to exit the dialog.

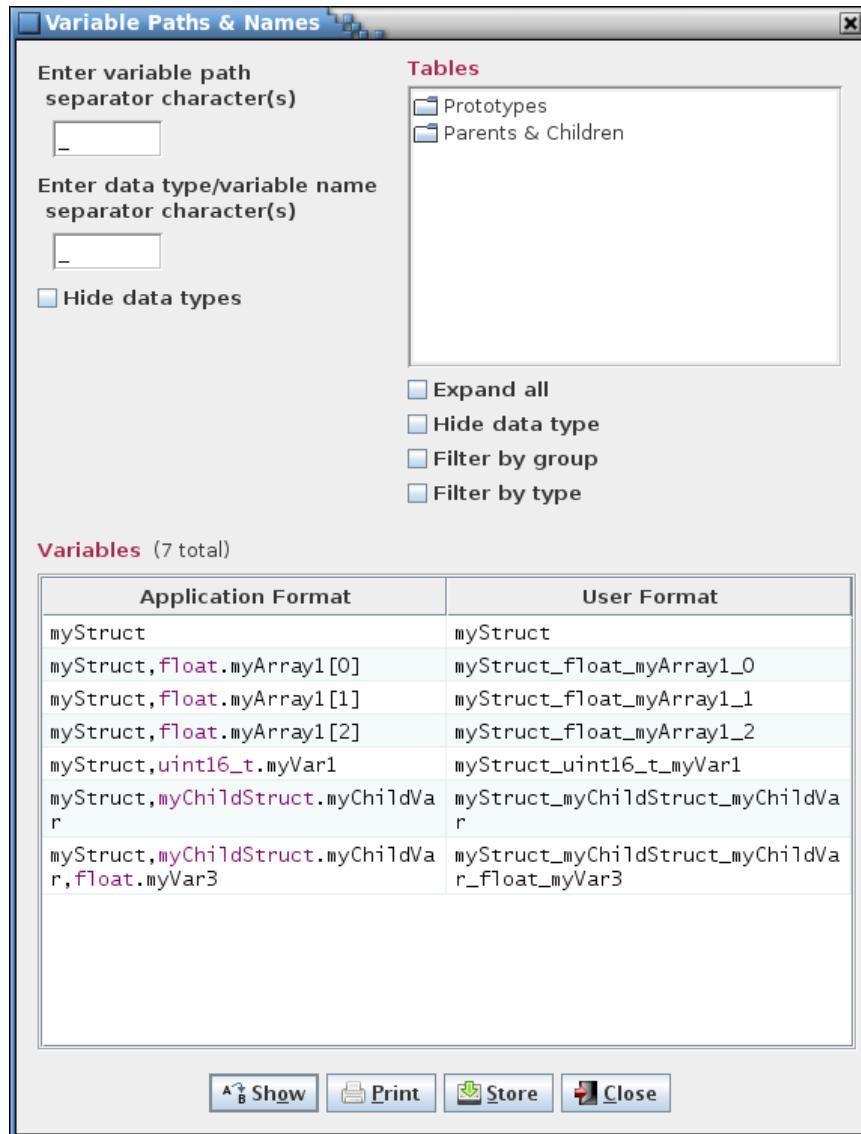


Figure 66. Example variable paths &amp; names dialog

#### 4.9.3.18 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 67). Case sensitivity for the search is governed by the **Ignore text case** check box.

The **Allow regular expression** check box, when checked, allows the use of a regular expression to define the search pattern in the search text field. A regular expression can be formulated to find multiple matching conditions (for example, the search for **a.c** would match any string that has a single character between the characters 'a' and 'c'). Information regarding the use of regular expressions is beyond the scope of this document; however, resources and tutorials can be found online. When unchecked, the search text is matched as typed in the search text field.

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.4; data table cell values stored in the custom values tables are included in the search).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
Doc. No. JSC-37494	Version 1.2.31
Date: May 2018	Page 107 of 295

The search can be constrained by selecting one or more tables in the table tree. Only matches in the selected tables are reported. Selection of a header node in the table tree (e.g., 'Parents & Children', or a group name if group filtering is enabled) selects all tables under that header. If no table or header node is selected in the tree then all tables are searched.

Enter the search text in the input field and select the **Search** button. The search results are displayed in the dialog's **Search results** table. The number of results is displayed beside the results table's label. The first column, **Owner**, 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 text field uses auto-completion to fill in the search string. The previous search strings (those for the event log, table, and script) are remembered, including those from previous sessions. The number of remembered search strings can be changed via the Preferences dialog, and defaults to 30. Case sensitivity for auto-completion is based on the **Ignore text case** check box selection state.

The input text can be changed and the **Search** button pressed again to initiate another search of the tables. 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. A row is ignored if it contains a reference to other than a table. 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.

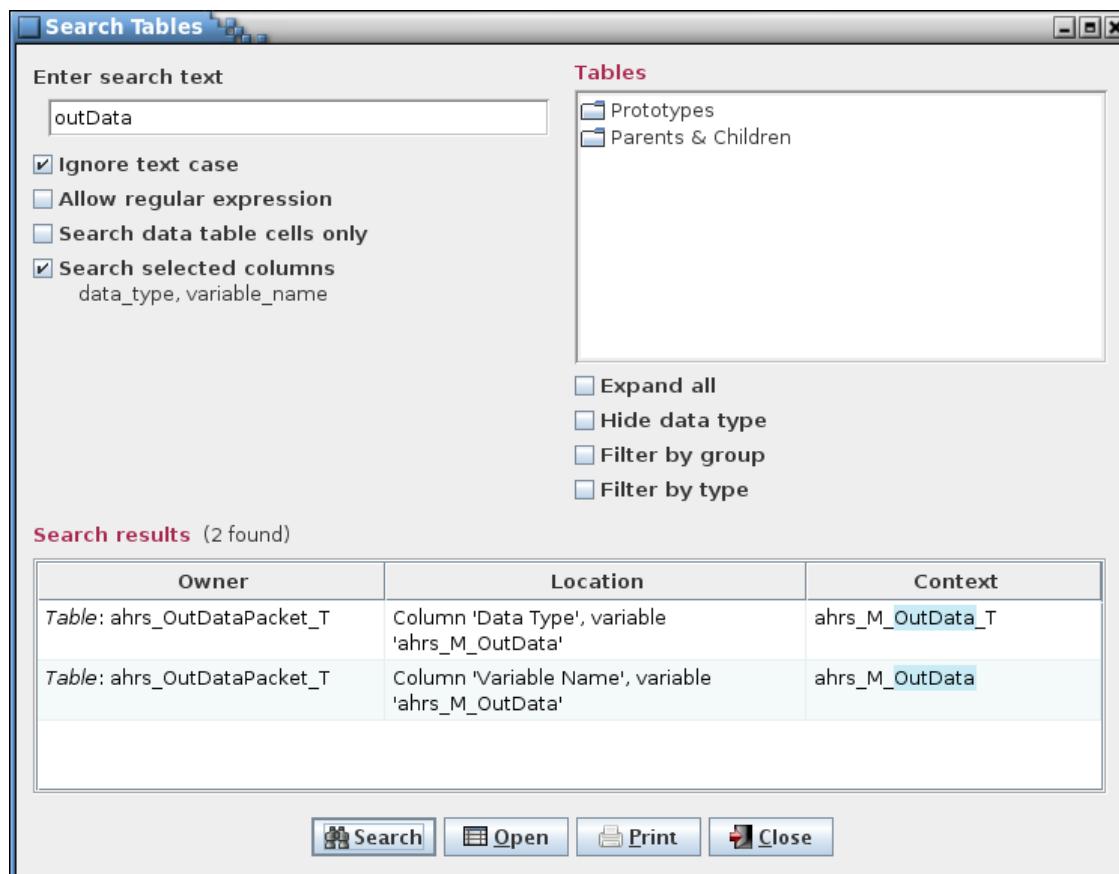


Figure 67. Search tables dialog

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 108 of 295
---	---	-----------------------------------

## 4.9.4 Scheduling

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

### 4.9.4.1 Manage links

The **Manage links** command opens the Manage Links dialog (Figure 68). This command is disabled if no rate columns are defined. 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 4.9.4.2) 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.

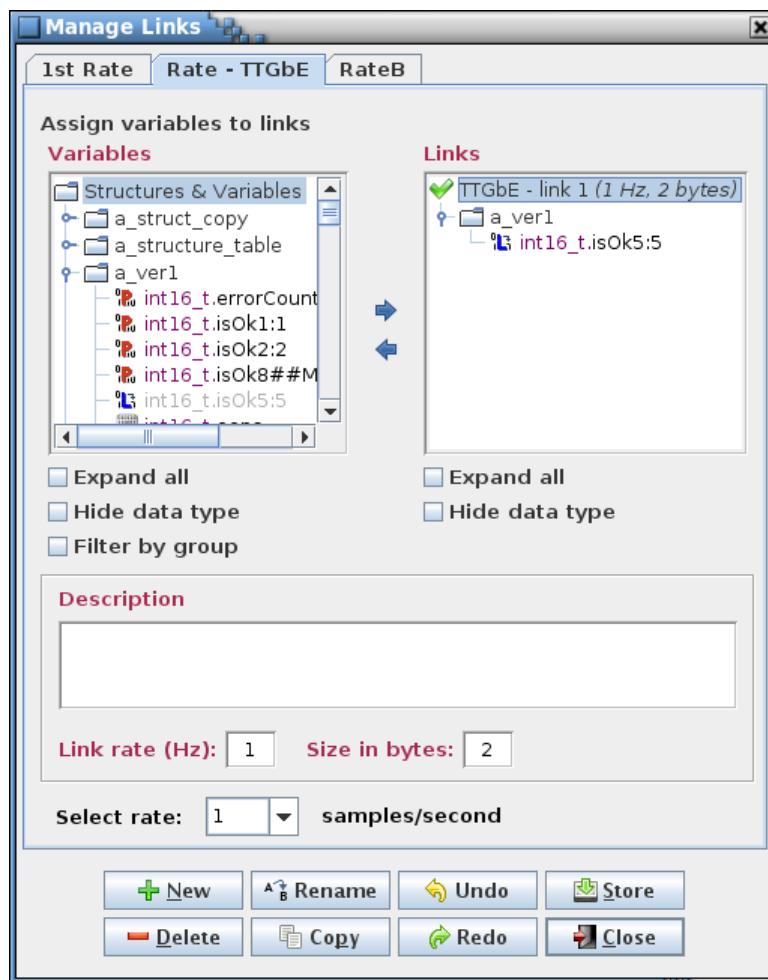


Figure 68. Manage Links dialog

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. The variables that are displayed in the tree are determined by the rate chosen from the **Select rate** combo box pull down menu near the bottom of the dialog. Rate values are grayed out and can't be selected if there is no variable that has that rate assigned. 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
Doc. No. JSC-37494		Version 1.2.31
Date: May 2018	Page 109 of 295	

adding or removing a variable from a link. Each tree also has one or more check boxes, to expand/collapse the tree and 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.

The space separating the variable and link trees delineates a split pane control that is used to resize these panels relative to one another. Position the mouse pointer between the two tree panels and when the pointer changes to a double-headed arrow press and hold the left mouse button. Space permitting, the adjoining panes can be resized by moving the mouse pointer left or right. Release the mouse button to exit resizing.

In the link tree, displayed in parentheses next to each link name, are 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 (the tree text is also grayed out for incompatible links).

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.

If a variable is selected in the **Variables** tree then the link to which it belongs is selected automatically in the **Links** tree. Since linked variables are disabled in the **Variables** tree the variable isn't highlighted when selected. Selecting a non-linked variable deselects any highlighted link in the **Links** tree.

To create a link select the **New** button and provide a link name and, optionally, a description, in the input dialog that appears (Figure 69). 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 69. New Link dialog

To add variables to a link select the link in the link tree using the mouse or keyboard. Expand the variable 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 its root structure, but does not include any of its sibling variables (i.e.,

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 110 of 295
---	---	-----------------------------------

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 4.5.5). 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 link 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 link 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** 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 link tree to be copied, then press the **Copy** button. A dialog appears (Figure 70) 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). Select **Cancel** to exit the copy dialog without copying the link.

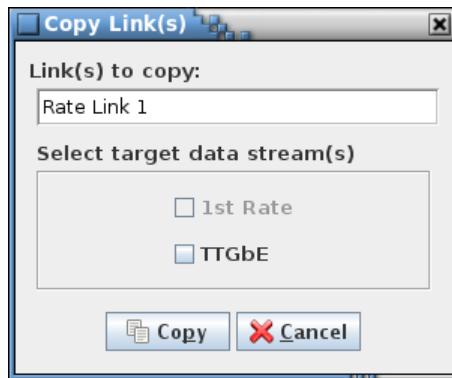


Figure 70. Copy Link(s) dialog

If the targeted data stream doesn't support the link's sample rate or the link name already exists in the stream then the link isn't copied to that stream. If a variable's sample rate differs between the copied stream and the target stream, or if the variable is unavailable in the target stream (i.e., the structure containing the variable doesn't have the rate column corresponding to the target data stream) then the variable isn't copied. For these cases a dialog is displayed indicating which link(s) or link member variable(s) could not be copied to which stream(s) and the reason for the failure(s) (see example in Figure 71). The **Print** button allows outputting the copy failure table to the selected printer or file. The **Close** button exits the copy failure dialog.

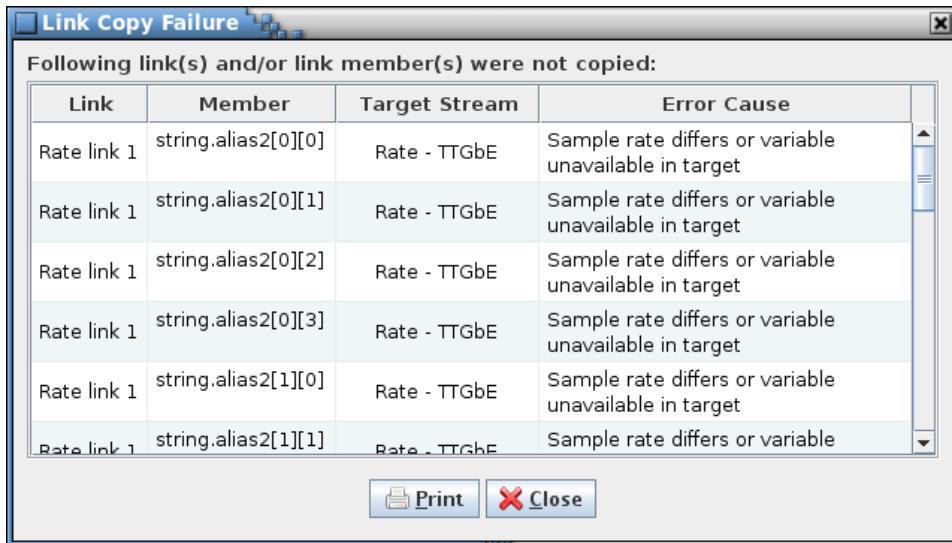


Figure 71. Example link copy failure dialog

A link's description can be added or changed by first selecting the link in the link 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 project 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 project database.

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.

The link manager button commands are summarized below:

<b>New</b>	Create a new link.
<b>Delete</b>	Delete the selected link(s).
<b>Rename</b>	Rename the selected link.
<b>Copy</b>	Copy the selected link, including its member tables, so another data stream.
<b>Undo</b>	Undoes the last action performed (table assignment, typing, paste, insert, delete, redo, etc.) on the selected link.
<b>Redo</b>	Reverses the last action undone (table assignment, typing, paste, insert, delete, undo, etc.) on the selected link.
<b>Store</b>	Stores the changes made to the links in the link manager into the project database.
<b>Close</b>	Closes the link manager 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.

#### 4.9.4.2 Telemetry

The **Telemetry** command opens the Telemetry Scheduler dialog (Figure 72). This command is disabled if no rate columns are defined. The telemetry scheduler is used to assign a project's variables to telemetry messages. The message information can 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 4.9.4.4). 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 4.9.3.2)
- (Optional) Assign variables that are desired to be sent down in the same message to a link using the link manager (see paragraph 4.9.4.1)

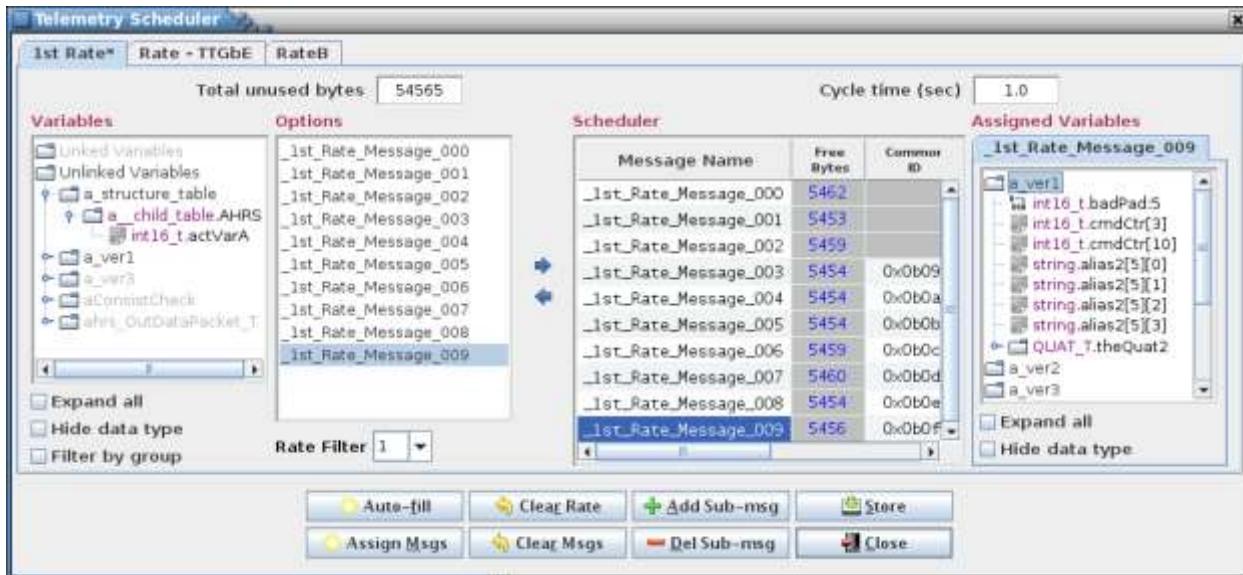


Figure 72. Telemetry Scheduler dialog

The Telemetry Scheduler dialog is composed of number of components. Along the top are tabs for each defined data stream (see paragraph 4.8). 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 the 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 at least once every 2 seconds.

The **Variables** tree, at the left of the dialog, displays in tree format the variables available for downlink. 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 data stream and rate filter. Each link contains 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 selected. 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. Rate values are grayed out and

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 113 of 295

can't be selected if there is no variable in the data stream that has that rate assigned. The variable tree only displays variables that have a rate that matches the selected rate filter value. 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.

The **Variables**, **Options**, **Scheduler**, and **Assigned Variables** portions of the telemetry scheduler can be resized. Position the mouse pointer between adjoining panes and when the pointer changes to a double-headed arrow press and hold the left mouse button. Space permitting, the adjoining panes can be resized by moving the mouse pointer left or right. Release the mouse button to exit resizing.

At the bottom of the telemetry scheduler dialog is the button panel. The button 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. During the auto-fill operation a progress/cancellation dialog appears (Figure 73). Pressing the dialog's **Halt** button stops the auto-fill operation; however, any variable assignments made up to that point are retained.



Figure 73. Telemetry message auto-fill progress/cancellation dialog

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.

- Assign Msgs** Opens the Assign Telemetry Messages dialog (Figure 74). 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.

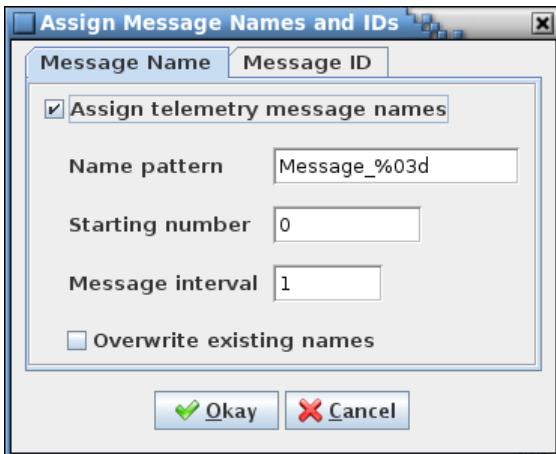


Figure 74. Assign telemetry message names and IDs dialog

To assign message names the **Assign telemetry message names** check box under the **Message name** tab 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 4.7) 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 74 the message names are "Message\_001", "Message\_002", "Message\_003", etc., until all messages are named.

To assign message IDs the **Assign telemetry message IDs** check box under the **Message ID** tab 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. Message IDs in the reserved message ID list (see 4.9.3.13.2) and IDs already assigned to message ID data fields for structure and command tables are automatically skipped when assigning IDs to the telemetry messages.

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. The names and/or IDs are assigned to the current data stream; the process must be repeated for each data stream. Avoid using the identical message ID name pattern for different streams since this can lead to duplicate output message ID names when creating the housekeeping copy table.

- Clear Rate** Removes all message and sub-message variable assignments for variables with a rate matching the currently selected rate filter.
- Clear Msgs** Removes all message and sub-message variable assignments.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Version 1.2.31

- 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.
- 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.
- 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.

If a variable is selected in the **Variables** tree then every message to which it belongs is selected automatically in the **Scheduler** table, including sub-messages. The first message to which the variable belongs sets the tab in the **Assigned Variables** tree. The option corresponding to the message selection is highlighted in the **Options** list. Since assigned variables are disabled in the **Variables** tree the variable isn't highlighted when selected. Choosing a non-assigned variable in the **Variables** tree does not change the option and message selections.

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 variables 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 deciding 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** 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 message(s) 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.

#### 4.9.4.3 Applications

The **Applications** command displays the Application Scheduler dialog (Figure 75). The application scheduler is used to schedule the execution frequency and order of a project's applications. The data

created from the application scheduler is used to create scheduler tables for the project. The schedule table is used by the CFS scheduler application (SCH) to determine when to execute the project's applications (demonstration scripts are provided that create the scheduler tables). The available time slots for when an application can be executed are determined by the application parameters that can be altered in the application parameters dialog (see paragraph 4.9.4.5). Before the application scheduler can be used the following must be done:

- Set the application parameters to establish the correct boundaries
- Create applications using the **Group Manager** dialog (see paragraph 4.9.3.9)

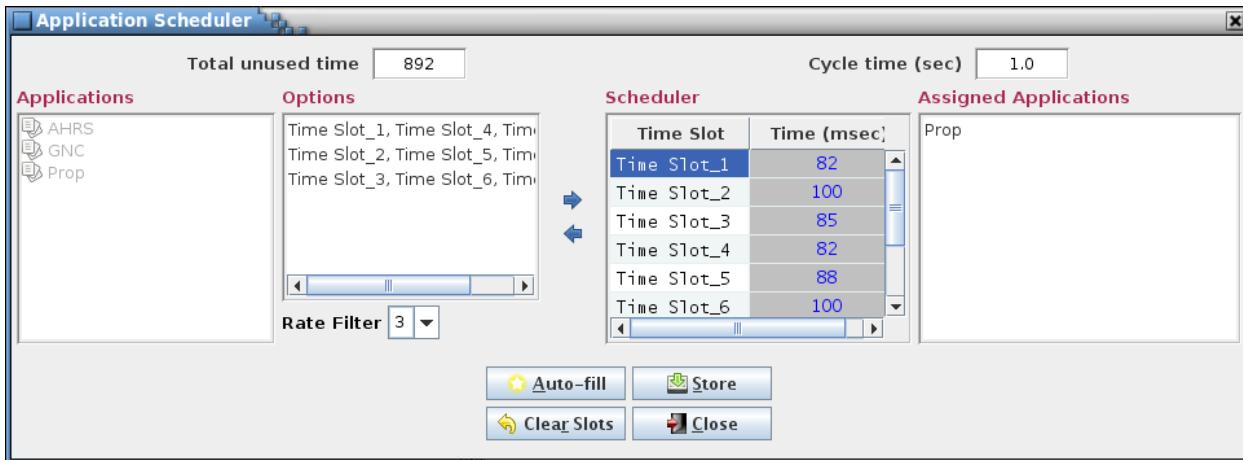


Figure 75. Application Scheduler dialog

The application scheduler dialog is composed of multiple components. At the top left is displayed the total number of milliseconds remaining to be assigned. At the upper right is the cycle time which is amount of time it will take for the schedule to repeat; e.g., a cycle time of 2 seconds means that the schedule table will be executed once every 2 seconds.

The **Applications** tree, at the left of the dialog, displays all the available applications to be scheduled. Any application that has already been assigned or has an execution rate that does not match the rate filter is grayed out and cannot be selected.

To the right of the **Applications** tree is the **Options** list. This list displays the available options, based on the selected rate filter, for assignment of the applications to the time slots. For example, if the rate filter is set to 1 and forty time slots are available then the options will be "Option1: TimeSlot\_1", "Option2: TimeSlot\_2", "Option3: TimeSlot\_3", etc. for all forty time slots (assuming the cycle time is one second).

The **Rate Filter**, which is located below the **Options** list, contains a drop-down list of all the available execution rates. Changing the rate causes the **Applications** tree to gray out any applications that are not at the selected rate, and the **Options** list changes to display options for the newly selected rate.

The **Scheduler** table, which is located to the right of the **Options** list, is a table of all available time slots. The **Time Slot** column displays the time slots and the **Time (msec)** column displays the remaining available time for that time slot (in milliseconds). The available time decreases as applications are added to that time slot and increase if an application is removed. If the available time ever becomes negative (i.e., the sum of the assigned applications' execution times exceeds the maximum available for a time slot) then the time slot's text changes to red.

The **Assigned Applications** list, located to the right of the **Scheduler** table, displays the application(s) assigned to the currently selected time slot. If multiple time slots are selected only the first selected

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 117 of 295
---	---	-----------------------------------

time slot's applications are displayed. This allows the user a quick way to view the applications currently assigned to a time slot.

In between the **Options** list and **Scheduler** table are two arrow buttons. The right arrow button is used to assign one or more applications to a time slot. The left arrow button removes one or more applications from a time slot. The assignment and removal processes are described below.

The **Applications**, **Options**, **Scheduler**, and **Assigned Applications** portions of the application scheduler can be resized. Position the mouse pointer between adjoining panes and when the pointer changes to a double-headed arrow press and hold the left mouse button. Space permitting, the adjoining panes can be resized by moving the mouse pointer left or right. Release the mouse button to exit resizing.

At the bottom of the application scheduler dialog is the button panel. The button functions are as follows:

- Auto-fill** Assigns all the applications in the application tree that are not yet assigned to time slots. Auto-fill does this optimally so each slot is filled as evenly as possible. During the auto-fill operation a progress/cancellation dialog appears (Figure 76). Pressing the dialog's **Halt** button stops the auto-fill operation; however, any application assignments made up to that point are retained.



Figure 76. Application time slot auto-fill progress/cancellation dialog

If auto-fill is successful then all the applications are assigned to a time slot. If auto-fill is unable to assign every application (due to insufficient room or no available option) it displays a dialog indicating how many applications are left unassigned.

- Clear Slots** Removes all application time slot assignments.
- Store** Stores the application scheduler data in the project database. Any changes not stored before closing the application scheduler dialog are lost.
- Close** Closes the application scheduler dialog. If there are any unsaved changes a dialog appears requesting confirmation to discard the changes. Select **Okay** to exit the application scheduler, losing any unsaved changes. Select **Cancel** to return to the application scheduler dialog.

The following describes the process to manually assign an application to a time slot. First, one or more applications are selected in the **Applications** tree. A grayed-out application indicates that it is already assigned or doesn't have the same rate as that shown in the **Rate Filter**, and can't be selected. After selecting one or more applications an option is chosen from the **Options** list. To aid in deciding which option to choose, the **Scheduler** table temporarily subtracts the application run time(s) from the **Time (msec)** column for the selected option's time slot(s), displaying the time remaining if that option is chosen. Also, the text of the **Time Slot** column changes to either green, signifying there is enough room in the slot for the application(s), or red, signifying there is insufficient room. Changing which option is selected resets any of the temporary changes and updates the time slot(s) based on the new option. This allows evaluation each option before committing to a choice. After deciding on an option, pressing

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 118 of 295
---	---	-----------------------------------

the dialog's right arrow button assigns the selected applications(s) to each time slot in the selected option. Once an application is assigned to a time slot it is grayed out in the **Applications** tree so it can't be assigned more than once.

An application or applications can be removed manually from the time slot(s). First a time slot is selected in the **Scheduler** table; this causes the **Assigned Applications** list to display the applications for the selected slot. Select from the list one or more of the applications that are to be removed and then press the dialog's left arrow button. The selected application(s) is removed from the slots to which it is assigned. Any de-assigned application is no longer grayed out in the **Applications** tree, signifying it is available to be re-assigned.

#### 4.9.4.4 Rate parameters

The **Rate parameters** command displays the dialog shown in Figure 77. This command is disabled if no rate columns are defined.



Figure 77. Rate Parameters dialog

This dialog is used to set the bounds for the sample rates for each defined data stream (see paragraph 4.8) 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 4.9.4.1), and the telemetry scheduler (see paragraph 4.9.4.2). 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 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.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 119 of 295
---	---	-----------------------------------

<b>Maximum messages per second</b>	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 <b>Maximum messages per cycle</b> value.
<b>Data stream name</b>	This value is specific for a data stream. This is a user-defined alternate name to associate with the rate column and is used in the link manager and telemetry scheduler for the tab name (if no data stream name is entered the rate column name is used instead). The script access methods use the data stream name.
<b>Maximum messages per cycle</b>	This value is specific for a data stream. This value is 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 <b>Maximum messages per second</b> value.
<b>Maximum bytes per second</b>	This value is specific for a data stream. This is the maximum number of bytes that can be downlinked during a single second.

The **Available rates** field displays the rates that are available based on the input values.

The evenly time-spaced sub-second rates are calculated using the above values. For example, given a maximum 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).

#### 4.9.4.5 App parameters

The **App Parameters** command displays the Application Parameters dialog (Figure 78).

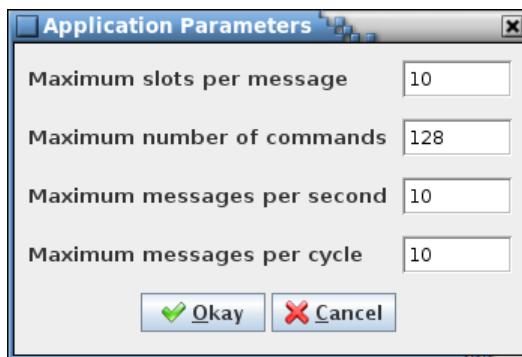


Figure 78. Application Parameters dialog

This dialog is used to set parameters for the application scheduler table. The values for **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:

<b>Maximum slots per message</b>	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
----------------------------------	--

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	<i>Version 1.2.31</i>
	Date: May 2018	Page 120 of 295

a time slot to have more applications assigned to it than this parameter.

<b>Maximum number of commands</b>	The maximum number of commands that can be created for the scheduler table.
<b>Maximum messages per second</b>	The maximum number of time slots that are available for an application to be scheduled in a second.
<b>Maximum messages per cycle</b>	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 <b>Maximum messages per second</b> value.

#### 4.9.5 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 project 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, Groovy, and Scala. Example scripts are provided with the application. These can be modified, or new scripts written as needed by the user. See paragraph 4.10 for more information on the use of scripts to access the table data.

The Manage Script Associations (see paragraph 4.9.5.1) and Execute Script(s) (see paragraph 4.9.5.2) dialogs are mutually exclusive; opening one causes the other, if displayed, to be closed.

##### 4.9.5.1 Manage

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

When the command is selected the Manage Script Associations dialog (Figure 79) is displayed. If the Execute Script(s) dialog (see paragraph 4.9.5.2) is currently displayed then it is closed. The dialog is divided into four sections: script selection, table selection, script associations, and command buttons. The line separating the table selection tree and script associations list delineates a split pane control that is used to resize these panels relative to one another. Position the mouse pointer over the separator line and when the pointer changes to a double-headed arrow press and hold the left mouse button. Space permitting, the adjoining panes can be resized by moving the mouse pointer up or down. Release the mouse button to exit resizing.

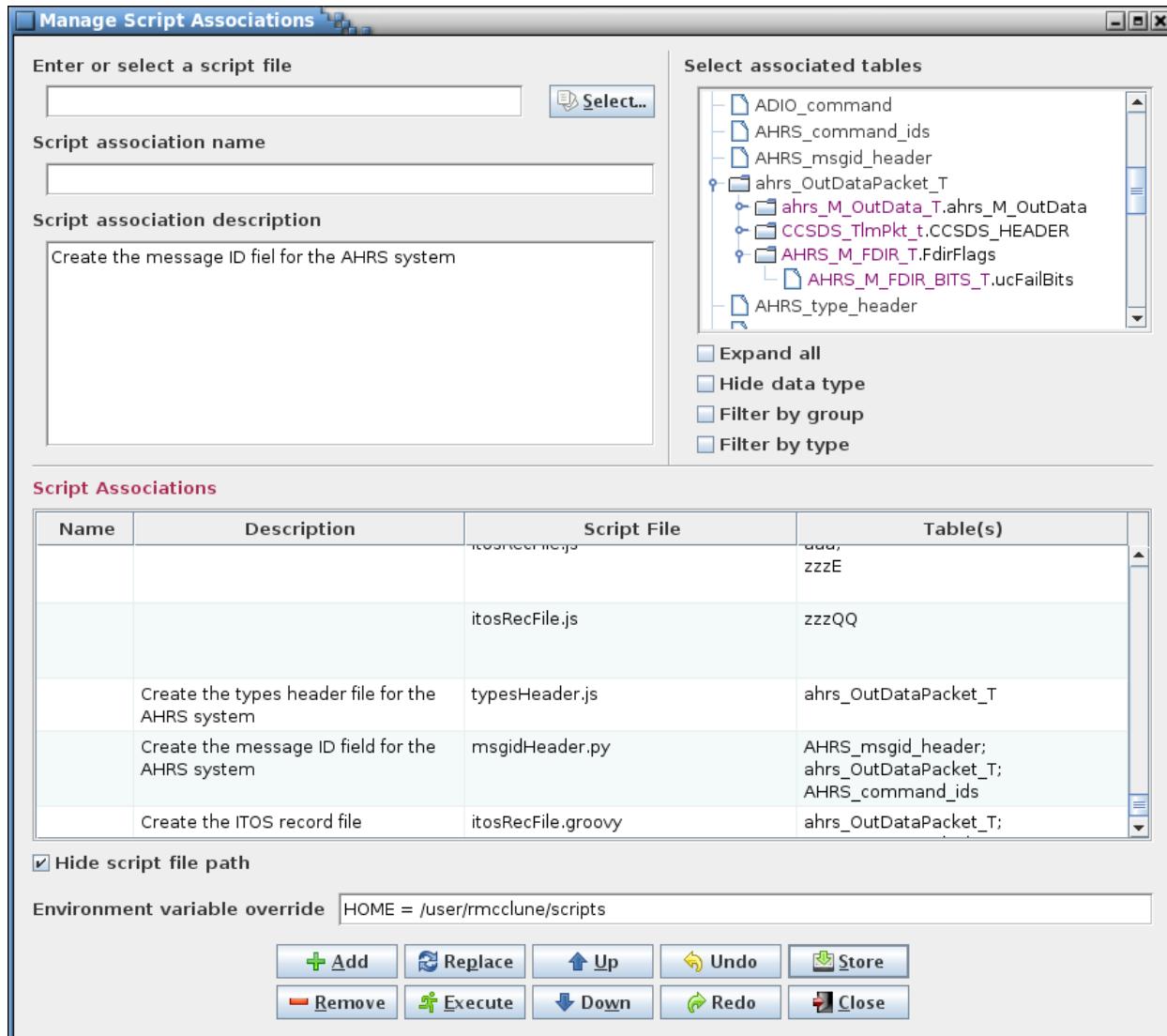


Figure 79. Manage Script Associations dialog

The script association name field allows a name to be assigned to an association. The name may not match one already in use by another association and must adhere to the alphanumeric input type (see paragraph 4.7). The association name is optional; it's purpose is to be used with the command line execute option (see paragraph 4.1) to reference an existing association, in place of typing the script name and table(s).

The script association description field allows a description to be added to an association. The description may remain blank.

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 root tables and their child tables (if applicable). The user expands the tree branches and selects one or more tables (see paragraph 4.5.3 for more information on table trees). When a structure table is chosen all of its child tables are automatically included when the script

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018		Page 122 of 295

association is executed; therefore the child tables do not have to be explicitly selected when creating the association, and any child nodes that are selected are eliminated from the **Script Associations** table's **Table(s)** column when the association is added. It makes no difference in what order the tables are selected for assignment since, when loaded for use by the script, the tables are sorted so that the root tables are in alphabetical order and the child tables appear in the order defined by their table type definition.

If the table tree is filtered by group then one or more group names can be selected from the tree for association with the chosen script. When the association is added to the table the group name (prepended with "Group:") appears in the **Table(s)** column. Any of the group's member tables that were also selected don't appear in the table, however. When the script association is executed the group's current member tables are loaded. This allows the script association to remain unchanged even if tables are added or removed from the group.

The **Script Associations** table displays the script associations that are stored in the project database, plus any that have been added while this dialog is open. 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. The **Description** column may be edited to add, alter, or remove an association's description. The **Script File** column displays the association's script file and file path. Below the table is a check box that allows toggling between hiding and displaying the script file paths; this selection does not affect storing and retrieving the file paths and the check box's selection state is remembered between sessions. The **Table(s)** column display each table associated with the script in the **Script File** column. Each table is displayed on a single line (space permitting) and includes its full path.

Below the table is a check box, **Hide script file path**. When selected the file paths in the association table's **Script File** column are not displayed. Deselecting the check box restores the paths. The paths are used when executing the associations even if not displayed.

Script file paths are allowed to have environment variables within them. When the association is executed any variables in the script path are replaced by their corresponding value in the system environment variable map. The **Environment variable override** field allows the variable values in the system map to be temporarily replaced, or added if the variable doesn't exist in the map. The format for entries in this field is:

```
<key1 = value1<,key2 = value2<, ...>>>
```

Each key/value pair must be separated by a comma. A dollar sign (\$) can precede the key name, and spaces (except those embedded within a key or value) and double quotes bounding the keys and values are ignored. The contents of this field are stored as a program preference, so the field is automatically populated with the overrides that were last used.

Script associations may be executed from within this dialog. This is similar to execution of the associations from the Execute Scripts dialog (see paragraph 4.9.5.2) except that in this dialog the associations do not have to be stored in the project database to be executed. This provides a means to create a one-use association for immediate execution. See paragraph 4.9.5.2 for further detail on script execution.

The button commands are described below:

- |            |   |
|------------|---|
| <b>Add</b> | After entering a name (optional), description (optional), choosing a script file, and (if needed by the script) one or more data tables or groups, selecting the <b>Add</b> button creates the script association. The new association is inserted as the first row in the <b>Script Associations</b> table (any existing associations are moved down a row) and is automatically selected. A script may be used in more than one association, and a data |
|------------|---|

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Date: May 2018

table or group may be used in any number of associations; however, duplicate associations, i.e., those utilizing the same script, table(s), and group(s), are not added to the list.

A script can be added without associating it with a table or group. This is the case when the script performs actions that do not need data from a specific data table (for an example see the script that creates the housekeeping (HK) copy table).

<b>Remove</b>	An association may deleted from the table by selecting it using the mouse or keyboard, then pressing the <b>Remove</b> button. Multiple associations may be removed simultaneously by selecting more than one from the list by using the Ctrl or Shift keys.
<b>Replace</b>	Replace the currently selected association in the <b>Script Associations</b> table with the association defined by the currently entered script name, description, script file, and selected table(s).
<b>Execute</b>	Execute the script associations(s) that are selected in the script associations table (disabled associations are ignored). The application GUI is disabled during script execution in order to prevent possible alteration of the data while a script is accessing it.
<b>Up</b>	Move the currently selected row(s) up one row relative to the remaining rows. The order of the script associations in the table affects the order in which the associations are executed if multiple associations are chosen to execute. Otherwise the capability to arrange the rows is solely for the user to group the associations as desired. The ordering is preserved when the associations are stored and retrieved from the project database, and can be useful for keeping affiliated associations near one another.
<b>Down</b>	Move the row(s) of the currently selected cell(s) down one row relative to the remaining rows. The order of the script associations in the table affects the order in which the associations are executed if multiple associations are chosen to execute. Otherwise the capability to arrange the rows is solely for the user to group the associations as desired. The ordering is preserved when the associations are stored and retrieved from the project database, and can be useful for keeping affiliated associations near one another.
<b>Undo</b>	Undoes the last action performed (add, remove, move up, move down, redo).
<b>Redo</b>	Reverses the last action undone (add, remove, move up, move down, undo).
<b>Store</b>	Stores the changes made to script associations in the script association manager into the project database. The order of the associations in the table is preserved. If changes have been made a confirmation dialog first appears. Select <b>Okay</b> to store the updates; select <b>Cancel</b> to exit the confirmation dialog without altering the database.
<b>Close</b>	Closes the script association manager window. If there are any unsaved association changes a dialog appears requesting confirmation to discard the changes. Select <b>Okay</b> to exit script associations dialog, losing any unsaved changes. Select <b>Cancel</b> to return to the Manage Script Associations dialog.

#### 4.9.5.2 Execute

Selecting the **Execute** command causes the Execute Script(s) dialog to appear (Figure 80). If the Manage Script Associations dialog (see paragraph 4.9.5.1) is currently open then it is closed unless there are unsaved changes; in this case a dialog appears requesting confirmation to discard the changes. If the user doesn't elect to discard the changes then the Manage Script Associations dialog remains open and the Execute Script(s) dialog doesn't appear.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
Doc. No. JSC-37494	Version 1.2.31	
Date: May 2018	Page 124 of 295	

The dialog displays a table of the stored script associations. Associations are grayed out if the script file script file can't be found in the folder indicated by the path or an associated data table doesn't exist in the project database. Grayed-out associations can't be selected for execution. Below the table is a check box that allows toggling between hiding and displaying the script file paths in the **Script File** column. See paragraph 4.9.5.1 for more information on the script associations table.

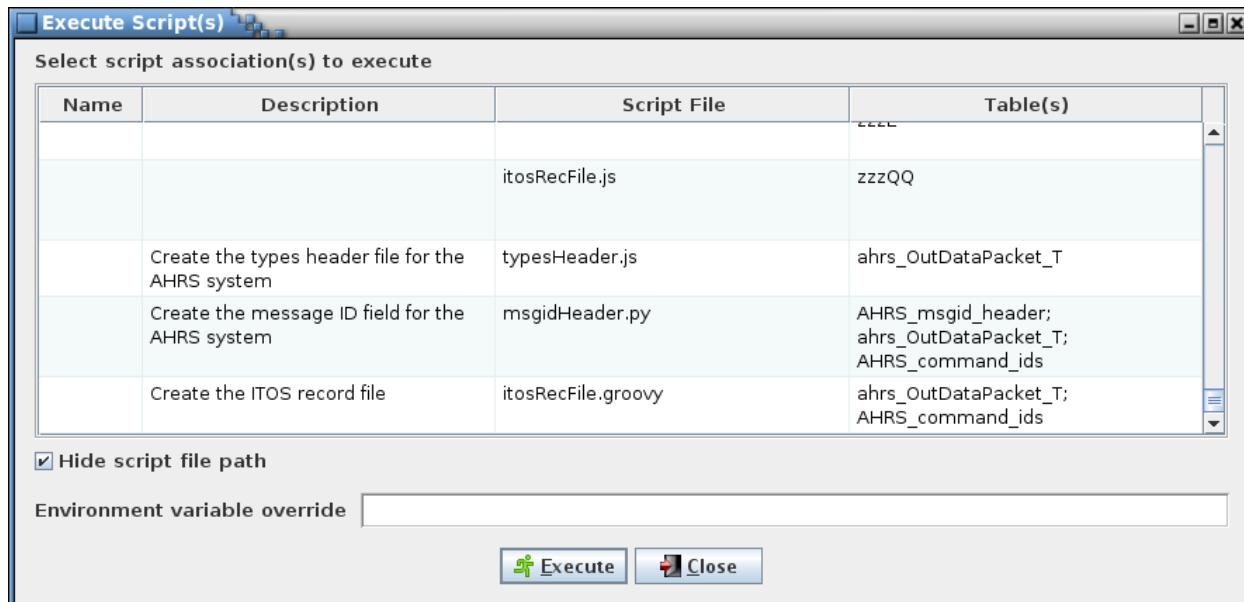


Figure 80. Execute Script(s) dialog

Select one or more associations from the table and press the **Execute** button to execute the selected association(s). The application GUI is disabled during script execution in order to prevent possible alteration of the data while a script is accessing it. While the script is executing the dialog shown in Figure 81 is displayed. Pressing the **Halt** button stops script execution immediately and returns control to the CCDD application – this is necessary, for instance, if the script contains an infinite loop.



Figure 81. Halt script execution dialog

When script execution completes a status message is written to the event log. If script execution is halted the log entry indicates that the scripts failed to execute; however, if multiple scripts are executed it's possible that one or more successfully completed prior to the action to halt execution. If an error occurs, preventing successful script completion, an error dialog appears indicating that the script(s) failed to execute. The log entry in this case displays the script name(s) and provides details on the cause of the error.

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

#### 4.9.5.3 Store

The **Store** command is used to store scripts in the project database. This provides 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 82).

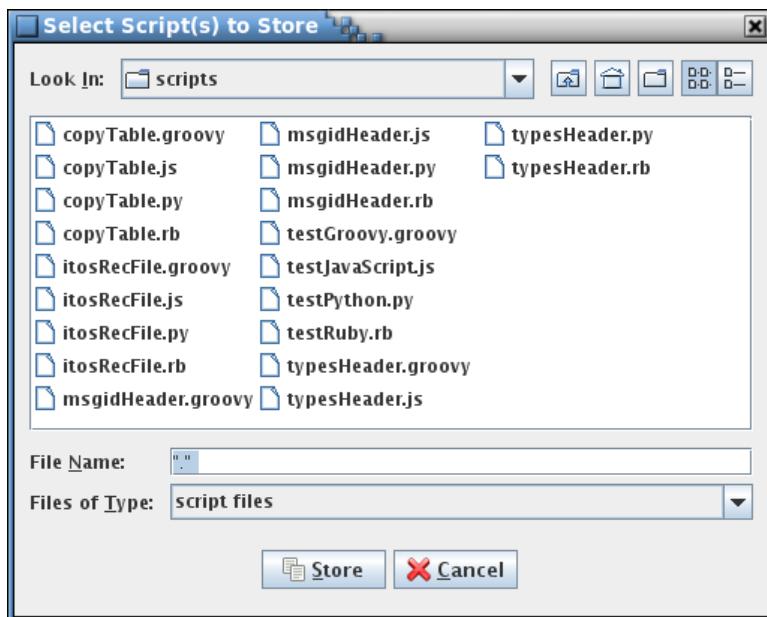


Figure 82. Script selection dialog

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 database table. Select the **Cancel** button to exit the dialog without storing any files. The **Retrieve** command (see paragraph 4.9.5.4) provides the means for retrieving the stored scripts 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 83). 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 project database, not only script files.

#### 4.9.5.4 Retrieve

Selecting the **Retrieve** command causes the Retrieve Script(s) dialog to appear (Figure 83). This command allows for extracting scripts (or other text files) that are stored in the project database using the **Store** command (see paragraph 4.9.5.3).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide
	Doc. No. JSC-37494
	Version 1.2.31

Date: May 2018

Page 126 of 295

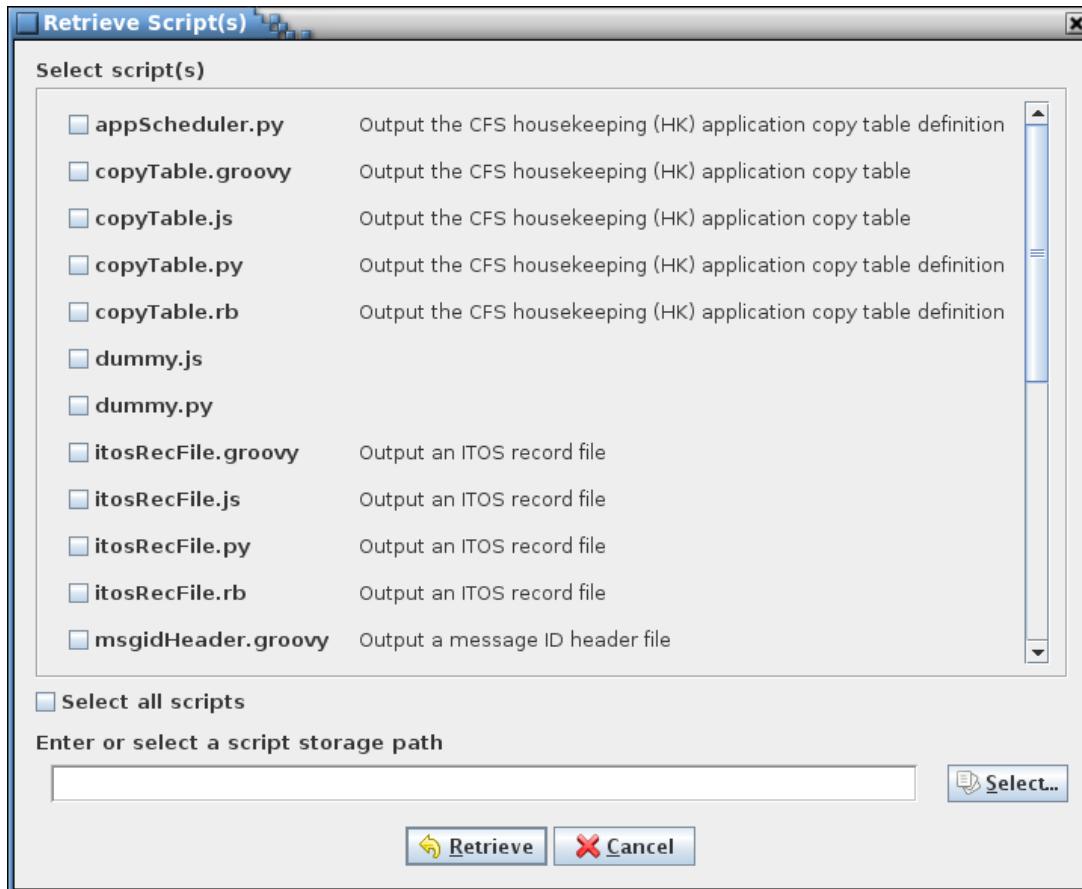


Figure 83. Retrieve Script(s) dialog

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.

#### 4.9.5.5 Delete

Selecting the **Delete** command causes the Delete Script(s) dialog to appear (Figure 84). This command allows for deleting script (or other text files) stored in the project database using the **Store** command (see paragraph 4.9.5.3).

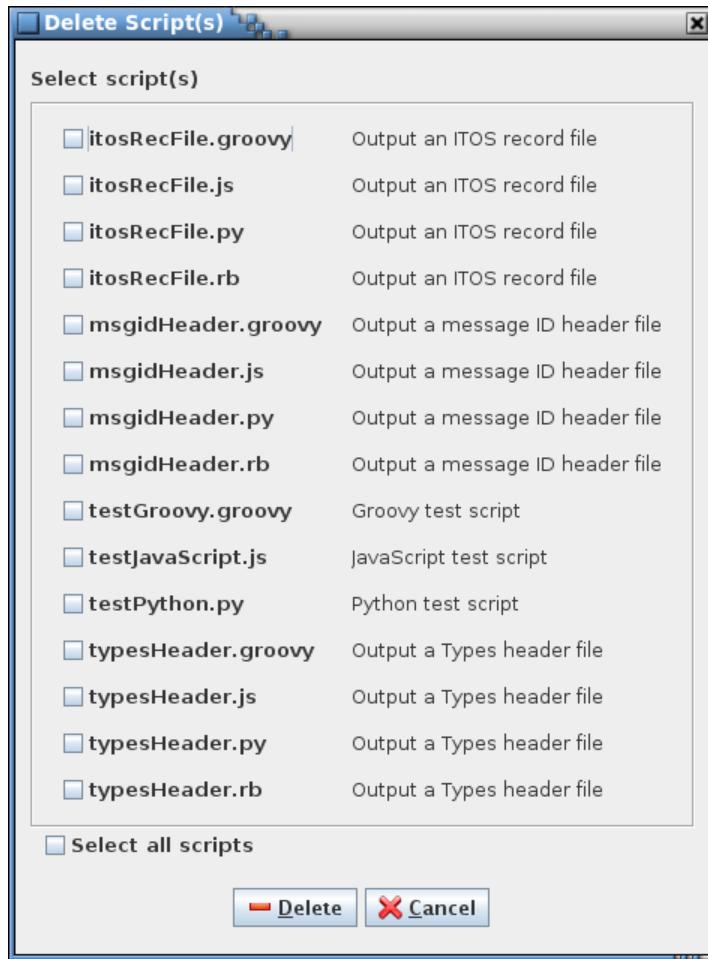


Figure 84. Delete Script(s) dialog

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. A confirmation dialog first appears; select **Okay** to continue with the script deletion, or **Cancel** to exit the dialog without deleting a script. Select the **Cancel** button to exit the dialog without deleting any files.

#### 4.9.5.6 Search

The script **Search** dialog provides a means of searching for a specified text string within the scripts stored in the project database (see Figure 85). Case sensitivity for the search is governed by the **Ignore text case** check box.

The **Allow regular expression** check box, when checked, allows the use of a regular expression to define the search pattern in the search text field. A regular expression can be formulated to find multiple matching conditions (for example, the search for **a.c** would match any string that has a single character between the characters 'a' and 'c'). Information regarding the use of regular expressions is beyond the scope of this document; however, resources and tutorials can be found online. When unchecked, the search text is matched as typed in the search text field.

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, **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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 128 of 295

highlighted. Leading and trailing white space characters are removed from the context cells' text prior to display.

The search text field uses auto-completion to fill in the search string. The previous search strings (those for the event log, table, and script) are remembered, including those from previous sessions. The number of remembered search strings can be changed via the Preferences dialog, and defaults to 30. Case sensitivity for auto-completion is based on the **Ignore text case** check box selection state.

Another search can be performed by altering the search text and selecting the **Search** button again. 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.

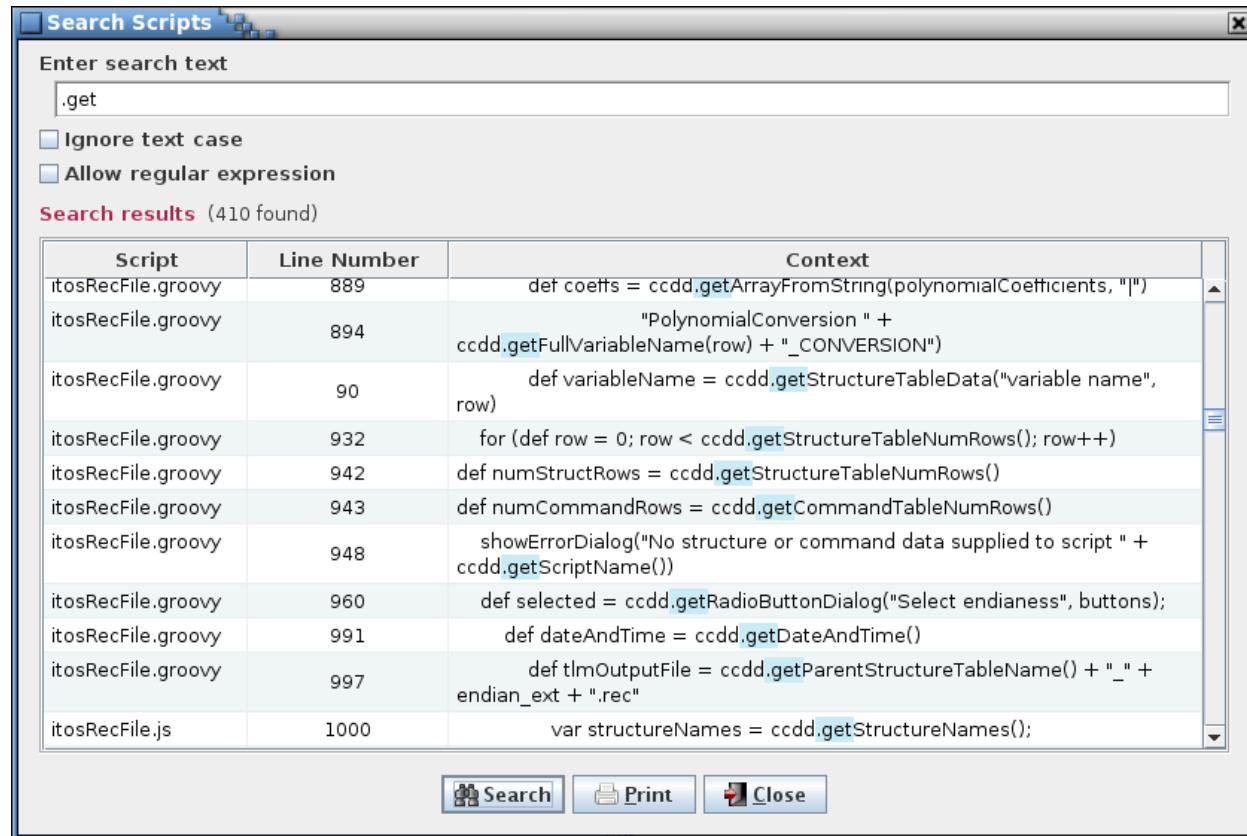


Figure 85. Script search dialog

## 4.9.6 Help menu

### 4.9.6.1 Guide

The **Guide** command displays a copy of this user's guide in PDF format. The file `CCDD_Users_guide.pdf` must be in the same folder in which the `CCDD.jar` file is located in order for the user's guide to be opened using this command.

### 4.9.6.2 About

Selecting the **About** menu item brings up an informational dialog (see Figure 86) providing the application's version number and date, version numbers for Java, PostgreSQL, JDBC, and Jetty that are in use, and the copyright notice. Also displayed is the list of available scripting languages and associated scripting engines, if any, and their respective version information.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 129 of 295

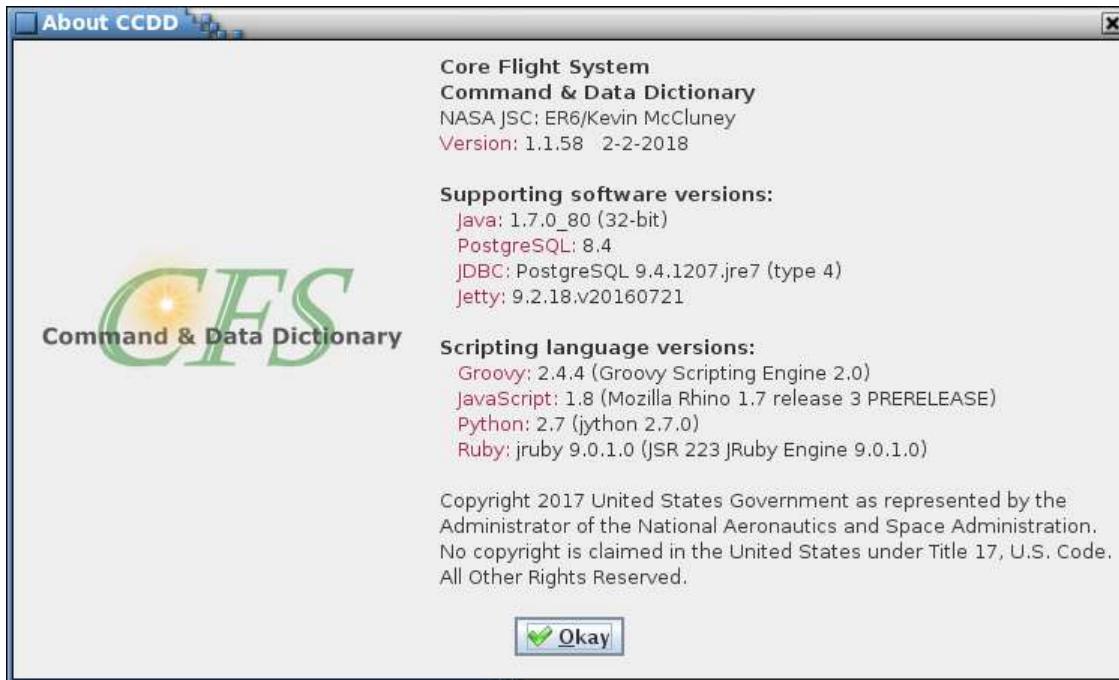


Figure 86. About dialog

## 4.10 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. Five of these languages, JavaScript, Python, Ruby, Groovy, and Scala, have been tested with the CCDD application, though any of the other compliant scripting languages should work as well. A language must be installed before it can be used by CCDD; paragraph 4.1 provides details on the library files required for using each of the five tested scripting languages. The About dialog (see paragraph 4.9.6.2 and Figure 86) 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 4.9.5.1 and 4.9.5.2) or from the command line (see Table 1 and paragraph 4.10.5).

The scripts have access to the project data via a set of script data access methods written in Java. Additional methods are provided for displaying dialog boxes (both output and input), opening and writing to an output file, and making direct queries to the database. The methods are called from within a script using the method name and, dependent on the language, prepended by the class name **ccdd** or **ccdds**:

```
<ccdd or ccdds.>methodName(arguments...)
```

where *methodName* is the name of the script data access method (function) and *arguments...* are the parameters required by the particular method. **ccdd** is a reference to the non-static version of the script data access class, whereas **ccdds** is a static reference to the non-static class' methods. JavaScript scripts require the non-static reference in Java 8, but can use either in Java 7. Ruby scripts require the non-static reference, but Python and Groovy scripts can use either. Scala scripts must use the static

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
<b>Version 1.2.31</b>		Page 130 of 295

reference; however, the class name is not used in the script. Details on the script data access methods are provided in Table 9.

In order to access these methods the script requires that the data access class (non-static or static version) be imported; the import statement format 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 4.9.5.1 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.

#### 4.10.1 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);

```

The following is the example script described earlier in this section, written in JavaScript:

```

// 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.getStructureTableNames();

// 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 Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. JSC-37494 Date: May 2018		
	<i>Version 1.2.31</i>	Page 131 of 295	

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

#### 4.10.2 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
```

The following is the example script described earlier in this section, written in Python:

```
# 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.getStructureTableNames()

# 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)
```

#### 4.10.3 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
```

The following is the example script described earlier in this section, written in Ruby:

```
# 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.getStructureTableNames()

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}")

    index += 1

end
```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
	<i>Version 1.2.31</i>	Page 132 of 295

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

#### 4.10.4 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

// 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.getStructureTableNames()

// 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)
```

#### 4.10.5 Scala

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

```
import CCDD.CcddScriptDataAccessHandlerStatic._
```

The following is the example script described earlier in this section, written in Scala:

```
// Import the script data access method class
import CCDD.CcddScriptDataAccessHandlerStatic._

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

// Get the array of structure names
var structNames = getStructureTableNames()

// Step through each name found
for (index <- 0 to structNames.length - 1)
{
    // Write the structure name to the output file
    writeToFileLn(file,
                  "structNames[" +
                  index +
                  "] = " +
                  structNames(index))
```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
	Version 1.2.31 Page 133 of 295	

```

}

// Close the output file
closeFile(file)

```

#### 4.10.6 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]]][;...]]>
```

Groups can be used in place of, or along with tables. Each referenced group name must be preceded by 'Group:' in order to be recognized as a group. For example:

```
script_name:Group:group_name
```

The project database, host, user, and password (if required) command line options must be specified as part of 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 in the same `execute` command then the individual associations must be separated by a semi-colon (;) and the entire string containing the associations for that `execute` command must be bounded by single or double quotes. Multiple `execute` commands in the same command line command can be used as well to execute multiple script associations; the format for each is as described above. If multiple script associations are specified then these are run serially in the order they appear in the command line command.

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 (warning and error) streams. Dialogs within a script 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
```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	<i>Version 1.2.31</i>
	Date: May 2018	Page 134 of 295

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
```

#### 4.10.7 XTCE export

The XTCE export dialog (paragraph 4.9.3.8.4) provides a means of exporting table data into a file in XTCE format. The XTCE schema allows for some interpretation as to how data is parsed to the output and provides for many features not covered by the export methods within the CCDD application, so the content of the CCDD-produced XTCE file may not be as desired. A script can be created to produce the conversion; however, there are certain operations performed in the internal code that would be difficult to replicate in a script - for example, the conversion of the command header structure table to the equivalent command metadata required by the XTCE schema. To make XTCE conversion via a script easier a hybrid of the internal code and script code can be used.

The internal code has “hooks” that allow for several of the key conversion methods to be replaced with code in a script. These hooks are activated when exporting via the XTCE export dialog by selecting the **Use external methods** check box, or when the script access method **xtceExport** is called from a script.

Table 8 below provides a brief description of each of the XTCE export methods that can be overridden by a script function. The script must use the same function name as shown in the table, and the **Input(s)** and **Output** columns describe the parameters that are passed to and expected from the script functions. Figure 87 displays a flow chart of the methods’ calling sequence. If the script doesn’t have the function to replace a method then the internal method is used by default.

The user can elect to arrange the conversion process in a manner other than by the predefined methods. Assuming the **xtceExport()** script data access method or the export dialog is used as the conversion initiator, the **addSpaceSystemParameters** and **addSpaceSystemCommands** must be the entry points in the script.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 135 of 295	

Internal Method / Script Function Name	Description	Input(s)	Output
addSpaceSystemHeader	Add the Header element to the space system. For the root space system the AuthorSet and NoteSet elements are automatically created and populated with the user, project, creation, and endianness information. The script data access method <b>xtceAddSpaceSystemHeader</b> can be called instead to use the internal method.	ObjectFactory: Object factory reference SpaceSystemType: Space system reference String: Classification attribute String: Validation status attribute String: Version attribute String: Creation time and date	
createTelemetryMetadata	Create the TelemetryMetaElement element within the specified space system. The script data access method <b>xtceCreateTelemetryMetadata</b> can be called instead to use the internal method.	ObjectFactory: Object factory reference SpaceSystemType: Space system reference	
addSpaceSystemParameters	Overall parameter assignment handler; steps through each row of data in the structure table and calls functions that assign variables to the ParameterTypeSet, ParameterSet, and ContainerSet elements. If the application ID is provided, the table is a root structure, and it has a variable with a data type referencing the telemetry header table, then a BaseContainer element is created with a RestrictionCriteria element set to the application ID. The script data access method <b>xtceAddSpaceSystemParameters</b> can be called instead to use the internal method.	JAXBElement<SpaceSystemType>: Top-level space system element reference ObjectFactory: Object factory reference boolean: true if the data is big endian; false for little endian boolean: true if the telemetry and command headers are big endian String: Telemetry header table name SpaceSystemType: Space system to which the table belongs String: Table name String[][]: Array containing the table's data int: Variable (parameter) name column index int: Parameter data type column index int: Parameter array size column index int: Parameter bit length column index	

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 136 of 295	

Internal Method / Script Function Name	Description	Input(s)	Output
		int: Parameter enumeration column index; -1 if no the table has no enumeration column int: Parameter description column index; -1 if no the table has no description column int: Parameter units column index; -1 if no the table has no units column int: Minimum parameter value column index; -1 if no the table has no minimum column int: Maximum parameter value column index; -1 if no the table has no maximum column boolean: true if this table represents the telemetry header or one of its descendants String: telemetry header table system path; null or blank is none boolean: true if the table is a root structure table String: Name of the telemetry header application ID data field String: Telemetry header application ID	
addParameterAndType	Add a structure table variable, if it has a primitive data type, to the ParameterTypeSet and ParameterSet elements. The script data access method <b>xtceAddParameterAndType</b> can be called instead to use the internal method.	ObjectFactory: Object factory reference boolean: true if the data is big endian; false for little endian	

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 137 of 295	

Internal Method / Script Function Name	Description	Input(s)	Output
		boolean: true if the telemetry and command headers are big endian String: Telemetry header table name SpaceSystemType: Space system reference String: Parameter name String: Parameter primitive data type String: Parameter array size; null or blank if the parameter isn't an array String: Parameter bit length; null or blank if not a bit-wise parameter String: Enumeration in the format <enum label> <enum value>[ ...][,...]; null to not specify String: Parameter units String: Minimum parameter value String: Maximum parameter value String: Parameter description int: Size, in characters, of a string parameter; ignored if not a string or character	
setParameterDataType	Add a structure table variable as a parameter type in the ParameterTypeSet element. The parameter type is based on the variable's data type. An array variable generates two entries, one to describe the array type, and a second to describe the data type of the array elements. The script data access method <b>xtceSetParameterDataType</b> can be called instead to use the internal method.	ObjectFactory: Object factory reference boolean: true if the data is big endian; false for little endian boolean: true if the telemetry and command headers are big endian String: Telemetry header table name SpaceSystemType: Space system	

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 138 of 295	

Internal Method / Script Function Name	Description	Input(s)	Output
		String: Parameter name; null to not specify String: Data type; null to not specify String: Parameter array size; null or blank if the parameter isn't an array String: Parameter bit length; null or empty if not a bit-wise parameter String: Enumeration in the format <enum label> <enum value>[ ...][,...]; null to not specify String: Parameter units; null to not specify String: Minimum parameter value; null to not specify String: Maximum parameter value; null to not specify String: Parameter description; null to not specify int: Size, in characters, of a string parameter; ignored if not a string or character	
addParameterSequenceEntry	Add the structure table variables as EntryList entries in a SequenceContainer element within a ContainerSet element. The script data access method <b>xtceAddParameterSequenceEntry</b> can be called instead to use the internal method.	ObjectFactory: Object factory reference String: Telemetry header table name SpaceSystemType: Reference to the space system to which the parameter belongs String: Parameter name String: Data type String: Array size	boolean: true if the parameter's data type references the telemetry header or one of its descendants; otherwise return the flag status unchanged

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 139 of 295	

Internal Method / Script Function Name	Description	Input(s)	Output
		EntryListType: Reference to the entry list into which to place the parameter (for a primitive data type) or container (for a structure data type) reference  boolean: true if this table represents the telemetry header or one of its descendants	
createCommandMetadata	Create the CommandMetaData element within the specified space system. The script data access method <b>xtceCreateCommandMetadata</b> can be called instead to use the internal method.	ObjectFactory: Object factory reference  SpaceSystemType: Space reference	
addSpaceSystemCommands	Overall command assignment handler; steps through each row of data in the command table, parses the command's arguments, and calls functions that assign arguments to the ArgumentTypeSet element. For each command in the table a function is called to create the MetaCommand element, which describes the command, within the MetaCommandSet. The script data access method <b>xtceAddSpaceSystemCommands</b> can be called instead to use the internal method.	JAXBElement<SpaceSystemType>: Top-level space system element reference  ObjectFactory: Object factory reference  boolean: true if the data is big endian; false for little endian  boolean: true if the telemetry and command headers are big endian  String: Command header table name  AssociatedColumns[]: Array of AssociatedColumns class instances that have the associated command argument column indices  SpaceSystemType: Space system reference  String[][]: Table data array  int: Command name column index  int: Command code column index  int: Command description column index	

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 140 of 295	

Internal Method / Script Function Name	Description	Input(s)	Output
		boolean: true if this table represents the command header String: Command header table system path String: Command code name String: Name of the application ID String: Application ID	
addCommand	Add a MetaCommand element to the MetaCommandSet. Each MetaCommand element has a BaseMetaCommand (if the command header table is defined), which uses ArgumentAssignment elements to set the application ID and command function code. The ArgumentList and CommandContainer are populated with the command argument information. The script data access method <b>xtceAddCommand</b> can be called instead to use the internal method.	JAXBElement<SpaceSystemType>: Top-level space system element reference ObjectFactory: Object factory reference String: Command header table name SpaceSystemType: Space system reference String: Command name String: Command code name String: Command code String: Name of the application ID String: Application ID boolean: true if this table represents the command header String: Command header table system path String[]: Array of command argument names String[]: Array of of command argument data types String[]: Array of of command argument array sizes; the array item is null or	

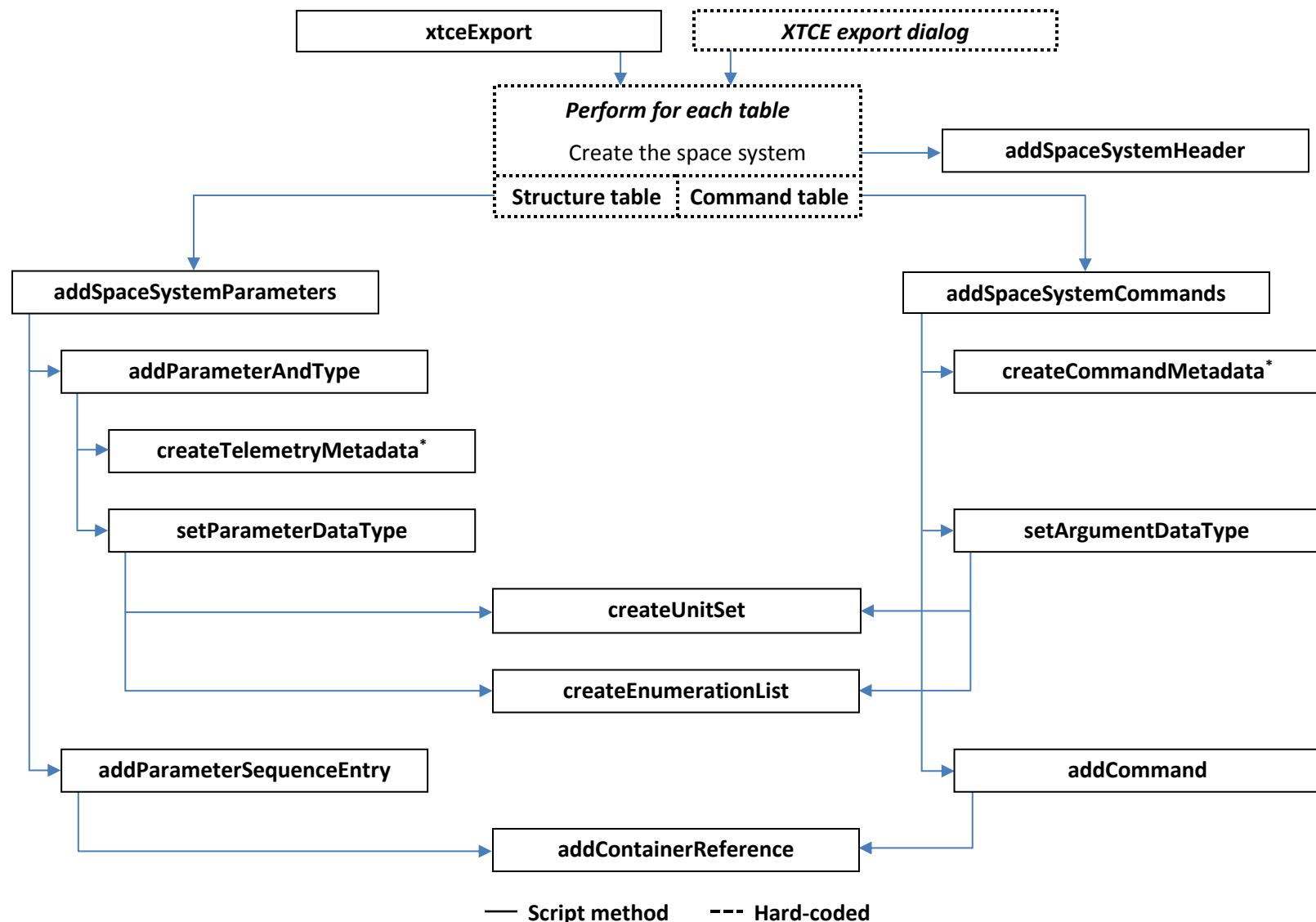
Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 141 of 295	

Internal Method / Script Function Name	Description	Input(s)	Output
		blank if the corresponding argument isn't an array  String: Description of the command	
setArgumentDataType	Add a command table command argument as an argument type in the ArgumentTypeSet element.  The argument type is based on the argument's data type. An array variable generates two entries, one to describe the array type, and a second to describe the data type of the array elements. The script data access method <b>xtceSetArgumentDataType</b> can be called instead to use the internal method.	boolean: true if the data is big endian; false for little endian  boolean: true if the telemetry and command headers are big endian  String: Command header table name  SpaceSystemType: Space system reference  String: Command argument name; null to not specify  String: Command argument data type; null to not specify  String: Command argument array size; null or blank if the argument isn't an array  String: Command argument bit length  String: Command argument enumeration in the format <enum label> <enum value>[ ...][,...]; null to not specify  String: Command argument units; null to not specify  String: Minimum parameter value; null to not specify  String: Maximum parameter value; null to not specify  String: Command argument description ; null to not specify	NameDescriptionType:  Command description of the type corresponding to the primitive data type with the specified attributes set

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 142 of 295	

Internal Method / Script Function Name	Description	Input(s)	Output
		int: String size in bytes; ignored if the command argument does not have a string data type	
addContainerReference	Generic function used by both the telemetry and command functions to add a container reference to the specified EntryList element. The script data access method <b>xtceAddContainerReference</b> can be called instead to use the internal method.	ObjectFactory: object factory reference EntryListType: Reference to the telemetry or command entry list into which to place the parameter or parameter array container reference(s) String: Parameter name String: Data type String: Parameter array size; null or blank if the parameter isn't an array	
createUnitSet	Use the provided units information to create a UnitSet element. The script data access method <b>xtceCreateUnitSet</b> can be called instead to use the internal method.	ObjectFactory: object factory reference String: Parameter or command argument units; null to not specify	UnitSet: Unit set for the supplied units string; an empty unit set if no units are supplied
createEnumerationList	Use the provided enumeration information to create an EnumerationList element. The script data access method <b>xtceCreateEnumerationList</b> can be called instead to use the internal method.	ObjectFactory: object factory reference SpaceSystemType: Space system reference String: Enumeration in the format <enum value><enum value separator><enum label>[<enum value separator>...][<enum pair separator>...]	EnumerationList: Enumeration list for the supplied enumeration string

Table 8. Overridable XTCE export methods



\* Called once per space system, and only if a telemetry/command parameter exists in the table

Figure 87. XTCE export script method flow

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 144 of 295

#### 4.10.8 Data access methods

Table 9 provides details on each of the project data access methods available for use in the scripts. The first column is the method name. The scripts are automatically assigned a variable, `ccdd`, which references the class containing the data access methods. When calling one of the access methods from a script the method name must be preceded by `ccdd` for JavaScript, Python, and Groovy, and `$ccdd` for Ruby; for Scala only the method name is used. 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 `getCommandTableData` and `getStructureTableData`.

Method Name	Description	Input(s)	Output
closeFile	Close the specified output file	PrintWriter: Output file PrintWriter object obtained from the openOutputFile method	
formatArrayIndex	Convert an integer array containing the size of each array dimension into a string in the format [#]<[#]<...>>	int[]: Array of integers containing the size of each array dimension	String: Array size in the format [#]<[#]<...>>
getApplicationMessageDefinitionTable	Get the application scheduler message definition table		String[]: Array containing the message definition table information
getApplicationNames	Get the array containing the groups that represent CFS applications		String[]: Array containing names of the groups that represent CFS applications
getApplicationScheduleDefinitionTable	Get the specified entry in the application scheduler schedule definition table	int: Row index for the entry in the schedule definition table	String[][]: Array containing the specified entry in the schedule definition table
getApplicationScheduleDefinitionTableDefines	Get the array of defined parameters for the schedule definition table		String[]: Two-dimensional array containing the defined parameters

Method Name	Description	Input(s)	Output
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  String: Character string to use to delineate the separation point(s) between columns. The separator is eliminated from the array members  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
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  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
getArrayIndexFromSize	Get the integer array containing the size of each array dimension from the supplied array size string	String: Array size in the format [#]<[#]<...>> or #<,#<...>>	int[]: Array of integers containing the size of each array dimension

Method Name	Description	Input(s)	Output
getAssociatedGroupNames	Get the array of group names referenced in the script association		String[]: Array containing the group names referenced in the script association; empty array if no groups are referenced
getBaseDataType	Get the base type for the specified data type	String: Primitive data type	String: Base type for the specified data type; returns null if the data type doesn't exist or isn't a primitive type
getCDATAType	Get the C type for the specified data type	String: Primitive data type	String: C type for the specified data type; returns null if the data type doesn't exist or isn't a primitive type
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) 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
getCommandArgArraySize	Get the argument array size for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument array size for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value; null if the argument number or row index is invalid

Method Name	Description	Input(s)	Output
getCommandArgArraySizeWithMacros	Get the argument array size for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument array size for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place; null if the argument number or row index is invalid
getCommandArgBitLength	Get the argument bit length for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument bit length for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value; null if the argument number or row index is invalid
getCommandArgBitLengthWithMacros	Get the argument bit length for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument bit length for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place; null if the argument number or row index is invalid
getCommandArgByColumnName	Get the argument value (as a string) for the column belonging to the specified command argument at the specified row in the command data, with any macro name replaced by its corresponding value	int: Command argument index. The first argument is 0 int: Table data row index String: Name of the argument column for which the value is requested	String: Argument value (as a string) for the column belonging to the specified command argument at the specified row in the command data; null if the argument number, row index, or column name is invalid

Method Name	Description	Input(s)	Output
getCommandArgByColumnNameWithMacros	Get the argument value (as a string) for the column belonging to the specified command argument at the specified row in the command data, with any embedded macro(s) left in place	int: Command argument index. The first argument is 0  int: Table data row index  String: Name of the argument column for which the value is requested	String: Argument value (as a string) for the column belonging to the specified command argument at the specified row in the command data, with any embedded macro(s) left in place; null if the argument number, row index, or column name is invalid
getCommandArgColumnNames	Get the array of column names belonging to the specified command argument at the specified row in the command data	int: Command argument index. The first argument is 0  int: Table data row index	String[]: Array of column names belonging to the specified command argument at the specified row in the command data; null if the argument number or row index is invalid
getCommandArgDataType	Get the argument data type for the specified command argument at the specified row in the command data	int: Command argument index. The first argument is 0  int: Table data row index	String: Argument data type for the specified command argument at the specified row in the command data; null if the argument number or row index is invalid
getCommandArgEnumeration	Get the argument enumeration for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value	int: Command argument index. The first argument is 0  int: Table data row index	String: Argument enumeration for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value; null if the argument number or row index is invalid

Method Name	Description	Input(s)	Output
getCommandArgEnumerationWithMacros	Get the argument enumeration for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument enumeration for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place; null if the argument number or row index is invalid
getCommandArgMaximum	Get the argument maximum (as a string) for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument maximum value (as a string) for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value; null if the argument number or row index is invalid
getCommandArgMaximumWithMacros	Get the argument maximum (as a string) for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument maximum value (as a string) for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place; null if the argument number or row index is invalid

Method Name	Description	Input(s)	Output
getCommandArgMinimum	Get the argument minimum value (as a string) for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument minimum value (as a string) for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value; null if the argument number or row index is invalid
getCommandArgMinimumWithMacros	Get the argument minimum value (as a string) for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument minimum value (as a string) for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place; null if the argument number or row index is invalid
getCommandArgName	Get the argument name for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument name for the specified command argument at the specified row in the command data, with any macro replaced by its corresponding value; null if the argument number or row index is invalid
getCommandArgNameWithMacros	Get the argument name for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place	int: Command argument index. The first argument is 0 int: Table data row index	String: Argument name for the specified command argument at the specified row in the command data, with any embedded macro(s) left in place; null if the argument number or row index is invalid

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 152 of 295	

Method Name	Description	Input(s)	Output
getCommandCode	Get the command code (as a string) at the specified row in the command data	int: Table data row index	String: Command code (as a string) at the specified row in the command data; null if the row index is invalid
getCommandName	Get the command name at the specified row in the command data	int: Table data row index	String: Command name at the specified row in the command data; null if the row index is invalid
getCommandTableColumnNames	Get the column names for the table referenced on the specified row of the command table data	int: Command table data row index	String[]: Array containing the names of the columns of the command table referenced in the specified row of the command table data
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) int: Table data row index	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)

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) int: Table data row index	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: Table data row index	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; 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 for the table type "command"; return -1 if an instance of the command table type doesn't exist

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 for getTypeNameByRow 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: Table data row index	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 int: Size of the message header in bytes. For example, the CCSDS header size is 12  String: Name of the message ID name data field (e.g., 'Message ID name') boolean: true to combine memory copy calls for consecutive variables in the copy table boolean: false to retain any macros in the variable names; true to replace any macros with their corresponding values	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

Method Name	Description	Input(s)	Output
getCopyTableEntries	Get the copy table for the messages of the specified data stream	String: Data stream name  int: Size of the message header in bytes. For example, the CCSDS header size is 12  String[][]: Array containing string array entries giving the structure table path+name and the table's associated message ID name  boolean: true to combine memory copy calls for consecutive variables in the copy table  boolean: false to retain any macros in the variable names; true to replace any macros with their corresponding values	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

Method Name	Description	Input(s)	Output
getCopyTableEntriesWithMacros	Get the copy table for the messages of the specified data stream. Any macro embedded in a variable name is left in place	String: Data stream name int: Size of the message header in bytes. For example, the CCSDS header size is 12  String: Name of the message ID name data field (e.g., 'Message ID name') boolean: true to combine memory copy calls for consecutive variables in the copy table	String[][]: Array containing the copy table entries with any macro embedded in a variable name left in place; returns blank if there are no entries for the specified data stream or if data stream name is invalid
getCopyTableEntriesWithMacros	Get the copy table for the messages of the specified data stream. Any macro embedded in a variable name is left in place	String: Data stream name int: Size of the message header in bytes. For example, the CCSDS header size is 12  String[][]: Array containing string array entries giving the structure table path+name and the table's associated message ID name  boolean: true to combine memory copy calls for consecutive variables in the copy table	String[][]: Array containing the copy table entries with any macro embedded in a variable name left in place; returns blank if there are no entries for the specified data stream or if data stream name is invalid

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 157 of 295	

Method Name	Description	Input(s)	Output
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
getDataStreamNames	Get a string array containing all of the data stream names in the project		String[]: Array containing the unique data stream names
dataTypeDefinitions	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
dataTypeSizeInBits	Get the number of bits for the specified data type	String: Structure or primitive data type	int: Number of bits required to store the data type; returns 0 if the data type doesn't exist
dataTypeSizeInBytes	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

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
getFullVariableName	Get a variable's full name which includes the variables in the structure path separated by underscores, and with the data types removed	int: Table data row index	String: The variable's full path and name with each variable in the path separated by an underscore, and with the data types removed; returns a blank if the row is invalid

Method Name	Description	Input(s)	Output
getFullVariableName	Get a variable's full name which includes the variables in the structure path separated by the supplied separator character(s)	int: Table data row index String: Character(s) to place between the variable path members	String: The variable's full path and name with each variable in the path separated by the specified separator character(s), and with the data types removed; returns a blank if the row is invalid
getFullVariableName	Get a variable's full name which includes the variables in the structure path separated by the supplied separator character(s). Data types may be excluded or retained, based on the input flag. If retained, the data types and variable names are separated by the supplied separator character(s)	int: Table data row index String: Character(s) to place between the variable path members boolean: true to exclude the data types from the path + name String: Character(s) to place between the data types and variable names	String: The variable's full path and name with each variable in the path separated by the specified separator character(s); returns a blank if the row is invalid
getFullVariableName	Get a variable's full name which includes the variables in the structure path separated by the specified separator character(s) and with the data types removed. In case there are any array member variable names in the full name, replace left square brackets with # underscores and remove right square brackets (example: a[0],b[2] becomes a_Oseparatorb_2)	String: Path to the variable in the format <i>rootTable[,structureDataT ype1.variable1[,structureD ataType2.variable2[...]]]</i> String: Name of the variable in the format <i>primitiveDataType.variable</i> String: Character(s) to place between the variable path members	String: The variable's full path and name with each variable in the path separated by the specified separator character(s) and with the data types removed; returns a blank if the row is invalid

Method Name	Description	Input(s)	Output
getFullVariableName	Get a variable's full name which includes the variables in the structure path separated by the supplied separator character(s). Data types may be excluded or retained, based on the input flag	String: Path to the variable in the format <i>rootTable[,structureDataType1.variable1[,structureDataType2.variable2[,...]]]</i> String: Name of the variable in the format <i>primitiveDataType.variable</i> String: Character(s) to place between the variable path members boolean: true to exclude the data types from the path + name String: Character(s) to place between the data types and variable names	String: The variable's full path and name with each variable in the path separated by the specified separator character(s); returns a blank if the row is invalid
getFullVariableName	Get a variable's full name which includes the variables in the structure path separated by the supplied separator character(s)	String: Variable path + name in the format <i>rootTable[,structureDataType1.variable1[,structureDataType2.variable2[,...]]],primitiveDataType.variable</i> String: Character(s) to place between the variable path members	String: The variable name for the specified row prepended with each structure variable in its path, separated by the supplied separator character(s)

Method Name	Description	Input(s)	Output
getFullVariableName	Get a variable's full name which includes the variables in the structure path separated by the supplied separator character(s). Data types may be excluded or retained, based on the input flag	String: Variable path + name in the format <i>rootTable[,structureDataType1.variable1[,structureDataType2.variable2[,...]]],primitiveDataType.variable</i>  String: Character(s) to place between the variable path members  boolean: true to exclude the data types from the path + name  String: Character(s) to place between the data types and variable names	String: The variable name for the specified row prepended with each structure variable in its path, separated by the supplied separator character(s), and with the data types removed if the input flag is set to true
getFullVariableNameRaw	Get a variable's full name in the application's native format, which includes the variables in the structure path separated by commas, and with the data type and variable names separated by periods	int: Table data row index	String: The variable's full path and name with each variable in the path separated by a comma, and with each data type and variable name separated by a period; returns a blank if the row is invalid
getGroupDataFieldDescription	Get the description of the data field for the specified group's specified data field	String: Group name String: Data field name	String: Data field's description; returns a blank if the group name or data field name is invalid

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 162 of 295	

Method Name	Description	Input(s)	Output
getGroupDataFieldNames	Get the name(s) of the data field(s) associated with the specified group	String: Name of the group to which the field is a member	String: Array of the data field names associated with the specified group; returns an empty array if the group name is invalid or the group has no data fields
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 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
getGroupFields	Get the data field information for the specified group	String: Group name	String[][][]: Array containing the data field information for the specified group; an empty array if the group has no data fields, or the group doesn't exist. Each row in the array describes a single data field in the format: field name, description, size, input type, required (true or false), applicability, value

Method Name	Description	Input(s)	Output
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
getGroupTables	Get an array containing the table members, including the member table ancestor tables, for the specified group	String: Group name	String[]: Array containing the table members for the specified group; an empty array if the group has no table members, or the group doesn't 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

Method Name	Description	Input(s)	Output
getITOSEncodedDataType	Convert a primitive data type into its ITOS encoded form	String: Data type; e.g., uint16, double  String: ITOS encoding type: <b>SINGLE_CHAR</b> to get the single character encoding (e.g., "I" for any integer type) <b>TWO_CHAR</b> to get the encoding character with the data type size (e.g., "I4" for a 4-byte integer) <b>BIG_ENDIAN</b> to get the encoding as big endian <b>BIG_ENDIAN_SWAP</b> to get the encoding as a big endian with byte swapping <b>LITTLE_ENDIAN</b> to get the encoding as little endian <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; returns the data type, unmodified, if the data type is a table (i.e., it's a structure), or null if the data type is unrecognized. 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

Method Name	Description	Input(s)	Output
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
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 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 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 Integer: Initial minimum width; null to use zero as the minimum	int: Character length of the longest string in the supplied array; null if an input is invalid

Method Name	Description	Input(s)	Output
getLongestStrings	Get the character length of the longest string for each column in the supplied string array	String[][]: Array of string arrays Integer[]: Initial minimum widths; null to use zero as the minimums	Integer[]: Character length of the longest string in each column of the supplied array; null if any of the inputs is invalid
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
getMessageIDsOwnersIDsAndNames	Get an array containing every message ID name and its corresponding message ID, and the owning entity from every table cell, data field (table or group), and telemetry message. ID names and IDs are determined by the input data type assigned to the table column or data field, and are matched one-to-one by relative position; i.e., the first message ID name data field for a table or group is paired with the first message ID data field, and so on. If more names are defined than IDs or vice versa then a blank ID/name is paired with the unmatched name/ID		String[][]: Two-dimensional array containing every message ID name and its corresponding message ID, and the owning entity, sorted by owner name. Each row in the array is an array in the form [owner name], [message ID name], [message ID]. The owner name is preceded by 'Group:' if the owner is a group, and by "Tlm:" if the owner is a telemetry message
getNumberOfTimeSlots	Get the number of time slots for the scheduler definition table		int: Number of time slots for the scheduler definition table table
getNumCommandArguments	Get the number of arguments associated with the command table type at the specified row in the command data	int: Row index	int: Number of arguments associated with the command table type at the specified row in the command data

Method Name	Description	Input(s)	Output
getNumCommandArguments	Get the number of arguments associated with the specified command table type	String: Table type (case insensitive)	int: Number of arguments associated with the specified command table type; -1 if the table type is invalid
getOutputPath	Get the script output file path set via the program preferences dialog or command line		String: Script output file path; blank if no path has been set
getPathByRow	Get the path to which the specified row's data belongs with any embedded macro replaced by its corresponding value	String: Table type (case insensitive). 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"  int: Table data row index	String: The path to the current row's parameter with any embedded macro replaced by its corresponding value; returns a blank if an instance of the table type doesn't exist or the row number is invalid. The path starts with the root table name. For structure tables the root 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:  <i>rootTable[,structureDataType 1.variable1[,structureDataType e2.variable2[...]]]</i>

Method Name	Description	Input(s)	Output
getPathByRowWithMacros	Get the path to which the specified row's data belongs with any embedded macro(s) left in place	String: Table type (case insensitive). 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"  int: Table data row index	String: The path to the current row's parameter with any embedded macro(s) left in place; returns a blank if an instance of the table type doesn't exist or the row number is invalid. The path starts with the root table name. For structure tables the root 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:  <i>rootTable[,structureDataType1.variable1[,structureDataType2.variable2[...]]]</i>
getProject	Get the project's name		String: Name of the project
getProjectDataFieldDescription	Get the description of the data field for the specified project data field	String: Data field name	String: Data field's description; returns a blank if the project data field name is invalid
getProjectDataFieldNames	Get the name(s) of the data field(s) associated with the project		String: Array of the data field names associated with the project; returns an empty array if the project has no data fields

Method Name	Description	Input(s)	Output
getProjectFieldValue	Get the contents of the data field for the specified project data field	String: Data field name	String: Data field's value; returns a blank if the project data field name is invalid
getProjectDescription	Get the project's description		String: Description of the project
getProjectFields	Get the data field information for the project		String[][]: Array containing the data field information for the project; an empty array if the project has no data fields. Each row in the array describes a single data field in the format: field name, description, size, input type, required (true or false), applicability, value
getPrototypeName	Get the name of the prototype table for the specified table	String: Table name	String: The name of the prototype table for the specified table
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 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
getRootStructureTableNames	Get the name(s) of the root structure table(s). Convenience method for getRootTableNames that assumes the table type is "structure"		String[]: Array containing the root structure table names; returns an empty array if an instance of the structure table type doesn't exist

Method Name	Description	Input(s)	Output
getRootTableNames	Get the name(s) of the root table(s) for the supplied table type. Note that only structure tables can have child tables so using this method for non-structure tables returns the same list of tables as getTableNames(typeName)	String: Table type (case insensitive). 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"	String[]: Array containing the root table names for the type specified; returns an empty array if an instance of the table type doesn't exist
getScriptName	Get the name of the script file being executed		String: Script file name
getStructureArraySize	Get the variable array size at the specified row in the structure data, with any macro name replaced by its corresponding value	int: Table data row index	String: Variable array size at the specified row in the structure data, with any macro name replaced by its corresponding value; null if the row index is invalid
getStructureArraySizeWithMacros	Get the variable array size at the specified row in the structure data, with any embedded macro(s) left in place	int: Table data row index	String: Variable array size at the specified row in the structure data, with any embedded macro(s) left in place; null if the row index is invalid
getStructureBitLength	Get the variable bit length at the specified row in the structure data, with any macro name replaced by its corresponding value	int: Table data row index	String: Variable bit length at the specified row in the structure data, with any macro name replaced by its corresponding value; null if the row index is invalid

Method Name	Description	Input(s)	Output
getStructureBitLengthWithMacros	Get the variable bit length at the specified row in the structure data, with any embedded macro(s) left in place	int: Table data row index	String: Variable bit length at the specified row in the structure data, with any embedded macro(s) left in place; null if the row index is invalid
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 for getTableDataByColumnName 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  String: Variable name  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 getTableDataByColumnName 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  String: Variable name  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
getStructureDataType	Get the variable data type at the specified row in the structure data	int: Table data row index	String: Variable data type at the specified row in the structure data; null if the row index is invalid

Method Name	Description	Input(s)	Output
getStructureDescription	Get the variable description at the specified row in the structure data, with any macro name replaced by its corresponding value	int: Table data row index	String: Variable description at the specified row in the structure data, with any macro name replaced by its corresponding value; null if the row index is invalid or no column has the 'Units' input type
getStructureDescriptionWithMacros	Get the variable description at the specified row in the structure data, with any embedded macro(s) left in place	int: Table data row index	String: Variable description at the specified row in the structure data, with any embedded macro(s) left in place; null if the row index is invalid or no column has the 'Units' input type
getStructureEnumerations	Get the variable enumeration(s) at the specified row in the structure data, with any macro name replaced by its corresponding value	int: Table data row index	String: Array containing the variable enumeration(s) at the specified row in the structure data, with any macro name replaced by its corresponding value; null if the row index is invalid
getStructureEnumerationsWithMacros	Get the variable enumeration(s) at the specified row in the structure data, with any embedded macro(s) left in place	int: Table data row index	String: Array containing the variable enumeration(s) at the specified row in the structure data, with any embedded macro(s) left in place; null if the row index is invalid

Method Name	Description	Input(s)	Output
getStructureParentRowByChildRow	Get the row index in the structure data for the first entry associated with the parent structure of the entry on the specified row of the structure data	int: Structure table data row index	int: The row index in the structure data for the first entry associated with the parent structure of the entry on the specified row of the structure data
getStructurePathByRow	Get the structure path to which the specified row's data belongs with any embedded macro replaced by its corresponding value. Convenience method that assumes the table type is "structure"	int: Table data row index	String: The structure path to the current row's parameter with any embedded macro replaced by its corresponding value; returns a blank if an instance of the table type doesn't exist or the row number is invalid. The path starts with the root table name. For structure tables the root 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:  <i>rootTable[,structureDataType 1.variable1[,structureDataType e2.variable2[...]]]</i>

Method Name	Description	Input(s)	Output
getStructurePathByRowWithMacros	Get the structure path to which the specified row's data belongs with any embedded macro(s) left in place. Convenience method that assumes the table type is "structure"	int: Table data row index	String: The structure path to the current row's parameter with any embedded macro(s) left in place; returns a blank if an instance of the table type doesn't exist or the row number is invalid. The path starts with the root table name. For structure tables the root 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:  <i>rootTable[,structureDataType 1.variable1[,structureDataType e2.variable2[...]]]</i>
getStructureRates	Get the variable rate(s) at the specified row in the structure data	int: Table data row index	String: Array containing the variable rate(s) at the specified row in the structure data; null if the row index is invalid

Method Name	Description	Input(s)	Output
getStructuresByReferenceOrder	Get an array containing the names of the prototype structures in the order in which they are referenced; that is, the structure array is arranged so that a structure appears in the array prior to a structure that references it		String[]: Array containing the names of the prototype structures in the order in which they are referenced
getStructureTableColumnNames	Get the column names for the table referenced on the specified row of the structure table data	int: Structure table data row index	String[]: Array containing the names of the columns of the structure table referenced in the specified row of the structure table data
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) int: Table data row index	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)

Method Name	Description	Input(s)	Output
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) int: Table data row index	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
getStructureTableITOSPathByRow	Get the structure path to which the specified row's data belongs, formatted for use in an ITOS record statement, and with any macro name replaced by its corresponding value	int: Table data row index	String: The path to the current row's parameter formatted for use in an ITOS record statement and with any macro name replaced by its corresponding value; returns a blank if an instance of the table type doesn't exist or the row number is invalid. The path starts with the root table name. The root 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: <i>rootTable[.variable1[.variable2[...]]]</i>

Method Name	Description	Input(s)	Output
getStructureTableITOSPathByRowWithMacros	Get the structure path to which the specified row's data belongs, formatted for use in an ITOS record statement, and with any macro name(s) left in place	int: Table data row index	String: The path to the current row's parameter formatted for use in an ITOS record statement and with any macro name(s) left in place; returns a blank if an instance of the table type doesn't exist or the row number is invalid. The path starts with the root table name. The root 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: <i>rootTable[.variable1[.variable2[...]]]</i>
getStructureTableNameByRow	Get the prototype structure table name to which the specified row's data belongs. Convenience method for getTableNameByRow that assumes the table type is "structure"	int: Table data row index	String: Prototype 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 array of all prototype structure table names referenced in the table data. Convenience method that specifies the table type as "structure"		String[]: Array of all prototype structure table names; returns an empty array if an instance of the structure table type doesn't exist

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 for the table type "structure"; return -1 if an instance of the structure table type doesn't exist
getStructureTablePaths	Get array of all structure table names, including paths for child structure tables, referenced in the table data. Convenience method that specifies the table type as "structure"		String[]: Array of all structure table names, including paths for child structure tables; returns an empty array 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 root structure and variable names and with any embedded macro replaced by its corresponding value. This format is used when referencing a structure table's data fields	int: Table data row index	String: The path to the current row's parameter with any embedded macro replaced by its corresponding value; returns a blank if an instance of the table type doesn't exist or the row number is invalid. The path starts with the root table name. The root 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: <i>rootTable[,variable1[,variable2[...]]]</i>

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 179 of 295	

Method Name	Description	Input(s)	Output
getStructureTableVariablePathByRowWithMacros	Get the structure path to which the specified row's data belongs and with any embedded macro(s) left in place, showing only the root structure and variable names. This format is used when referencing a structure table's data fields	int: Table data row index	String: The path to the current row's parameter with any embedded macro(s) left in place; returns a blank if an instance of the table type doesn't exist or the row number is invalid. The path starts with the root table name. The root 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:  <i>rootTable[,variable1[,variable2[...]]]</i>
getStructureTypeNameByRow	Get the table type name referenced in the specified row of the structure table type data. Convenience method for <code>getTableNameByRow</code> 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: Table data row index	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

Method Name	Description	Input(s)	Output
getStructureUnits	Get the variable units at the specified row in the structure data, with any macro name replaced by its corresponding value	int: Table data row index	String: Variable units at the specified row in the structure data, with any macro name replaced by its corresponding value; null if the row index is invalid or no column has the 'Description' input type
getStructureUnitsWithMacros	Get the variable units at the specified row in the structure data, with any embedded macro(s) left in place	int: Table data row index	String: Variable units at the specified row in the structure data, with any embedded macro(s) left in place; null if the row index is invalid or no column has the 'Description' input type
getStructureVariableName	Get the variable name at the specified row in the structure data, with any macro name replaced by its corresponding value	int: Table data row index	String: Variable name at the specified row in the structure data, with any macro name replaced by its corresponding value; null if the row index is invalid
getStructureVariableNameWithMacros	Get the variable name at the specified row in the structure data, with any embedded macro(s) left in place	int: Table data row index	String: Variable name at the specified row in the structure data, with any embedded macro(s) left in place; null if the row index is invalid

Method Name	Description	Input(s)	Output
getTableColumnNames	Get the table column names for the table referenced on the specified row of the table data for the table type specified	String: Table type (case insensitive). 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"  int: Table data row index	String[]: Array containing the names of the columns of the table type referenced in the specified row of the type's table data
getTableColumnNamesByType	Get the table column names for the table type specified	String: Table type name	String[]: Array containing the names of the columns of the table type specified
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 (case insensitive). 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"  String: Table column name (case insensitive)  int: Table data row index	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

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 (case insensitive). 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"  String: Full table path  String: Name of the column containing that matching name (case insensitive)  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  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

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 (case insensitive). 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"  String: Full table path  String: Name of the column containing that matching name (case insensitive)  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  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  String: Data field name	String: Data field's description; returns a blank if the table name or data field name is invalid

Method Name	Description	Input(s)	Output
getTableDataFieldNames	Get the name(s) of the data field(s) associated with the specified table	String: Name of the table, including the path if this table references a structure, to which the field is a member	String: Array of the data field names associated with the specified table; returns an empty array if the table name is invalid or the table has no data fields
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 String: Data field name	String: Data field's value; returns a blank if the table name or data field name is invalid
getTableDataFieldValues	Get the data field value for all tables of the specified type that have the specified data field	String: Table type (case insensitive). 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". null to include tables of any type 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)
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)

Method Name	Description	Input(s)	Output
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 (case insensitive). 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"  String: Table column name (case insensitive)  int: Table data row index	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
getTableDescription	Get the description of the specified table	String: Table name, including the full path for child structure tables	String: Description of the specified table; returns a blank if the table doesn't exist
getTableDescriptionByRow	Get the description of the table at the row indicated for the table type specified	String: Table type (case insensitive). 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"  int: Table data row index	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

Method Name	Description	Input(s)	Output
getTableNameByRow	Get the prototype table name for the type specified to which the specified row's parameter belongs	String: Table type (case insensitive). 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"  int: Table data row index	String: Prototype table name to which the current row's parameter belongs; return a blank if an instance of the table type or the row doesn't exist
getTableNames	Get array of all table names referenced in the table data of the specified table type	String: Table type (case insensitive). 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"  boolean: true to return only the prototype name for any child structures; false to include the full path for child structures	String[]: Array of all table names, with paths for child structure tables excluded based on the input flag, represented by the table type; returns an empty array if an instance of the table type doesn't exist
getTableNames	Get array of all table names, including paths for child structure tables, referenced in the table data for all table types		String[]: Array of all table names, including paths for child structure tables, referenced in the table data; empty array if no tables exists in the data

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 187 of 295	

Method Name	Description	Input(s)	Output
getTableNames	Get array of all table names, including paths for child structure tables, referenced in the table data of the specified table type	String: Table type (case insensitive). 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"	String[]: Array of all table names, including paths for child structure tables, represented by the table type (prototype names for child structures); returns an empty array if an instance of the table type doesn't exist
getTableNumRows	Get the number of rows of data for all table types		int: Number of rows of data for all table types; return 0 if there is no table data
getTableNumRows	Get the number of rows of for the table type specified	String: Table type (case insensitive). 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"	int: Number of rows of data in the table for the table type specified; return -1 if an instance of the table type doesn't exist
getTelemetryMessageIDs	Get the copy table message ID names and their corresponding ID values for the specified data stream	String: Data stream name	String: Array containing the copy table 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

Method Name	Description	Input(s)	Output
getTypeDataFieldDescription	Get the description of the data field for the specified table type's specified data field	String: Table type (case insensitive). 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" String: Data field name	String: Data field's description; returns a blank if the table type name or data field name is invalid
getTypeDataFieldNames	Get the name(s) of the data field(s) associated with the specified table type	String: Name of the table type to which the field is a member	String: Array of the data field names associated with the specified table type; returns an empty array if the table type name is invalid or the table type has no data fields
getTypeDataFieldValue	Get the contents of the data field for the specified table type's specified data field	String: Table type (case insensitive). 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" String: Data field name	String: Data field's value; returns a blank if the table type name or data field name is invalid

Method Name	Description	Input(s)	Output
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 (case insensitive). 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"  int: Table data row index	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
getTypeNameByTable	Get the the table type name for the specified table	String: Name of the table. For a child structure this includes the path	String: Type name for the specified table
getUser	Get the name of the user executing the script		String: Name of the user executing the script
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  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

Method Name	Description	Input(s)	Output
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)
isArrayMember	Check if the supplied variable name represents an array member	Object: Variable name	boolean: true if the variable name is an array member
isDataTypeCharacter	Determine if the supplied data type is a character or string	String: data type	boolean: true if the supplied data type is a character or string; false otherwise
isDataTypeFloat	Determine if the supplied data type is a float or double	String: data type	boolean: true if the supplied data type is a float or double; false otherwise
isDataTypeInteger	Determine if the supplied data type is a signed or unsigned integer	String: data type	boolean: true if the supplied data type is a signed or unsigned integer; false otherwise
isDataTypePrimitive	Determine if the supplied data type is a primitive type	String: data type	boolean: true if the supplied data type is a primitive; false otherwise
isDataTypeString	Determine if the supplied data type is a character string	String: data type	boolean: true if the supplied data type is a character string; false otherwise
isDataTypeUnsignedInt	Determine if the supplied data type is an unsigned integer	String: data type	boolean: true if the supplied data type is an unsigned integer; false otherwise

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide		
	Doc. No. JSC-37494	Version 1.2.31	
	Date: May 2018	Page 191 of 295	

Method Name	Description	Input(s)	Output
isStructureShared	Determine if the specified structure is referenced by more than one root structure	String: Prototype name of the structure to check	boolean: true if the specified structure is referenced by more than one root structure; false otherwise
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
parseEnumerationParameters	Divide the supplied enumeration string into the values and labels. The enumeration value/label separator character and the enumerated pair separator character are automatically determined. Any leading or trailing white space characters are removed from each array member	String: Enumeration in the format <enum value><enum value separator><enum label>[<enum value separator>...][<enum pair separator>...]	String[][]: Two-dimensional array representing the enumeration parameters ; returns null if the input text is empty or the enumeration separator characters cannot be determined
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	

Method Name	Description	Input(s)	Output
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 PrintWriter object obtained from the openOutputFile method String: Text to write to the output file	
writeToFileFormat	Write the supplied formatted text in the indicated format to the specified output file PrintWriter object	PrintWriter: Output file PrintWriter object obtained from the openOutputFile method String: Print format string to write to the output file 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 PrintWriter object obtained from the openOutputFile method String: Text to write to the output file	

Method Name	Description	Input(s)	Output
xmlCleanSystemPath	Replace each invalid character with an underscore and move any leading underscores to the end of each path segment	String: System path in the form <>/>path1</path2<...>>	String: Path with each invalid character replaced with an underscore and any leading underscores moved to the end of each path segment
xmlGetBaseDataType	Convert the primitive data type into the base equivalent used by the xtceSetParameterDataType() and xtceSetArgumentDataType() methods	String: Data type	BasePrimitiveDataType: Base primitive data type corresponding to the specified primitive data type; null if no match

Method Name	Description	Input(s)	Output
xtceAddCommand	Add a command to the command metadata set	SpaceSystemType: Space system reference String: Command name String: Command code String: Application ID boolean: true if this table represents the command header String: Command header table system path String[]: Array of command argument names String[]: Array of of command argument data types String[]: Array of of command argument array sizes; the list item is null or blank if the corresponding argument isn't an array String: Description of the command	

Method Name	Description	Input(s)	Output
xtceAddContainerReference	Add a container reference(s) for the telemetry or command parameter or parameter array to the specified entry list	String: Reference to the telemetry or command entry list into which to place the parameter or parameter array container reference(s)  String: Parameter name  String: Data type  String: Parameter array size; null or blank if the parameter isn't an array	

Method Name	Description	Input(s)	Output
xtceAddParameterAndType	Add a parameter with a primitive data type to the parameter set and parameter type set	SpaceSystemType: Space system reference String: Parameter name String: Parameter primitive data type String: Parameter array size; null or blank if the parameter isn't an array String: Parameter bit length; null or blank if not a bit-wise parameter String: Enumeration in the format <enum label> <enum value>[ ...][,...]; null to not specify String: Parameter units String: Minimum parameter value String: Maximum parameter value String: Parameter description int: Size, in characters, of a string parameter; ignored if not a string or character	

Method Name	Description	Input(s)	Output
xtceAddParameterSequenceEntry	Add the parameter to the sequence container entry list	SpaceSystemType: Reference to the space system to which the parameter belongs String: Parameter name String: Data type String: Array size EntryListType: Reference to the entry list into which to place the parameter (for a primitive data type) or container (for a structure data type) reference boolean: true if this table represents the telemetry header or one of its descendants	boolean: true if the parameter's data type references the telemetry header or one of its descendants; otherwise return the flag status unchanged

Method Name	Description	Input(s)	Output
xtceAddSpaceSystemCommands	Add the command(s) from a table to the specified space system	SpaceSystemType: Space system reference String[][]: Table data array int: Command name column index int: Command code column index int: Command description column index boolean: true if this table represents the command header String: Command header table system path String: Application ID	
xtceAddSpaceSystemHeader	Set the space system header attributes	SpaceSystemType: Space system reference String: Classification attribute String: Validation status attribute String: Version attribute String: Export creation time and date	

xtceAddSpaceSystemParameters	Add a structure table's parameters to the telemetry meta data	SpaceSystemType: Space system reference String: Table name String[][]: Array containing the table's data int: Variable (parameter) name column index int: Parameter data type column index int: Parameter array size column index int: Parameter bit length column index int: Parameter enumeration column index; -1 if no the table has no enumeration column int: Parameter description column index; -1 if no the table has no description column int: Parameter units column index; -1 if no the table has no units column int: Minimum parameter value column index; -1 if no the table has no minimum column int: Maximum parameter value column index; -1 if no the table has no maximum column	
------------------------------	---	---	--

Method Name	Description	Input(s)	Output
		boolean: true if this table represents the telemetry header or one of its descendants String: Telemetry header table system path; null or blank is none boolean: true if the table is a root structure table String: Telemetry header application ID	
xtceCreateCommandMetadata	Create the space system command metadata	SpaceSystemType: Space system reference	
xtceCreateEnumerationList	Build an enumeration list from the supplied enumeration string	SpaceSystemType: space system reference String: Enumeration in the format <enum value><enum value separator><enum label>[<enum value separator>...][<enum pair separator>...]	EnumerationList:
xtceCreateTelemetryMetadata		SpaceSystemType: Space system reference	
xtceCreateUnitSet	Build a unit set from the supplied units string	String: Parameter or command argument units; null to not specify	UnitSet:

Method Name	Description	Input(s)	Output
xtceExport	Export the tables in XTCE XML format to the specified file. This is the main entry point when using a script association to perform the export. It calls the internal method to set up and parse the tables for export	String: Output file name boolean: true if the data is big endian boolean: true if the telemetry and command headers big endian String: Version attribute (for the space system headers) String: Validation status attribute (for the space system headers) String: First level classification attribute (for the space system headers) String: Second level classification attribute (for the space system headers) String: Third level classification attribute (for the space system headers)	boolean: true if an error occurred preventing exporting the project to the file

xtceSetArgumentDataType	Set the command argument data type and set the specified attributes	SpaceSystemType: Space system reference String: Command argument name; null to not specify String: Command argument data type; null to not specify String: Command argument array size; null or blank if the argument isn't an array String: Command argument bit length String: Command argument enumeration in the format <enum label> <enum value>[ ...][,...]; null to not specify String: Command argument units; null to not specify String: Minimum parameter value; null to not specify String: Maximum parameter value; null to not specify String: Command argument description ; null to not specify int: String size in bytes; ignored if the command argument does not have a string data type	NameDescriptionType: Command description of the type corresponding to the primitive data type with the specified attributes set
-------------------------	---	---	--

Method Name	Description	Input(s)	Output
		String: Text used to uniquely identify data types with the same name; blank if the data type has no name conflict	

Method Name	Description	Input(s)	Output
xtceSetParameterDataType	Create the telemetry parameter data type and set the specified attributes	SpaceSystemType: Space system reference  String: Parameter name; null to not specify  String: Data type; null to not specify  String: Parameter array size; null or blank if the parameter isn't an array  String: Parameter bit length; null or empty if not a bit-wise parameter  String: Enumeration in the format <enum label> <enum value>[ ...][,...]; null to not specify  String: Parameter units; null to not specify  String: Minimum parameter value; null to not specify  String: Maximum parameter value; null to not specify  String: Parameter description; null to not specify  int: Size, in characters, of a string parameter; ignored if not a string or character	

Table 9. Script Data Access Methods

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. <i>JSC-37494</i>	<i>Version 1.2.31</i>
	Date: <i>May 2018</i>	Page 205 of 295

## Appendix A. Acronyms

CCDD	CFS Command & Data Dictionary
CCSDS	Consultative Committee for Space Data Systems
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
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
XML	Extensible Markup Language
XTCE	XML Telemetric and Command Exchange

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 206 of 295

## Appendix B. Definitions

<b>Array definition</b>	In a structure table, the row where the variable name and array size are specified.
<b>Array member</b>	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.
<b>Child table</b>	A structure table that is referenced as a data type for a variable in another structure table.
<b>Data type</b>	A primitive or structure data type; see <b>Primitive type</b> and <b>Structure type</b> .
<b>Database</b>	A collection of data within a PostgreSQL server organized as tables. A CCDD <i>project</i> is a PostgreSQL database representing the data for a CFS project.
<b>Encoded type</b>	The byte order for primitive data types composed of two or more bytes. CCDD recognizes four encodings:  <i>little endian</i> bytes are stored with the least significant byte first. <i>big endian</i> bytes are stored with the least significant byte last. <i>little endian, swap</i> similar to little endian, except that each byte pair is reversed; applies only to integer and unsigned integer data types composed of one or more pairs of bytes. <i>big endian, swap</i> similar to big endian, except that each byte pair is reversed; applies only to integer and unsigned integer data types composed of one or more pairs of bytes.
<b>Instance table</b>	A structure table that is a child of another structure table (the child's <i>parent</i> table).
<b>Macro</b>	An alphanumeric string, bounded by special delimiter characters, that can be inserted into a data table cell to represent text defined by the user.
<b>Parent table</b>	The structure table for which a table is an immediate descendant (child). The parent and root tables are the same if this table is a child of a root table.
<b>Path</b>	Refers to a table path or variable path.
<b>Primitive type</b>	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 using the Data Type Manager (see paragraphs 4.5.4 and 4.9.3.11).
<b>Project</b>	Synonymous with the term <i>database</i> in this guide except when referring to the PostgreSQL default database, <i>postgres</i> .
<b>Prototype table</b>	A table created via the <b>Data   New table(s)</b> command, based on one of the table types. Instances of this table are created by using this table as the data type for a variable in a structure table. If this table is not referenced as a child in another table then it is also a root table.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 207 of 295

<b>Regular expression</b>	A special set of characters that defines a pattern for matching all or part of a text string. The search dialogs optionally allow the use of regular expressions in order to tailor a search more specifically (for example, the use of wild card characters). Information regarding the construction and use of regular expressions can be found on the web.
<b>Root table</b>	The top-level table in the hierarchical table tree; the highest level ancestor of a child table. All non-structure tables and prototype tables not referenced as the data type for a variable in a structure table are root tables.
<b>Structure type</b>	Data type that references a structure table prototype (the data type name is the structure prototype name).
<b>Table path</b>	The path to a table beginning with its root table. For a non-structure table or other top-level table the table path is the root table name. For a child structure table the path lists the child's root table and all intervening ancestor tables and variable names in direct descent to the child table .
<b>Table type</b>	A table template created using the Table Type Manager (see paragraph 4.9.3.104.9.3.7). Any number of tables may be created of a given table type.
<b>Variable path</b>	The path to a variable beginning with its root table. For a non-structure table or other top-level table the variable path is the root table name. For a variable in a child structure table the path lists the child's root table and all intervening variable names in direct descent to the child table (identical to the table path, but without the structure names other than the root's).

## Appendix C. Import and Export Format

Examples and descriptions of the CSV, EDS, JSON, and XTCE file formats used when importing and exporting tables are provided in the subsequent sections. For each format the identical project data is used to create the export output. This project data is shown in Figure 88 and Figure 89 (the data tables), Figure 90 and Figure 91 (the table type definitions), and Figure 92 (the data field, data type, and macro definitions).

The screenshot shows a Windows application window titled "MyStructure". The menu bar includes File, Edit, Row, Column, and Field. The main area displays a table with the following data:

Variable Name	Description	Units	Data Type	Array Size	Bit Length	Enumeration	Rate	Minimum	Maximum
latitude	Location: north-south		float				2		
longitude	Location: east-west		float						
width		##SIZE##	uint16_t						
depth		##SIZE##	uint16_t						
height		##SIZE##	uint16_t						
velocity			double	3			10		
engine_arm	Engine armed status		uint8_t		1	0 Off, 1 Arm	1		
engine_fire	Engine fire status		uint8_t		1	0 Off, 1 Fire	1		
thrust_level	Engine thrust level	percent	uint16_t					0	100

Below the table, a text input field contains the placeholder text "Example import/export table". At the bottom, there are fields for "System" (GNC/Engine/TLM) and "Application ID" (0xeeaa). A toolbar at the bottom right includes buttons for Ins Row, Up, Left, Undo, Store, Del Row, Down, Right, Redo, and Close.

Figure 88. Structure table for import/export format examples

The screenshot shows a software application window titled "MyCommand". The menu bar includes "File", "Edit", "Row", "Column", and "Field". A toolbar at the top has icons for "New", "Save", "Print", and "Exit". Below the toolbar is a table with the following data:

Command Name	Command Code	Description	Arg 1 Name	Arg 1 Description	Arg 1 Units	Arg 1 Data Type	Arg 1 Array Size	Arg 1 Bit Length	Arg 1 Enumeration	Arg 1 Minimum	Arg 1 Maximum
NoOp	0x0	No operation									
EngineArmEnableDisable	0x1	Arm main engine	ARM			uint8_t		1	0 Enable, 1 Disable		
EngineFireEnableDisable	0x2	Fire main engine	FIRE			uint8_t		1	0 Enable, 1 Disable		
EngineThrustLevel	0x3	Engine thrust level	THROTTLE	Set thrust level	percent	Float				0.0	100.0

A "Description" input field is located below the table. At the bottom, there are buttons for "Ins Row", "Up", "Left", "Unde", "Store", "Del Row", "Down", "Right", "Redo", and "Close".

System: GN/C/Engine/CMD Application ID: 0x1222

Figure 89. Command table for import/export examples

**Table Type Editor**

File Edit Row Field

Command Structure

Column Name	Description	Input Type	Unique	Required	Enable if Structure	Enable if Pointer
Variable Name	Parameter name	Variable name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Description	Parameter description	Description	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Units	Parameter units	Units	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Data Type	Parameter data type	Primitive & Structure	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Array Size	Parameter array size	Array index	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Bit Length	Parameter number of bits (bit values only)	Bit length	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Enumeration	Enumerated parameters	Enumeration	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Rate	Downlink data rate, samples/second	Rate	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Minimum	Minimum value	Minimum	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Maximum	Maximum value	Maximum	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Description Telemetry and data structure table definition

Telemetry message ID name  Telemetry message ID

Figure 90. Structure table type definition for import/export example

**Table Type Editor**

File Edit Row Field

Command Structure

Column Name	Description	Input Type	Unique	Required
Command Name	Command name	Command name	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Command Code	Command function code	Command code	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Description	Command description	Description	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Name	Command argument 1 name	Argument name	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Description	Command argument 1 description	Description	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Units	Command argument 1 units	Units	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Data Type	Command argument 1 data type	Primitive	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Array Size	Command argument 1 array size	Array index	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Bit Length	Command argument 1 bit length	Bit length	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Enumeration	Command argument 1 enumeration	Enumeration	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Minimum	Command argument 1 minimum value	Minimum	<input type="checkbox"/>	<input type="checkbox"/>
Arg 1 Maximum	Command argument 1 maximum value	Maximum	<input type="checkbox"/>	<input type="checkbox"/>

Command table definition

Description

Command ID name      Command message ID

Figure 91. Command table type definition for import/export example

The figure displays three windows used for defining data structures:

- Data Field Editor: MyStructure**: A table showing fields for a structure named "MyStructure". The columns are Field Name, Description, Size, Input Type, Required, and Applicability.

Field Name	Description	Size	Input Type	Required	Applicability
System		15	System Path	<input type="checkbox"/>	All tables
Application ID		15	Message ID	<input type="checkbox"/>	All tables

- Data Type Editor**: A table showing C data types and their corresponding sizes and base types.

Type Name	C Name	Size	Base Type
int8_t	signed char	1	signed integer
int16_t	signed short int	2	signed integer
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	2	character
address	void *	4	pointer

- Macro Editor**: A table showing macro definitions.

Macro Name	Value
SIZE	meters

Figure 92. Data field, data type, and macro definitions for import/export example

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
	Version 1.2.31	Page 213 of 295

## Appendix C.1. CSV

The CSV import/export file is divided into five sections: table type definitions, data type definitions, macro definitions, reserved message IDs, and table definitions. These sections can appear in any order. Not all sections need be present. The table definitions are further sub-divided into table name & type, description, column data, and data field sections.

Each section is designated by a tag in the format `_tag_name_`. The subsequent row(s) are interpreted based on the last tag name until another tag is detected. The various values in the rows following a tag are separated by commas, with each value enclosed in double quotes in order to preserve quotes and commas in the values.

The name & type tag begins each table definition. The name & type and column data for at least one table must be present in the file.

Empty rows and rows beginning with a # character are ignored and can be used for section spacing and inserting comments. A '#' character beginning a quoted string is not interpreted as a comment (note that a spreadsheet application, if used to create the CSV file, may automatically surround text intended as a comment in quotes). Extra commas and white space characters appended to a row are ignored. This allows a CSV file generated from a spreadsheet application (which appends trailing commas to ensure that each row has the same number of columns) to be used without further editing.

The tag names and formats for the sections are as follows:

### Table type definition section:

```



```

See paragraph 4.9.3.10 for more information on the table type definition components.

### Data type definition section:

```

_data_type_
"data type name", "C type", "size in bytes", "base type"

```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 214 of 295

*... repeat previous row for each data type definition*

See paragraph 4.9.3.11 for more information on the data type definition components.

#### **Macro definition section:**

*\_macros\_*

*“macro name”, “macro value”*

*... repeat previous row for each macro definition*

See paragraph 4.9.3.12 for more information on the macro definition components.

#### **Reserved message ID section:**

*\_reserved\_msg\_ids\_*

*“message ID (or ID range)”, “message ID description”*

*... repeat previous row for each reserved message ID (or ID range)*

See paragraph 4.9.3.13.2 for more information on the reserved message ID components.

#### **Table definition section:**

*\_name\_type\_*

*“table path and name”, “table type”<, “system name”>*

*\_description\_*

*“table description”*

*\_column\_data\_*

*“column 1 name”, “column 2 name”, ....*

*“row 1 column 1 value”, “row 1 column 2 value”, ...*

*“row 2 column 1 value”, “row 2 column 2 value”, ...*

*... repeat previous row for each row in the table*

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. JSC-37494 Date: May 2018	<b>Version 1.2.31</b> Page 215 of 295
---	--	--

### \_data\_fields\_

"field name", "description", "size in characters", "input type", "required", "applicability", "value"

... repeat previous row for each data field associated with the table. If left blank, default values for "size in characters" (10) and "input type" (Text) are used. The "applicability" input is unused by the table definition and may be blank

...repeat above, starting with \_name\_type\_, for each table definition

The system name under \_name\_type\_ is optional; when exporting this is the value of the data field with the **System path** input type (if present), but when importing the value is unused.

### PROJECT-LEVEL FIELDS...

The example tables, MyStructure and MyCommand, are shown below as exported in CSV format.

```
# Created Wed Apr 11 14:29:44 CDT 2018 : project = user_s_guide : host = localhost:5432 : user = rmccclune

_name_type_
"MyCommand", "Command", "GNC/Engine/CMD"
_column_data_
"Command Name", "Command Code", "Description", "Arg 1 Name", "Arg 1 Description", "Arg 1 Units", "Arg 1 Data
Type", "Arg 1 Array Size", "Arg 1 Bit Length", "Arg 1 Enumeration", "Arg 1 Minimum", "Arg 1 Maximum"
"NoOp", "0x0", "No operation", "", "", "", "", "", "", ""
"EngineArmEnableDisable", "0x1", "Arm main engine", "ARM", "", "", "uint8_t", "", "1", "0|Enable, 1|Disable", "", ""
"EngineFireEnableDisable", "0x2", "Fire main engine", "FIRE", "", "", "uint8_t", "", "1", "0|Enable,
1|Disable", "", ""
"EngineThrustLevel", "0x3", "Engine thrust level", "THRUST_LEVEL", "Set thrust
level", "percent", "float", "", "", "0.0", "100.0"
_data_fields_
"System", "", "15", "System Path", "false", "All tables", "GNC/Engine/CMD"
"Application ID", "", "7", "Message ID", "false", "All tables", "0x1222"

_name_type_
"MyStructure", "Structure", "GNC/Engine/TLM"
_description_
"Example import/export table"
_column_data_
"Variable Name", "Description", "Units", "Data Type", "Array Size", "Bit
Length", "Enumeration", "Rate", "Minimum", "Maximum"
"latitude", "Location: north-south", "", "float", "", "", "", "2", "", ""
longitude", "Location: east-west", "", "float", "", "", "", "", ""
```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. JSC-37494 Date: May 2018	<i>Version 1.2.31</i> Page 216 of 295
---	--	--

```

"width","","##SIZE##","uint16_t","","","","","","","",""
"depth","","##SIZE##","uint16_t","","","","","","","",""
"height","","##SIZE##","uint16_t","","","","","","","",""
"velocity","","","double","3","","","10","",""
"velocity[0]","X-direction","m/s","double","3","","","10","",""
"velocity[1]","Y-direction","m/s","double","3","","","10","",""
"velocity[2]","Z-direction","m/s","double","3","","","10","",""
"engine_arm","Engine armed status","","uint8_t","","1","0|Off, 1|Arm","1","",""
"engine_fire","Engine fire status","","uint8_t","","1","0|Off, 1|Fire","1","",""
"thrust_level","Engine thrust level","percent","uint16_t","","","","","","0","100"
_data_fields
"System","","15","System Path","false","All tables","GNC/Engine/TLM"
"Application ID","","7","Message ID","false","All tables","0xeea"


"Command","Command table definition"
"Command Name","Command name","Command name","true","true","false","true"
"Command Code","Command function code","Command code","true","true","false","true"
"Description","Command description","Description","false","false","false","true"
"Arg 1 Name","Command argument 1 name","Argument name","false","false","false","true"
"Arg 1 Description","Command argument 1 description","Description","false","false","false","true"
"Arg 1 Units","Command argument 1 units","Units","false","false","false","true"
"Arg 1 Data Type","Command argument 1 data type","Primitive","false","false","false","true"
"Arg 1 Array Size","Command argument 1 array size","Array index","false","false","false","true"
"Arg 1 Bit Length","Command argument 1 bit length","Bit length","false","false","false","true"
"Arg 1 Enumeration","Command argument 1 enumeration","Enumeration","false","false","false","true"
"Arg 1 Minimum","Command argument 1 minimum value","Minimum","false","false","false","true"
"Arg 1 Maximum","Command argument 1 maximum value","Maximum","false","false","false","true"
_table_type_data_fields
"Command ID name","Command ID name","15","Message ID name","true","All tables","");
"Command message ID","Command message ID","7","Message ID","true","All tables",""


"Structure","Telemetry and data structure table definition"
"Variable Name","Parameter name","Variable name","true","true","true","true"
"Description","Parameter description","Description","false","false","true","true"
"Units","Parameter units","Units","false","false","true","true"
"Data Type","Parameter data type","Primitive & Structure","false","true","true","true"
"Array Size","Parameter array size","Array index","false","false","true","true"
"Bit Length","Parameter number of bits (bit values only)","Bit length","false","false","true","true"
"Enumeration","Enumerated parameters","Enumeration","false","false","true","true"

```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 217 of 295
---	---	-----------------------------------

```

"Rate", "Downlink data rate, samples/second", "Rate", "false", "false", "false", "true"
"Minimum", "Minimum value", "Minimum", "false", "false", "false", "false"
"Maximum", "Maximum value", "Maximum", "false", "false", "false", "false"


```

## Appendix C.2. EDS

The **Namespace** *name* field is constructed using the contents of the table's data field having a **System path** input type (if present) followed by the table's full path, with all invalid characters (spaces, commas, periods, etc.) converted to underscores (this is a constraint imposed on the *name* field by the EDS schema).

When exporting, the **Namespace** *shortDescription* field is used to store the original table name with its full path. During importing this field is checked and if it exists and adheres to the constraints for a table path and name then it is used as the table name. If the field doesn't meet the requirements then the **Namespace** *name* field is used instead. The *name* field isn't used by default since it is not allowed to contain commas or periods, which are required for a table's full path.

Each **Namespace** defines a structure or command table. The **Interface** *name* field is set to "Telemetry" for structures, and "Command" for commands; however, the presence of a **ParameterSet** or **CommandSet** in the **Interface** determines how the information is assigned when the file is imported. Both a **ParameterSet** and a **CommandSet** can exist in the same **Namespace**, a condition that can exist if the EDS file is constructed by other means than via the CCDD export operation. Since the **Namespace** fields are used to set the table name, both a **ParameterSet** and a **CommandSet** within the **Namespace** would result in the creation of tables with duplicate names. This is avoided by appending "\_lm" to structure table name and "\_cmd" to the command table name for this case.

The original integer data type size for parameters with a bit length is not preserved when exporting in EDS format. The **IntegerDataEncoding** *sizeInBits* field is used to set the bit length; however, there is no field for the overall size. When the file is imported the data type size for a bit-wise parameter is set to the smallest integer into which the number of bits will fit. This can lead to a difference in the original data type sizes which in turn can affect bit-packing.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018		
		Version 1.2.31	Page 218 of 295

The **Device Metadata** is used to store the telemetry header table name (if defined), the command header table name (if defined), the telemetry and command header table application ID variable name, and the command header command function code variable name. These are defined in project-level data fields (see Table 7). Default values are used for the application ID variable name (“applicationID”) and the command function code variable name (“functionCode”) if these fields are not defined in the project database.

The structure table identified by the project-level data field as the command header is treated specially during conversion. When exported, the structure is converted into a **CommandSet** and stored in the same manner a command table. The **Interface abstract** field is set to ‘true’; this is used to indicate that the commands in this **Namespace** are common to the other command tables. When the file is imported the command header information is converted back into a structure.

When exporting, each root table (structure and command) is checked for the existence of data fields with the **Message ID** and **System path** input types. The contents of the field containing a system path is prepended to the table’s path and the result is used to construct the **Namespace name** field (after converting invalid characters to underscores as previously mentioned). When importing the file the data field for the system path is reconstructed from the **Namespace name** (the system path is separated from the table name portion using a ‘/’ character; the system path may contain ‘/’ characters as well). If the data field with a **Message ID** input type exists then its contents is used to set the application ID variable in the table’s telemetry or command header in the export file (by setting the **ValueConstraint value** field in a **ContainerDataType** within the table’s **DataTypeSet**). When imported, the data field is reconstructed to contain the ID value.

When exporting, the values in the command table’s column with the input type **Command code** are stored in the export file by setting the **ValueConstraint value** field (in a **ContainerDataType** within the table’s **DataTypeSet**) to the function code value. When the command table is imported, these values are placed in the table’s command code column.

The table defined using the project-level data field as the command header is stored once in the export file and each command table **Namespace** references this single header definition. Since the telemetry header table is referenced in the root structure tables as a header variable’s data type, each root structure table will have an individual instance of the header constructed in the export file. When importing, a single telemetry header structure table is recreated. The root structure telemetry header variable use this table as the data type reference.

The example tables, MyStructure and MyCommand, are shown below as exported in EDS format.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DataSheet xmlns="http://www.ccsds.org/schema/sois/seds" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.ccsds.org/schema/sois http://www.ccsds.org/schema/sois/seds.xsd">
  <Device name="User_s_Guide">
    <LongDescription>Project used for the import and export example in Appendix C of the user's guide

```

Author: rmcclune  
 Generated by CCDD 1.2.11 (4-11-2018)  
 Date: Wed Apr 11 14:31:48 CDT 2018  
 Project: User's Guide  
 Host: localhost:5432

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Date: <i>May 2018</i>	<i>Version 1.2.31</i> Page 219 of 295
---	--	--

```

Endianess: big</LongDescription>
<Metadata>
  <MetadataValueSet>
    <StringValue value="CCSDS_TLM_HDR" name="XML: Telemetry Header"/>
    <StringValue value="CCSDS_CMD_HDR" name="XML: Command Header"/>
    <StringValue value="applicationID" name="XML: Application ID"/>
    <StringValue value="functionCode" name="XML: Function Code"/>
  </MetadataValueSet>
</Metadata>
</Device>
<Namespace name="GNC/Engine/CMD/MyCommand" shortDescription="MyCommand">
  <DataTypeSet>
    <EnumeratedDataType name="ARM_Type">
      <IntegerDataEncoding encoding="unsigned" sizeInBits="1" byteOrder="bigEndian"/>
      <EnumerationList>
        <Enumeration value="0" label="Enable"/>
        <Enumeration value="1" label="Disable"/>
      </EnumerationList>
    </EnumeratedDataType>
    <ContainerDataType baseType="CCSDS_CMD_HDR" name="EngineArmEnableDisable">
      <ConstraintSet>
        <Constraint entry="applicationID">
          <ValueConstraint value="0x1222"/>
        </Constraint>
        <Constraint entry="functionCode">
          <ValueConstraint value="0x1"/>
        </Constraint>
      </ConstraintSet>
    </ContainerDataType>
    <EnumeratedDataType name="FIRE_Type">
      <IntegerDataEncoding encoding="unsigned" sizeInBits="1" byteOrder="bigEndian"/>
      <EnumerationList>
        <Enumeration value="0" label="Enable"/>
        <Enumeration value="1" label="Disable"/>
      </EnumerationList>
    </EnumeratedDataType>
    <ContainerDataType baseType="CCSDS_CMD_HDR" name="EngineFireEnableDisable">
      <ConstraintSet>
        <Constraint entry="applicationID">
          <ValueConstraint value="0x1222"/>
        </Constraint>
      </ConstraintSet>
    </ContainerDataType>
  </DataTypeSet>
</Namespace>

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 220 of 295		
---	--	--	--

```

      <Constraint entry="functionCode">
        <ValueConstraint value="0x2"/>
      </Constraint>
    </ConstraintSet>
  </ContainerDataType>
<FloatDataType name="THRUST_LEVEL_Type">
  <Semantics unit="percent"/>
  <FloatDataEncoding encodingAndPrecision="IEEE754_2008_single" byteOrder="bigEndian"/>
  <Range>
    <PrecisionRange>single</PrecisionRange>
  </Range>
</FloatDataType>
<ContainerDataType baseType="CCSDS_CMD_HDR" name="EngineThrustLevel">
  <ConstraintSet>
    <Constraint entry="applicationID">
      <ValueConstraint value="0x1222"/>
    </Constraint>
    <Constraint entry="functionCode">
      <ValueConstraint value="0x3"/>
    </Constraint>
  </ConstraintSet>
</ContainerDataType>
</DataTypeSet>
<DeclaredInterfaceSet>
  <Interface name="Command">
    <BaseTypeSet>
      <BaseType type="CCSDS_CMD_HDR/Command"/>
    </BaseTypeSet>
    <CommandSet>
      <Command name="NoOp">
        <LongDescription>No operation</LongDescription>
      </Command>
      <Command name="EngineArmEnableDisable">
        <LongDescription>Arm main engine</LongDescription>
        <Argument type="ARM_Type" name="ARM"/>
      </Command>
      <Command name="EngineFireEnableDisable">
        <LongDescription>Fire main engine</LongDescription>
        <Argument type="FIRE_Type" name="FIRE"/>
      </Command>
      <Command name="EngineThrustLevel">

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Date: <i>May 2018</i>	<i>Version 1.2.31</i> Page 221 of 295
---	--	--

```

<LongDescription>Engine thrust level</LongDescription>
<Argument type="THRUST_LEVEL_Type" name="THRUST_LEVEL">
  <LongDescription>Set thrust level</LongDescription>
  <ValidRange/>
</Argument>
</Command>
</CommandSet>
</Interface>
</DeclaredInterfaceSet>
</Namespace>
<Namespace name="GNC/Engine/TLM/MyStructure" shortDescription="MyStructure">
  <LongDescription>Example import/export table</LongDescription>
  <DataTypeSet>
    <ContainerDataType name="MyStructure_Type">
      <EntryList>
        <Entry type="float_Type" name="latitude"/>
        <Entry type="float_Type" name="longitude"/>
        <Entry type="uint16_t_Type" name="width"/>
        <Entry type="uint16_t_Type" name="depth"/>
        <Entry type="uint16_t_Type" name="height"/>
        <Entry type="double_Array" name="velocity"/>
        <Entry type="uint8_t_Type" name="engine_arm"/>
        <Entry type="uint8_t_Type" name="engine_fire"/>
        <Entry type="uint16_t_Type" name="thrust_level">
          <ValidRange>
            <MinMaxRange min="0" max="100" rangeType="inclusiveMinInclusiveMax"/>
          </ValidRange>
        </Entry>
      </EntryList>
    </ContainerDataType>
    <FloatDataType name="latitude_Type">
      <LongDescription>Location: north-south</LongDescription>
      <FloatDataEncoding encodingAndPrecision="IEEE754_2008_single" byteOrder="bigEndian"/>
      <Range>
        <PrecisionRange>single</PrecisionRange>
      </Range>
    </FloatDataType>
    <FloatDataType name="longitude_Type">
      <LongDescription>Location: east-west</LongDescription>
      <FloatDataEncoding encodingAndPrecision="IEEE754_2008_single" byteOrder="bigEndian"/>
      <Range>

```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 222 of 295

```

        <PrecisionRange>single</PrecisionRange>
    </Range>
</FloatDataType>
<IntegerDataType name="width_Type">
    <IntegerDataEncoding encoding="unsigned" sizeInBits="16" byteOrder="bigEndian"/>
    <Range/>
</IntegerDataType>
<IntegerDataType name="depth_Type">
    <IntegerDataEncoding encoding="unsigned" sizeInBits="16" byteOrder="bigEndian"/>
    <Range/>
</IntegerDataType>
<IntegerDataType name="height_Type">
    <IntegerDataEncoding encoding="unsigned" sizeInBits="16" byteOrder="bigEndian"/>
    <Range/>
</IntegerDataType>
<ArrayType dataTypeRef="velocity_Type" name="velocity_Array">
    <DimensionList>
        <Dimension size="3"/>
    </DimensionList>
</ArrayType>
<FloatDataType name="velocity_Type">
    <FloatDataEncoding encodingAndPrecision="IEEE754_2008_double" byteOrder="bigEndian"/>
    <Range>
        <PrecisionRange>double</PrecisionRange>
    </Range>
</FloatDataType>
<EnumeratedDataType name="engine_arm_Type">
    <LongDescription>Engine armed status</LongDescription>
    <IntegerDataEncoding encoding="unsigned" sizeInBits="1" byteOrder="bigEndian"/>
    <EnumerationList>
        <Enumeration value="0" label="Off"/>
        <Enumeration value="1" label="Arm"/>
    </EnumerationList>
</EnumeratedDataType>
<EnumeratedDataType name="engine_fire_Type">
    <LongDescription>Engine fire status</LongDescription>
    <IntegerDataEncoding encoding="unsigned" sizeInBits="1" byteOrder="bigEndian"/>
    <EnumerationList>
        <Enumeration value="0" label="Off"/>
        <Enumeration value="1" label="Fire"/>
    </EnumerationList>

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 223 of 295		
---	--	--	--

```

      </EnumeratedDataType>
      <Integer DataType name="thrust_level_Type">
        <LongDescription>Engine thrust level</LongDescription>
        <Semantics unit="percent"/>
        <IntegerDataEncoding encoding="unsigned" sizeInBits="16" byteOrder="bigEndian"/>
        <Range/>
      </Integer DataType>
    </DataTypeSet>
    <DeclaredInterfaceSet>
      <Interface name="Telemetry">
        <ParameterSet>
          <Parameter type="latitude_Type" name="latitude">
            <LongDescription>Location: north-south</LongDescription>
          </Parameter>
          <Parameter type="longitude_Type" name="longitude">
            <LongDescription>Location: east-west</LongDescription>
          </Parameter>
          <Parameter type="width_Type" name="width"/>
          <Parameter type="depth_Type" name="depth"/>
          <Parameter type="height_Type" name="height"/>
          <Parameter type="velocity_Array" name="velocity"/>
          <Parameter type="engine_arm_Type" name="engine_arm">
            <LongDescription>Engine armed status</LongDescription>
          </Parameter>
          <Parameter type="engine_fire_Type" name="engine_fire">
            <LongDescription>Engine fire status</LongDescription>
          </Parameter>
          <Parameter type="thrust_level_Type" name="thrust_level">
            <LongDescription>Engine thrust level</LongDescription>
          </Parameter>
        </ParameterSet>
      </Interface>
    </DeclaredInterfaceSet>
  </Namespace>
</DataSheet>

```

### Appendix C.3. JSON

A JSON object consists of a name/value pair where the value can be a single value or an array containing more name/value pairs. The CCDD JSON export file uses a unique object name for specific portions of the exported data. These object names are described below (these names are a subset of those used in the web server JSON output).

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 224 of 295
---	---	-----------------------------------

<b>File Description</b>	Information on the export file including the creation time and date, project name, database host, and user name.
<b>Data Type Definition</b>	An array containing an entry for each data type definition. Each definition has the following entries: Base Type, C Name, Type Name, and Size.
<b>Table Type Definition</b>	Array containing the table type definitions. Each array entry defines a table type and contains a <b>Table Type Column</b> array, <b>Table Type Name</b> , <b>Table Type Description</b> , and <b>Table Type Data Field</b> array.
<b>Table Type Name</b>	Table type definition name.
<b>Table Type Description</b>	Table type definition description.
<b>Table Type Column</b>	Array containing the table type column definitions. Each definition contains the Column Name, Description, Input Type, Required, Unique, Enable if Structure, and Enable if Pointer.
<b>Table Type Data Field</b>	Array of the table type's data field definitions. Each definition has the following entries: Value, Required, Description, Applicability, Field Name, Input Type, and Size.
<b>Macro Definition</b>	Array containing the macro definitions. Each definition contains the Macro Name and Macro Value.
<b>Reserved Message ID Definition</b>	Array containing the reserved message ID definitions. Each definition has entries for the Message ID(s) and Description.
<b>Variable Path</b>	Array containing the variable paths. Each entry contains the full variable path in the application's native format, and the path formatted per the export dialog options.
<b>Table Definition</b>	Table definition array. Each array entry in the table definition contains the <b>Table Data</b> , <b>Table Name</b> , <b>Table Description</b> , <b>Table Type</b> , <b>System Name</b> , and <b>Data Field(s)</b> for a single table.
<b>Table Name</b>	Table name, including its full path.
<b>Table Type</b>	Table type name.
<b>Table Description</b>	Table description.
<b>Table Data</b>	Array containing the table data. Each array entry consists of a column name and its corresponding value.
<b>System Name</b>	The value of the table's data field with the input type of <b>System path</b> , if present.
<b>Data Field</b>	Array of the table's data field definitions. Each definition has the following entries: Value, Required flag, Description, Applicability type, Field Name, Input Type, and Size.
<b>Project Data Field</b>	

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 225 of 295
---	---	-----------------------------------

The example tables, MyStructure and MyCommand, are shown below as exported in JSON format.

```
{
  "Table Definition": [
    {
      "Table Data": [
        {
          "Command Name": "NoOp",
          "Description": "No operation",
          "Command Code": "0x0"
        },
        {
          "Command Name": "EngineArmEnableDisable",
          "Description": "Arm main engine",
          "Arg 1 Data Type": "uint8_t",
          "Command Code": "0x1",
          "Arg 1 Enumeration": "0|Enable, 1|Disable",
          "Arg 1 Bit Length": "1",
          "Arg 1 Name": "ARM"
        },
        {
          "Command Name": "EngineFireEnableDisable",
          "Description": "Fire main engine",
          "Arg 1 Data Type": "uint8_t",
          "Command Code": "0x2",
          "Arg 1 Enumeration": "0|Enable, 1|Disable",
          "Arg 1 Bit Length": "1",
          "Arg 1 Name": "FIRE"
        },
        {
          "Arg 1 Units": "percent",
          "Command Name": "EngineThrustLevel",
          "Description": "Engine thrust level",
          "Arg 1 Data Type": "float",
          "Arg 1 Description": "Set thrust level",
          "Arg 1 Maximum": "100.0",
          "Arg 1 Minimum": "0.0",
          "Command Code": "0x3",
          "Arg 1 Name": "THRUST_LEVEL"
        }
      ],
    ]
  ]
}
```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 226 of 295	
---	--	--

```

"Table Name": "MyCommand",
"System Name": "GNC/Engine/CMD",
"Data Field": [
  {
    "Value": "GNC/Engine/CMD",
    "Required": false,
    "Description": "",
    "Applicability": "All tables",
    "Field Name": "System",
    "Input Type": "System Path",
    "Size": 15
  },
  {
    "Value": "0x1222",
    "Required": false,
    "Description": "",
    "Applicability": "All tables",
    "Field Name": "Application ID",
    "Input Type": "Message ID",
    "Size": 7
  }
],
"Table Description": "",
"Table Type": "Command"
},
{
"Table Data": [
  {
    "Description": "Location: north-south",
    "Data Type": "float",
    "Rate": "2",
    "Variable Name": "latitude"
  },
  {
    "Description": "Location: east-west",
    "Data Type": "float",
    "Variable Name": "longitude"
  },
  {
    "Units": "meters",
    "Data Type": "uint16_t",
  }
]
}

```

```

    "Variable Name": "width"
},
{
    "Units": "meters",
    "Data Type": "uint16_t",
    "Variable Name": "depth"
},
{
    "Units": "meters",
    "Data Type": "uint16_t",
    "Variable Name": "height"
},
{
    "Array Size": "3",
    "Data Type": "double",
    "Rate": "10",
    "Variable Name": "velocity"
},
{
    "Array Size": "3",
    "Description": "X-direction",
    "Units": "m/s",
    "Data Type": "double",
    "Rate": "10",
    "Variable Name": "velocity[0]"
},
{
    "Array Size": "3",
    "Description": "Y-direction",
    "Units": "m/s",
    "Data Type": "double",
    "Rate": "10",
    "Variable Name": "velocity[1]"
},
{
    "Array Size": "3",
    "Description": "Z-direction",
    "Units": "m/s",
    "Data Type": "double",
    "Rate": "10",
    "Variable Name": "velocity[2]"
}

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 228 of 295	
---	--	--

```

},
{
  "Description": "Engine armed status",
  "Enumeration": "0|Off, 1|Arm",
  "Data Type": "uint8_t",
  "Bit Length": "1",
  "Rate": "1",
  "Variable Name": "engine_arm"
},
{
  "Description": "Engine fire status",
  "Enumeration": "0|Off, 1|Fire",
  "Data Type": "uint8_t",
  "Bit Length": "1",
  "Rate": "1",
  "Variable Name": "engine_fire"
},
{
  "Description": "Engine thrust level",
  "Units": "percent",
  "Maximum": "100",
  "Minimum": "0",
  "Data Type": "uint16_t",
  "Variable Name": "thrust_level"
}
],
"Table Name": "MyStructure",
"System Name": "GNC/Engine/TLM",
"Data Field": [
  {
    "Value": "GNC/Engine/TLM",
    "Required": false,
    "Description": "",
    "Applicability": "All tables",
    "Field Name": "System",
    "Input Type": "System Path",
    "Size": 15
  },
  {
    "Value": "0xeea",
    "Required": false,

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 229 of 295	
---	--	--

```

      "Description": "",  

      "Applicability": "All tables",  

      "Field Name": "Application ID",  

      "Input Type": "Message ID",  

      "Size": 7  

    }  

  ],  

  "Table Description": "Example import/export table",  

  "Table Type": "Structure"  

}  

],  

"Data Type Definition": [  

  {  

    "Base Type": "unsigned integer",  

    "C Name": "unsigned char",  

    "Type Name": "uint8_t",  

    "Size": "1"  

},  

{  

  "Base Type": "floating point",  

  "C Name": "float",  

  "Type Name": "float",  

  "Size": "4"  

},  

{  

  "Base Type": "unsigned integer",  

  "C Name": "unsigned short int",  

  "Type Name": "uint16_t",  

  "Size": "2"  

},  

{  

  "Base Type": "floating point",  

  "C Name": "double",  

  "Type Name": "double",  

  "Size": "8"  

},  

],  

"File Description": "Created Wed Apr 11 14:32:46 CDT 2018 : project = user_s_guide : host =  

localhost:5432 : user = rmcclune",  

"Table Type Definition": [  

{

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b>	
	Doc. No. <i>JSC-37494</i>	<i>Version 1.2.31</i>
	Date: <i>May 2018</i>	Page 230 of 295

```

"Table Type Column": [
  {
    "Required": true,
    "Description": "Command name",
    "Unique": true,
    "Column Name": "Command Name",
    "Enable if Structure": false,
    "Enable if Pointer": true,
    "Input Type": "Command name"
  },
  {
    "Required": true,
    "Description": "Command function code",
    "Unique": true,
    "Column Name": "Command Code",
    "Enable if Structure": false,
    "Enable if Pointer": true,
    "Input Type": "Command code"
  },
  {
    "Required": false,
    "Description": "Command description",
    "Unique": false,
    "Column Name": "Description",
    "Enable if Structure": false,
    "Enable if Pointer": true,
    "Input Type": "Description"
  },
  {
    "Required": false,
    "Description": "Command argument 1 name",
    "Unique": false,
    "Column Name": "Arg 1 Name",
    "Enable if Structure": false,
    "Enable if Pointer": true,
    "Input Type": "Argument name"
  },
  {
    "Required": false,
    "Description": "Command argument 1 description",
    "Unique": false,
    "Column Name": "Arg 1 Description"
  }
]

```

```
"Column Name": "Arg 1 Description",
"Enable if Structure": false,
"Enable if Pointer": true,
"Input Type": "Description"
},
{
    "Required": false,
    "Description": "Command argument 1 units",
    "Unique": false,
    "Column Name": "Arg 1 Units",
    "Enable if Structure": false,
    "Enable if Pointer": true,
    "Input Type": "Units"
},
{
    "Required": false,
    "Description": "Command argument 1 data type",
    "Unique": false,
    "Column Name": "Arg 1 Data Type",
    "Enable if Structure": false,
    "Enable if Pointer": true,
    "Input Type": "Primitive"
},
{
    "Required": false,
    "Description": "Command argument 1 array size",
    "Unique": false,
    "Column Name": "Arg 1 Array Size",
    "Enable if Structure": false,
    "Enable if Pointer": true,
    "Input Type": "Array index"
},
{
    "Required": false,
    "Description": "Command argument 1 bit length",
    "Unique": false,
    "Column Name": "Arg 1 Bit Length",
    "Enable if Structure": false,
    "Enable if Pointer": true,
    "Input Type": "Bit length"
},
```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 232 of 295		
---	--	--	--

```
{
  "Required": false,
  "Description": "Command argument 1 enumeration",
  "Unique": false,
  "Column Name": "Arg 1 Enumeration",
  "Enable if Structure": false,
  "Enable if Pointer": true,
  "Input Type": "Enumeration"
},
{
  "Required": false,
  "Description": "Command argument 1 minimum value",
  "Unique": false,
  "Column Name": "Arg 1 Minimum",
  "Enable if Structure": false,
  "Enable if Pointer": true,
  "Input Type": "Minimum"
},
{
  "Required": false,
  "Description": "Command argument 1 maximum value",
  "Unique": false,
  "Column Name": "Arg 1 Maximum",
  "Enable if Structure": false,
  "Enable if Pointer": true,
  "Input Type": "Maximum"
}
],
"Table Type Name": "Command",
"Table Type Data Field": [
  {
    "Value": "",
    "Required": true,
    "Description": "Command ID name",
    "Applicability": "All tables",
    "Field Name": "Command ID name",
    "Input Type": "Message ID name",
    "Size": 15
  },
  {
    "Value": ""
  }
]
```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 233 of 295	
---	--	--

```

      "Required": true,
      "Description": "Command message ID",
      "Applicability": "All tables",
      "Field Name": "Command message ID",
      "Input Type": "Message ID",
      "Size": 7
    }
  ],
  "Table Type Description": "Command table definition"
},
{
  "Table Type Column": [
    {
      "Required": true,
      "Description": "Parameter name",
      "Unique": true,
      "Column Name": "Variable Name",
      "Enable if Structure": true,
      "Enable if Pointer": true,
      "Input Type": "Variable name"
    },
    {
      "Required": false,
      "Description": "Parameter description",
      "Unique": false,
      "Column Name": "Description",
      "Enable if Structure": true,
      "Enable if Pointer": true,
      "Input Type": "Description"
    },
    {
      "Required": false,
      "Description": "Parameter units",
      "Unique": false,
      "Column Name": "Units",
      "Enable if Structure": true,
      "Enable if Pointer": true,
      "Input Type": "Units"
    },
    {
      "Required": true,

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 234 of 295	
---	--	--

```

      "Description": "Parameter data type",
      "Unique": false,
      "Column Name": "Data Type",
      "Enable if Structure": true,
      "Enable if Pointer": true,
      "Input Type": "Primitive & Structure"
    },
    {
      "Required": false,
      "Description": "Parameter array size",
      "Unique": false,
      "Column Name": "Array Size",
      "Enable if Structure": true,
      "Enable if Pointer": true,
      "Input Type": "Array index"
    },
    {
      "Required": false,
      "Description": "Parameter number of bits (bit values only)",
      "Unique": false,
      "Column Name": "Bit Length",
      "Enable if Structure": true,
      "Enable if Pointer": true,
      "Input Type": "Bit length"
    },
    {
      "Required": false,
      "Description": "Enumerated parameters",
      "Unique": false,
      "Column Name": "Enumeration",
      "Enable if Structure": true,
      "Enable if Pointer": true,
      "Input Type": "Enumeration"
    },
    {
      "Required": false,
      "Description": "Downlink data rate, samples/second",
      "Unique": false,
      "Column Name": "Rate",
      "Enable if Structure": false,
      "Enable if Pointer": true,

```

```

        "Input Type": "Rate"
    },
{
    "Required": false,
    "Description": "Minimum value",
    "Unique": false,
    "Column Name": "Minimum",
    "Enable if Structure": false,
    "Enable if Pointer": false,
    "Input Type": "Minimum"
},
{
    "Required": false,
    "Description": "Maximum value",
    "Unique": false,
    "Column Name": "Maximum",
    "Enable if Structure": false,
    "Enable if Pointer": false,
    "Input Type": "Maximum"
}
],
"Table Type Name": "Structure",
"Table Type Data Field": [
{
    "Value": "",
    "Required": true,
    "Description": "Telemetry message ID name",
    "Applicability": "Roots only",
    "Field Name": "Telemetry message ID name",
    "Input Type": "Message ID name",
    "Size": 15
},
{
    "Value": "",
    "Required": true,
    "Description": "Telemetry message ID",
    "Applicability": "Roots only",
    "Field Name": "Telemetry message ID",
    "Input Type": "Message ID",
    "Size": 7
}
]
}

```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
	Version 1.2.31	Page 236 of 295

```

    ],
    "Table Type Description": "Telemetry and data structure table definition"
}
]
}

```

## Appendix C.4. XTCE

The **SpaceSystem** hierarchy is based on the contents of the table's data field having a **System path** input type (if present) and the table's full path. The system path is assumed to use the '/' character to define each level of the path. The table name becomes the final level in the hierarchy. For example, the table 'MyStructure' has the system path 'GNC/Power/TLM' and the table 'MyCommand' has the system path 'GNC/Power/CMD'. The SpaceSystem hierarchy in the export file would be:

```

<SpaceSystem name="GNC">
  <SpaceSystem name="Power">
    <SpaceSystem name="TLM">
      <SpaceSystem name="MyStructure">
      </SpaceSystem>
    </SpaceSystem>
    <SpaceSystem name="CMD">
      <SpaceSystem name="MyCommand">
      </SpaceSystem>
    </SpaceSystem>
  </SpaceSystem>
</SpaceSystem>

```

Each **SpaceSystem** with **TelemetryMetaData** or **CommandMetaData** defines a structure or command table, respectively. Both a **TelemetryMetaData** and a **CommandMetaData** can exist in the same **SpaceSystem**, a condition that can exist if the XTCE file is constructed by other means than via the CCDD export operation. Since the **SpaceSystem** *name* field is used to set the table name, both a **TelemetryMetaData** and a **CommandMetaData** within the same **SpaceSystem** would result in the creation of tables with duplicate names. This is avoided by appending "\_tlm" to structure table name and "\_cmd" to the command table name for this case.

The original integer data type size for parameters with a bit length is not preserved when exporting in XTCE format. The **IntegerDataEncoding** *sizeInBits* field is used to set the bit length; however, there is no field for the overall size. The **IntegerDataType** also has a *sizeInBits* field which could be used to store the size of the integer; however, the **EnumeratedDataType** lacks this field. Rather than have one case covered (integers) and the other not (enumerations), and since the convention would be arbitrary (i.e., which *sizeInBits* field contains the integer size and which is the bit length), both are treated the same: when the file is imported the data type size for a bit-wise parameter is set to the smallest integer into which the number of bits will fit. This can lead to a difference in the original data type sizes which in turn can affect bit-packing.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018		
		Version 1.2.31	Page 237 of 295

The root space system **AncillaryData** is used to store the telemetry header table name (if defined), the command heard table name (if defined), the telemetry and command header table application ID variable name, and the command header command function code variable name. These are defined in project-level data fields (see Table 7). Default values are used for the application ID variable name (“applicationID”) and the command function code variable name (“functionCode”) if these fields are not defined in the project database.

The structure table identified by the project-level data field as the command header is treated specially during conversion. When exported, the structure is converted into **CommandMetaData** and stored in the same manner a command table. The **MetaCommand abstract** field is set to ‘true’; this is used to indicate that the commands in this **SpaceSystem** are common to the other command tables. When the file is imported the command header information is converted back into a structure.

When exporting, each root table (structure and command) is checked for the existence of data fields with the **Message ID** and **System path** input types. The contents of the field containing a system path is used to establish the SpaceSystem hierarchy as described previously. When importing the file the data field for the system path is reconstructed from the **SpaceSystem name** fields preceding it in the table’s hierarchy (each **SpaceSystem name** is separated by a ‘/’ character). If the data field with a **Message ID** input type exists then its contents is used to set the application ID variable in the table’s telemetry or command header in the export file (by setting the **Comparison value** field as a **RestrictionCriteria** within a **SequenceContainer** in a **ContainerSet** within the table’s **TelemetryMetaData** for structure tables, and by setting the **ArgumentAssignment argumentValue** field within the **BaseMetaCommand** in the **CommandMetaData** for command tables). When imported, the data field is reconstructed to contain the ID value.

When exporting, the values in the command table’s column with the input type **Command code** are stored in the export file by setting the **ArgumentAssignment argumentValue** field within the **BaseMetaCommand** in the **CommandMetaData**) to the function code value. When the command table is imported, these values are placed in the table’s command code column.

The table defined using the project-level data field as the command header is stored once in the export file and each command table **SpaceSystem** references this single header definition. Since the telemetry header table is referenced in the root structure tables as a header variable’s data type, each root structure table will have an individual instance of the header contracted in the export file. When importing, a single telemetry header structure table is recreated. The root structure telemetry header variable use this table as the data type reference.

The example tables, MyStructure and MyCommand, are shown below as exported in XTCE format.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<SpaceSystem xmlns="http://www.omg.org/space/xtce" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
name="User's Guide" xsi:schemaLocation="https://www.omg.org/spec/XTCE/20061101
https://www.omg.org/spec/XTCE/20061101/06-11-06.xsd">
  <LongDescription>Project used for the import and export example in Appendix C of the user's
guide</LongDescription>
  <AncillaryDataSet>
    <AncillaryData name="XML: Telemetry Header">CCSDS_TLM_HDR</AncillaryData>
    <AncillaryData name="XML: Command Header">CCSDS_CMD_HDR</AncillaryData>
  
```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Date: <i>May 2018</i>	<i>Version 1.2.31</i> Page 238 of 295
---	--	--

```

<AncillaryData name="XML: Application ID">applicationID</AncillaryData>
<AncillaryData name="XML: Function Code">functionCode</AncillaryData>
</AncillaryDataSet>
<Header version="1.0" date="Wed Apr 11 14:32:59 CDT 2018" classification="DOMAIN"
validationStatus="Working">
  <AuthorSet>
    <Author>rmccclune</Author>
  </AuthorSet>
  <NoteSet>
    <Note>Generated by CCDD 1.2.11 (4-11-2018)</Note>
    <Note>Date: Wed Apr 11 14:32:59 CDT 2018</Note>
    <Note>Project: User's Guide</Note>
    <Note>Host: localhost:5432</Note>
    <Note>Endianess: big</Note>
  </NoteSet>
</Header>
<SpaceSystem name="GNC">
  <Header version="1.0" classification="SYSTEM" validationStatus="Working"/>
  <SpaceSystem name="Engine">
    <Header version="1.0" classification="SYSTEM" validationStatus="Working"/>
  <SpaceSystem name="CMD">
    <Header version="1.0" classification="SYSTEM" validationStatus="Working"/>
    <SpaceSystem name="MyCommand">
      <Header version="1.0" classification="INTERFACE" validationStatus="Working"/>
      <CommandMetaData>
        <ArgumentTypeSet>
          <EnumeratedArgumentType name="ARM_Type">
            <IntegerDataEncoding encoding="unsigned" sizeInBits="1"
bitOrder="mostSignificantBitFirst"/>
            <EnumerationList>
              <Enumeration value="0" label="Enable"/>
              <Enumeration value="1" label="Disable"/>
            </EnumerationList>
          </EnumeratedArgumentType>
          <EnumeratedArgumentType name="FIRE_Type">
            <IntegerDataEncoding encoding="unsigned" sizeInBits="1"
bitOrder="mostSignificantBitFirst"/>
            <EnumerationList>
              <Enumeration value="0" label="Enable"/>
              <Enumeration value="1" label="Disable"/>
            </EnumerationList>
          </EnumeratedArgumentType>
        </ArgumentTypeSet>
      </CommandMetaData>
    </SpaceSystem>
  </SpaceSystem>
</SpaceSystem>

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. JSC-37494 Date: May 2018		
	<i>Version 1.2.31</i>	Page 239 of 295	

```

      </EnumeratedArgumentType>
      <FloatArgumentType sizeInBits="32" name="THRUST_LEVEL_Type">
        <LongDescription>Set thrust level</LongDescription>
        <FloatDataEncoding encoding="IEEE754_1985" sizeInBits="32"/>
        <ValidRangeSet>
          <ValidRange minExclusive="0.0" maxExclusive="100.0"/>
        </ValidRangeSet>
      </FloatArgumentType>
    </ArgumentTypeSet>
    <MetaCommandSet>
      <MetaCommand name="NoOp">
        <LongDescription>No operation</LongDescription>
      </MetaCommand>
      <MetaCommand name="EngineArmEnableDisable">
        <LongDescription>Arm main engine</LongDescription>
        <BaseMetaCommand metaCommandRef="/User's
Guide/CCSDS_CMD_HDR/CCSDS_CMD_HDR">
          <ArgumentAssignmentList>
            <ArgumentAssignment argumentName="applicationID"
argumentValue="0x1222"/>
            <ArgumentAssignment argumentName="functionCode"
argumentValue="0x1"/>
          </ArgumentAssignmentList>
        </BaseMetaCommand>
        <ArgumentList>
          <Argument argumentTypeRef="ARM_Type" name="ARM"/>
        </ArgumentList>
        <CommandContainer>
          <EntryList>
            <ArgumentRefEntry argumentRef="ARM"/>
          </EntryList>
        </CommandContainer>
      </MetaCommand>
      <MetaCommand name="EngineFireEnableDisable">
        <LongDescription>Fire main engine</LongDescription>
        <BaseMetaCommand metaCommandRef="/User's
Guide/CCSDS_CMD_HDR/CCSDS_CMD_HDR">
          <ArgumentAssignmentList>
            <ArgumentAssignment argumentName="applicationID"
argumentValue="0x1222"/>
          </ArgumentAssignmentList>
        </BaseMetaCommand>
      </MetaCommand>
    </MetaCommandSet>
  </CommandDictionary>

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. JSC-37494 Date: May 2018	<b>Version 1.2.31</b> Page 240 of 295
---	--	--

```

      <ArgumentAssignment argumentName="functionCode"
argumentValue="0x2"/>
      </ArgumentAssignmentList>
    </BaseMetaCommand>
    <ArgumentList>
      <Argument argumentTypeRef="FIRE_Type" name="FIRE"/>
    </ArgumentList>
    <CommandContainer>
      <EntryList>
        <ArgumentRefEntry argumentRef="FIRE"/>
      </EntryList>
    </CommandContainer>
  </MetaCommand>
  <MetaCommand name="EngineThrustLevel">
    <LongDescription>Engine thrust level</LongDescription>
    <BaseMetaCommand metaCommandRef="/User's
Guide/CCSDS_CMD_HDR/CCSDS_CMD_HDR">
      <ArgumentAssignmentList>
        <ArgumentAssignment argumentName="applicationID"
argumentValue="0x1222"/>
        <ArgumentAssignment argumentName="functionCode"
argumentValue="0x3"/>
        </ArgumentAssignmentList>
      </BaseMetaCommand>
      <ArgumentList>
        <Argument argumentTypeRef="THRUST_LEVEL_Type" name="THRUST_LEVEL"/>
      </ArgumentList>
      <CommandContainer>
        <EntryList>
          <ArgumentRefEntry argumentRef="THRUST_LEVEL"/>
        </EntryList>
      </CommandContainer>
    </MetaCommand>
  </MetaCommandSet>
</CommandMetaData>
</SpaceSystem>
</SpaceSystem>
<SpaceSystem name="TLM">
  <Header version="1.0" classification="SYSTEM" validationStatus="Working"/>
  <SpaceSystem name="MyStructure">
    <LongDescription>Example import/export table</LongDescription>

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 241 of 295		
---	--	--	--

```

<Header version="1.0" classification="INTERFACE" validationStatus="Working"/>
<TelemetryMetaDataSet>
  <ParameterTypeSet>
    <FloatParameterType sizeInBits="32" name="latitude_Type">
      <LongDescription>Location: north-south</LongDescription>
      <FloatDataEncoding encoding="IEEE754_1985" sizeInBits="32"/>
    </FloatParameterType>
    <FloatParameterType sizeInBits="32" name="longitude_Type">
      <LongDescription>Location: east-west</LongDescription>
      <FloatDataEncoding encoding="IEEE754_1985" sizeInBits="32"/>
    </FloatParameterType>
    <IntegerParameterType sizeInBits="16" signed="false" name="width_Type">
      <UnitSet>
        <Unit>meters</Unit>
      </UnitSet>
      <IntegerDataEncoding encoding="unsigned" sizeInBits="16"
bitOrder="mostSignificantBitFirst"/>
    </IntegerParameterType>
    <IntegerParameterType sizeInBits="16" signed="false" name="depth_Type">
      <UnitSet>
        <Unit>meters</Unit>
      </UnitSet>
      <IntegerDataEncoding encoding="unsigned" sizeInBits="16"
bitOrder="mostSignificantBitFirst"/>
    </IntegerParameterType>
    <IntegerParameterType sizeInBits="16" signed="false" name="height_Type">
      <UnitSet>
        <Unit>meters</Unit>
      </UnitSet>
      <IntegerDataEncoding encoding="unsigned" sizeInBits="16"
bitOrder="mostSignificantBitFirst"/>
    </IntegerParameterType>
    <ArrayType arrayTypeRef="velocity_Type" numberOfDimensions="1"
name="velocity_Array"/>
      <FloatParameterType sizeInBits="64" name="velocity_Type">
        <FloatDataEncoding encoding="IEEE754_1985" sizeInBits="64"/>
      </FloatParameterType>
      <EnumeratedParameterType name="engine_arm_Type">
        <LongDescription>Engine armed status</LongDescription>
        <IntegerDataEncoding encoding="unsigned" sizeInBits="1"
bitOrder="mostSignificantBitFirst"/>
      </EnumeratedParameterType>
    </ArrayType>
  </ParameterTypeSet>
</TelemetryMetaDataSet>

```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 242 of 295

```

<EnumerationList>
    <Enumeration value="0" label="Off"/>
    <Enumeration value="1" label="Arm"/>
</EnumerationList>
</EnumeratedParameterType>
<EnumeratedParameterType name="engine_fire_Type">
    <LongDescription>Engine fire status</LongDescription>
    <IntegerDataEncoding encoding="unsigned" sizeInBits="1"
bitOrder="mostSignificantBitFirst"/>
    <EnumerationList>
        <Enumeration value="0" label="Off"/>
        <Enumeration value="1" label="Fire"/>
    </EnumerationList>
</EnumeratedParameterType>
<IntegerParameterType sizeInBits="16" signed="false" name="thrust_level_Type">
    <LongDescription>Engine thrust level</LongDescription>
    <UnitSet>
        <Unit>percent</Unit>
    </UnitSet>
    <IntegerDataEncoding encoding="unsigned" sizeInBits="16"
bitOrder="mostSignificantBitFirst"/>
    <ValidRange minInclusive="0" maxInclusive="100"/>
</IntegerParameterType>
</ParameterTypeSet>
<ParameterSet>
    <Parameter parameterTypeRef="latitude_Type" name="latitude"/>
    <Parameter parameterTypeRef="longitude_Type" name="longitude"/>
    <Parameter parameterTypeRef="width_Type" name="width"/>
    <Parameter parameterTypeRef="depth_Type" name="depth"/>
    <Parameter parameterTypeRef="height_Type" name="height"/>
    <Parameter parameterTypeRef="velocity_Array" name="velocity"/>
    <Parameter parameterTypeRef="engine_arm_Type" name="engine_arm"/>
    <Parameter parameterTypeRef="engine_fire_Type" name="engine_fire"/>
    <Parameter parameterTypeRef="thrust_level_Type" name="thrust_level"/>
</ParameterSet>
<ContainerSet>
    <SequenceContainer name="MyStructure">
        <EntryList>
            <ParameterRefEntry parameterRef="latitude"/>
            <ParameterRefEntry parameterRef="longitude"/>
            <ParameterRefEntry parameterRef="width"/>

```

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Date: <i>May 2018</i>	<i>Version 1.2.31</i> Page 243 of 295
---	--	--

```

<ParameterRefEntry parameterRef="depth"/>
<ParameterRefEntry parameterRef="height"/>
<ArrayParameterRefEntry parameterRef="velocity">
  <DimensionList>
    <Dimension>
      <StartingIndex>
        <FixedValue>0</FixedValue>
      </StartingIndex>
      <EndingIndex>
        <FixedValue>3</FixedValue>
      </EndingIndex>
    </Dimension>
  </DimensionList>
</ArrayParameterRefEntry>
<ParameterRefEntry parameterRef="engine_arm"/>
<ParameterRefEntry parameterRef="engine_fire"/>
<ParameterRefEntry parameterRef="thrust_level"/>
</EntryList>
</SequenceContainer>
</ContainerSet>
</TelemetryMetaDataTable>
</SpaceSystem>
</SpaceSystem>
</SpaceSystem>
</SpaceSystem>
</SpaceSystem>
</SpaceSystem>

```

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 244 of 295

## Appendix D. 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 project 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 although 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	An association with this script and table(s) already exists in the script associations table	A script association using the same script file and data table(s) is already present in the script association table in the script manager
Error	An unanticipated error occurred; cause ' <i>error cause</i> '. Error trace: <i>class name</i> : <i>method name()</i> line <i>line number</i> , < <i>further trace</i> >	An exception occurred that is not otherwise covered by the error handling routines. The cause is specified by <i>error cause</i> , followed by a method trace showing the line where the error occurred and the calls sequence leading to it
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 detected mismatched rates for variable(s) associated with <i>variable path+name</i>	The telemetry scheduler auto-fill operation determined that one or more variables that are associated (via bit-packing or string membership) with the variable <i>variable path+name</i> do not have the same rate. Open the table containing the specified variable and adjust the rate for the associated variable(s)

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 245 of 295

Type	Message	Cause
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
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 file failed to close after being written
Warning	Cannot close import file ' <i>path+file name</i> '	The import 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 246 of 295

Type	Message	Cause
Warning	Cannot close script file ' <i>path+file name</i> '	The script file failed to close after being read
Error	Cannot close server connection	An error occurred preventing closing the PostgreSQL server connection (which is accomplished by closing the connection to the default database, <i>postgres</i> ). Detail on the cause is logged in the event log
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	An attempt to connect to the PostgreSQL server, (accomplished by connecting to the default database, <i>postgres</i> ) failed. Detail on the cause is logged in the event log. This may occur if the PostgreSQL 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 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 Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 247 of 295

Type	Message	Cause
Error	Cannot create structure functions in project database ' <i>database name</i> '	The SQL and pgsql functions related to structure 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 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 tables and functions in project database ' <i>database name</i> '	The SQL and pgsql 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 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 248 of 295

Type	Message	Cause
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 format JSON output using JavaScript; cause ' <i>cause</i> '	An error occurred during JSON table data export while attempting to use JavaScript to format the output. Information on the cause is displayed
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
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 data from the custom values table	The attempt to load the table path(s), column name(s), and column value(s) in the custom values table matching the specified column name and column value failed. Detail on the cause is logged in the event log
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 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 249 of 295

Type	Message	Cause
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
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 parse import file ' <i>path+file name</i> '	The JSON import file <i>path+file name</i> contains text that is not in the expected JSON format
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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 250 of 295

Type	Message	Cause
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
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. This 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 251 of 295

Type	Message	Cause
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
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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 252 of 295

Type	Message	Cause
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
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 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 253 of 295

Type	Message	Cause
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
Warning	Data field ' <i>field name</i> ' definition applicability type ' <i>applicability type</i> ' unrecognized in import file ' <i>file name</i> '; continue?	The applicability type, <i>applicability type</i> , referenced in a data field definition with the name <i>field name</i> in the import file <i>file name</i> is not one of those recognized. Change the input type, or select <b>Ignore</b> or <b>Ignore all</b> to use the default input type ('All tables')
Warning	Data field ' <i>field name</i> ' definition input type ' <i>input type</i> ' unrecognized in import file ' <i>file name</i> '; continue?	The input type, <i>input type</i> , referenced in a data field definition with the name <i>field name</i> in the import file <i>file name</i> is not one of those recognized. Change the input type, or select <b>Ignore</b> or <b>Ignore all</b> to use the default input type ('Text')
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 'data type' invalid; structure cannot reference itself or an ancestor	The data type pasted into a structure table's data type cell matches the name of the structure or one of its ancestor structures. This causes a recursive reference, so isn't allowed. Change the data type to a valid structure reference or a primitive data type

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 254 of 295

Type	Message	Cause
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	Database query failed	A project database query executed from within a script using the getDatabaseQuery script data access method failed. The script association dialogs can also produce this error. Detail on the cause is logged in the event log
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> ' format invalid in table ' <i>table name</i> '; initial non-negative integer or separator character between enumeration value and label missing	One or more of the enumeration definitions in enumeration <i>enumeration</i> in table <i>table name</i> imported from an EDS or XTCE XML file does not have a non-negative integer as the first enumeration parameter or the character separating the enumeration value and label can't be identified. EDS and XTCE XML enumerations must be in the format specified in paragraph 4.5.6
Warning	Enumeration ' <i>enumeration</i> ' format invalid in table ' <i>table name</i> '; separator character between enumerated pairs missing	The character separating each enumerated pair can't be identified in one or more of the enumeration definitions in enumeration <i>enumeration</i> in table <i>table name</i> imported from an EDS or XTCE XML file. EDS and XTCE XML enumerations must be in the format specified in paragraph 4.5.6
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
Warning	Environment variable override key ' <i>key name</i> ' has no corresponding value	The environment variable override key <i>key name</i> entered in the script manager or script executive dialog doesn't have a value associated with it. The key/value format is < <i>key1</i> = <i>value1</i> , <i>key2</i> = <i>value2</i> <, ...>>>

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 255 of 295

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 >= <i>minimum</i> and <= <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
Warning	Field size must be less than or equal to 100	The value entered for a data field's size is greater than the maximum allowed (if the field width is allowed to be too large then when the field is rendered the effect on the user interface can result in the application becoming unusable). The number of characters that can be entered into the field is not limited by this value, only the displayed width of the field. Enter a value less than the maximum
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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 256 of 295

Type	Message	Cause
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
Warning	Group name is already in use	The group name entered in the group name text field is already in use by another group or matches the pseudo-group's name, 'All tables'. 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 semi-colons. Remove the illegal character(s)
Warning	Illegal character(s) in table name ' <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> ' information missing	The import file <i>path+file name</i> has no tag (e.g., _description_ or _column_names_) prior to the table information

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 257 of 295

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(s) 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
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)

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 258 of 295

Type	Message	Cause
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 ( <code>__app_scheduler</code> ) 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 ( <code>__app_scheduler</code> ) references applications that do not exist in the project database. The <i>number</i> invalid application references are removed
Warning	Invalid characters in column ' <i>column name</i> '; characters consistent with input type ' <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. Enter only characters matching the input type or change the input type
Warning	Invalid characters in field ' <i>field name</i> '; characters consistent with input type ' <i>input 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 input type, <i>input type</i> . Enter only characters matching the input type or change the input type
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 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 Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 259 of 295

Type	Message	Cause
Warning	Invalid input value for column ' <i>column name</i> '; data type invalid or unknown in sizeof() call	The data type referenced in a sizeof() call entered in a cell in the column <i>column name</i> is either not a recognized primitive data type or structure name, or is a valid structure name but is the prototype of this structure table or one of its child structures, producing a recursive reference. Enter a valid data type name. Use Ctrl-S to display a pop-up dialog that allows selecting from the valid data types
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 table/path name ' <i>table name</i> ' format	The imported table path/name <i>table path</i> is in the incorrect format. The import operation is canceled. The expected format is: <i>rootTable[,structureDataType1.variable1[,structureDataType2.variable2[...]]]</i> where <i>root</i> , <i>parent</i> , and <i>name</i> must begin with an alphabetic character or underscore and only contain alphabetic characters, numerals, or underscores. Correct the table's path/name in the import file
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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 260 of 295

Type	Message	Cause
Warning	Link name must be entered	The link name text field is empty. Enter a valid link name into the text field
Warning	Macro ' <i>macro name</i> ' contains a recursive reference	The macro value for the macro <i>macro name</i> in the macro editor contains a circular reference (a macro references itself). Alter the macro value to remove the circular reference
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
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&lt;0#&gt;d&lt;endText&gt;</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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 261 of 295

Type	Message	Cause
Warning	Message starting number must be an integer $\geq 0$	The message starting number in the Assign Telemetry Messages dialog is invalid. Enter a positive integer value or zero
Warning	Missing or extra data type definition input(s) in import file ' <i>file path+name</i> '; continue?	A data type definition is missing or has too many inputs in import file <i>file path+name</i>
Warning	Missing or extra macro definition input(s) in import file ' <i>file path+name</i> '; continue?	A macro definition is missing or has too many inputs in import file <i>file path+name</i>
Warning	Missing or extra reserved message ID definition input(s) in import file ' <i>file path+name</i> '; continue?	A reserved message ID definition is missing or has too many inputs in import file <i>file path+name</i>
Warning	Missing table type name in import file ' <i>file path+name</i> '; continue?	A table type definition is missing the table type name in import file <i>file path+name</i>
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
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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 262 of 295

Type	Message	Cause
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	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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 263 of 295

Type	Message	Cause
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-level data field has missing or extra input(s) in import file ' <i>file path+name</i> '; continue?	A project-level data field is missing or has too many inputs in import file <i>file path+name</i> . Add the missing inputs or delete the extraneous ones
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
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
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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 264 of 295

Type	Message	Cause
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> ' 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
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> ' column name ' <i>column name</i> ' unrecognized in import file ' <i>file path+name</i> '; continue?	The column <i>column name</i> for table <i>table name</i> in the import file <i>file path+name</i> doesn't not exist in the table type definition
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> ' printing failed; cause ' <i>error cause</i> '	Output of the table <i>table name</i> to a printer or file was unsuccessful due to the cause specified. This can be due to the printer being offline

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 265 of 295

Type	Message	Cause
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
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
Warning	Table ' <i>table name</i> ' has missing or extra data field input(s)in import file ' <i>file path+name</i> '; continue?	A data field definition in table <i>table name</i> is missing or has too many inputs in import file <i>file path+name</i>

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 266 of 295

Type	Message	Cause
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> ' cannot begin with '__'	The table name, <i>table name</i> , entered into the table name text field begins with a double underscore. The double underscore prefix is reserved for use by the application to designate internal tables in the project database. Alter the table name to meet the table naming constraints
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
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 ' <i>table type</i> ' definition has missing or extra input(s) in import file ' <i>file path+name</i> '; continue?	The table type definition <i>table type</i> is missing or has too many inputs in import file <i>file path+name</i> . Add the missing inputs or delete the extraneous ones

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 267 of 295

Type	Message	Cause
Warning	Table type ' <i>table type</i> ' definition input type ' <i>input type name</i> ' unrecognized in import file ' <i>file path+name</i> '; continue?	The table type definition <i>table type</i> references an input type <i>input type name</i> that isn't a recognized input type in import file <i>file path+name</i> . Change the input type name to one of those recognized by the application. If the error is ignored then the invalid input type defaults to Text
Warning	Table type ' <i>table type</i> ' has missing or extra data field input(s) in import file ' <i>file path+name</i> '; continue?	The table type definition <i>table type</i> is missing or has too many data field inputs in import file <i>file path+name</i> . Add the missing inputs or delete the extraneous ones
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	Tables of type 'Structure or Command' may not have more than one column with input type(s): <i>input type</i> [, <i>input type</i> 2[, ...]]	In the table editor for a table type representing a 'Structure' or 'Command', one or more column definition input types ( <i>input type</i> [, <i>input type</i> 2[, ...]]) are used multiple times, but must be unique for this table type (e.g., input type 'Variable name' in a structure table). This can occur if these input types are assigned before the table has all the columns necessary to define the table type as a structure or command, then the final input type necessary to make the type a structure/command is assigned. Assign different input types to the affected row(s), or delete these row(s), then assign the input type
Warning	Too many/few data field inputs	The number of inputs for a data field following the _data_field_tag in the CSV 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 _macros_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 _name_type_tag in the CSV import file is incorrect (should be two)

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 268 of 295

Type	Message	Cause
Warning	Too many/few table type column inputs	The number of column description inputs following the table type name and description row under the _table_type_ tag in the CSV import file is incorrect (should be six)
Warning	Too many/few table type name and description inputs	The number of inputs immediately following the _table_type_ tag in the CSV 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 input type 'text'	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 Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	Version 1.2.31 Page 269 of 295
---	---	-----------------------------------

## Appendix E. Program Notes

### Appendix E.1. Key reference

The special keys and their contexts/actions are described below.

- ←→↑↓** The left, right, up, and down arrow keys move table cell selection from the currently highlighted cell to the cell to the left, right, above, or below respectively. If the bottom (top) of the table is reached then the down (up) arrow traverses to the next component within the GUI entity as with the Tab (Shift-Tab) key. If in edit mode the left (right) key repositions the text cursor one character to the left (right) of the current cursor position until the left (right) end of the text string is reached; the up and down arrow keys have no effect while in edit mode.
- Alt-Enter** If editing a table cell with an input type that supports multiple lines, a line break is inserted into the table cell at the current text insertion point, replacing any selected character(s).
- Ctrl-A** Selects all cells in the table that currently has the focus.
- Ctrl-Delete** Deletes the currently selected row(s).
- Ctrl-C** Copies the contents of the selected table cell(s) to the clipboard buffer.
- Ctrl-E** Expands (if collapsed) or collapses (if expanded) the currently selected table or variable tree node(s). If multiple nodes are selected then the state of the uppermost one determines which action is taken for all of the selected nodes.
- Ctrl-F** Opens the **Search Session Event Log** dialog if pressed while the main application window has the focus (same as selecting the main window **File | Search log** command). Opens the table **Search** dialog if pressed while a table or table type editor dialog is open and has the focus (same as selecting editor's **File | Search** command).
- Ctrl-I** Pastes the contents of the clipboard buffer to the table cell(s). New rows are inserted to contain the pasted data.
- Ctrl-M** When pressed while in edit mode in a data table cell a pop-up list appears showing the macro or macros that are allowed in the cell (no pop-up appears if none of the defined macros is appropriate, or if no macros are defined). The chosen macro is inserted into the table cell at the current text insertion point, replacing any selected character(s) and bounded by the macro identifier characters.
- Ctrl-Shift-M** Replaces every macro reference and sizeof() call in the current table or macro editor value column with its corresponding text string, evaluated as a mathematical expression if applicable. Releasing the keys restores the macro names.
- Ctrl-S** When pressed while in the edit mode in a data table cell a pop-up list appears displaying the primitive data types and prototype structure names. If the data table represents a structure then only those prototype structures that can be used as a variable data type are displayed. When pressed while in the macro editor table a pop-up list appears displaying all primitive data types and prototype structure names. When pressed while in the data type editor's **Type Name** or **C Name** cells, and the corresponding **Base Type** is either blank or 'pointer', a pop-up list appears displaying all of the prototype structure names. The chosen data type name is inserted into the table cell.

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Date: <i>May 2018</i>	
		<i>Version 1.2.31</i>
		Page 270 of 295

<b>Ctrl-V</b>	Pastes the contents of the clipboard buffer to the table cell(s), overwriting the current contents of the cell(s).
<b>Delete</b>	When not editing a cell deletes the contents of the currently selected table cell(s). If in edit mode the Delete key removes the character immediately to the left of the text cursor.
<b>End</b>	Changes the table cell selection to the leftmost column. If in edit mode the End key repositions the text cursor to the end of the text in the input cell or field.
<b>Enter</b>	Enters edit mode when pressed while an editable table cell is selected. If in edit mode then the cell text is entered into the cell (following any validation) and the next editable cell to the right is placed in edit mode (if the rightmost column is reached then the leftmost cell in the row below is used; after the last row is reached the first row is used). If a cell containing a check box gets the focus then pressing Enter toggles the check box state rather than traversing to the next editable cell.
<b>Escape</b>	Exits edit mode in a table cell or data field, restoring the original contents of the cell or field. Removed a pop-up list (macro) if displayed without making a selection.
<b>Home</b>	Changes the table cell selection to the rightmost column. If in edit mode the Home key repositions the text cursor to the beginning of the text in the input cell or field.
<b>Insert</b>	Inserts a new, empty row into the table below the row with the currently selected cell(s). If in edit mode the Insert key adds a space character at the text cursor location and moves the cursor immediately after the inserted space.
<b>Page Down</b>	Scrolls the table one page down from its current position. Changes the cell selection to the currently selected column, with the row one page down from its previous position.
<b>Page Up</b>	Scrolls the table one page up from its current position. Changes the cell selection to the currently selected column, with the row one page up from its previous position.
<b>Shift-Delete</b>	Replaces the selected cell(s) value with that from the corresponding cell value in the tables' prototype. See paragraph 4.9.3.2.2.8.2 for more details.
<b>Space</b>	Enters edit mode when pressed while an editable table cell is selected, then inserts a space in the cell (following any existing text).
<b>Tab</b>	Changes the focus within the current GUI entity to its next component. Pressing Shift-Tab traverses the components in the opposite direction.

## Appendix E.2. Program preferences

The program 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

Many of these preference values may be changed via the **Preferences** dialog; see paragraph 4.9.1.7. The preference keys and associated descriptions are provided below.

### General

---

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 271 of 295

<b>Database</b>	The name of the project database that was connected to most recently.
<b>DatabaseBackupPath</b>	The full path to the folder to (from) which a project database was most recently backed up (restored). Used to set the initial path in the project database backup and restore dialogs.
<b>HideDataType</b>	Determines if the data type is excluded or included when displaying the variable path in the <b>Show variables</b> dialog and in structure tables containing a column with the input type 'Variable path'. The default is <b>false</b> .
<b>LogFile Path</b>	The full path name for the location where an event log was most recently opened for reading. This is not necessarily the path of the current session log. Used to set the initial path in the read log dialog.
<b>PaddingAlignment</b>	The current padding variable byte alignment value. The default is <b>4</b> . The value must be a power of 2.
<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>ScriptPath</b>	The full path to the folder to (from) which a script was most recently retrieved from (stored in) the project database. Used to set the initial path in the Script storage and retrieval dialogs.
<b>SearchStrings</b>	Text string containing the previous search dialog searches, stored as a single string separated by special delimiter characters, and used for auto-completing input in the search dialogs' search text field. Once the maximum number of retained search strings is reached further searches cause the oldest search string to be removed so that the latest one can be added.
<b>ServerStrings</b>	Text string containing the previously entered server names, stored as a single string separated by special delimiter characters, and used for auto-completing input in the <b>Database Server</b> dialog's <b>Host</b> field. Once the maximum number of retained server names is reached further entries cause the oldest name to be removed so that the latest one can be added.
<b>TableExportPath</b>	The full path to the folder to (from) which a data table was most recently exported (imported). Used to set the initial path in the data table import and export dialogs.
<b>TypeNameSeparator</b>	The character(s) in a variable path used to separate the data type and variable name in the <b>Show variables</b> dialog and in structure tables containing a column with the input type 'Variable path'. The default is an underscore (_).
<b>User</b>	The name of the latest user to attempt a server connection.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. <i>JSC-37494</i>	<i>Version 1.2.31</i>
	Date: <i>May 2018</i>	Page 272 of 295

<b>VariablePathSeparator</b>	The character(s) in a variable path used to separate the root structure table from a child variable, and a parent structure variable from a child variable in the <b>Show variables</b> dialog and in structure tables containing a column with the input type 'Variable path'. The default is an underscore (_).
<b>WebServerPort</b>	The web server port number of the server that was connected to most recently.

## L&F

---

<b>LookAndFeel</b>	The name of the selected "look and feel" that governs the application's overall appearance.
--------------------	---

## Font

---

<b>DataTableCellFont</b>	Data table and table type editor cell font.
<b>DialogButtonFont</b>	Dialog box button label font.
<b>InputFieldTextFont</b>	Font used when inputting text into dialog input fields, description fields, and data fields.
<b>LabelBoldFont</b>	Font generally used for labels in dialogs.
<b>LabelPlainFont</b>	Font used for labels in dialogs for non-emphasized text. Also used in the telemetry and applications scheduler <b>Options</b> lists and <b>Assigned Applications</b> list.
<b>MenuItemFont</b>	Menu and sub-menu item font.
<b>OtherTableCellFont</b>	Table cell font for non-data tables (e.g., event log).
<b>TableHeaderFont</b>	Table column name font.
<b>ToolTipFont</b>	Font for use when displaying tool tips. Some look & feels ignore changes to the tool tip font.
<b>TreeNodeFont</b>	Font used when displaying tree node labels, such as in a table tree, link tree, etc.

## Color

---

<b>DataTypeTextColor</b>	Text color for the data type portion structure and primitive variable name in a table or variable tree.
<b>FocusBackgroundColor</b>	Background color for a table cell that has the input focus.
<b>InputBackgroundColor</b>	Color used for the background in dialog input fields, description fields, and data fields.
<b>InputDisabledBackgroundColor</b>	Color used for the background in dialog input fields, description fields, and data fields that are currently non-editable.
<b>InputTextColor</b>	Color used for the text in dialog input fields, description fields, and data fields.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 273 of 295

<b>InvalidTextColor</b>	Text color for invalid table rows. Used in the telemetry and application scheduler tables to denote messages and time slots which are valid for the selection option.
<b>LabelTextColor</b>	General color for text labels.
<b>PaddingBackgroundColor</b>	Background color for a padding variable cell.
<b>ProtectedBackgroundColor</b>	Background color for a protected (non-editable) table cell.
<b>ProtectedTextColor</b>	Text color for a protected (non-editable) table cell.
<b>RequiredBackgroundColor</b>	Background color for table cells and input fields that are required.
<b>SelectedBackgroundColor</b>	Background color for a table's cells when the cell is selected.
<b>SelectedTextColor</b>	Text color for a table's cells when the cell is selected.
<b>SpecialLabelTextColor</b>	Text color for a group of components in a dialog.
<b>TableAlternateBackgroundColor</b>	Background cell color for even numbered rows when the cell is not selected.
<b>TableBackgroundColor</b>	Table cell background color for odd numbered rows when the cell is not selected.
<b>TableGridColor</b>	Color for a table's grid lines. Log table grid lines are not colored (i.e., the log table uses the table background color between individual cells).
<b>TableTextColor</b>	Table cell text color when the cell is not selected.
<b>TextHighlightColor</b>	Color used to highlight macros and matching search text.
<b>ToolTipBackgroundColor</b>	Background color for tool tip pop-ups. Some look & feels ignore changes to the tool tip background color.
<b>ToolTipTextColor</b>	Text color for tool tip pop-ups. Some look & feels ignore changes to the tool tip text color.
<b>TypeRequiredBackgroundColor</b>	Background color for a cell in a table type column definition that is required to define the type.
<b>ValidTextColor</b>	Text color for valid table rows. Used in the telemetry and application scheduler tables to denote messages and time slots which are valid for the selection option.

### Size

---

<b>InitialViewableListRows</b>	Number of rows of radio buttons or check boxes to display initially when using the CcddDialogHandler class methods for creating a radio button or check box selection panel.
<b>InitialViewableDataTableRows</b>	Maximum number of rows to display when a data table or table type editor is opened. The table may be resized afterwards to display fewer or more rows.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 274 of 295

<b>InitialViewableTableRows</b>	Maximum number of rows to display when a non-data table is opened. The table may be resized afterwards to display fewer or more rows.
<b>MaximumConversionLists</b>	Maximum number of variable name conversion lists to maintain in memory. A conversion list is created when a full variable name, with path, is requested using separators other than those used in a previous request. When the maximum number of lists is reached an earlier list (the second one created) is removed to make room for the new one.
<b>MaximumDataFieldLength</b>	Maximum character length for displaying a data field. Note that this is for display purposes only; the number of characters entered into the field may exceed this value.
<b>MaximumDialogLineLength</b>	Maximum number of characters to display per line in a dialog message. The dialog text is wrapped for line lengths greater than this value.
<b>MaximumDialogMessageLength</b>	Maximum number of characters to display in a dialog message. This value only is applied to dialogs displaying a list of tables in the event the number of tables is large.
<b>MaximumInitialTableCellWidth</b>	Maximum pixel width for a table column when the table is initially displayed. The column size may be changed afterwards.
<b>MaximumLogMessageLength</b>	Maximum number of remembered search strings.
<b>MaximumToolTipLineLength</b>	Maximum number of characters to display per line in a tool tip.
<b>MinimumDialogWidth</b>	Minimum dialog window width, in pixels. Any dialog displayed is this width, or larger if the contents dictate.
<b>MinimumWindowHeight</b>	Minimum frame window height, in pixels. Examples of frame windows include the data table editors and table type editor. The frame window width may not be resized below this value.
<b>MinimumWindowWidth</b>	Minimum frame window width, in pixels. Examples of frame windows include the data table editors and table type editor. The frame window height may not be resized below this value.
<b>NumberOfRememberedSearches</b>	Maximum number of search strings to store. These are used for auto-completing input in the search dialogs' search text field. Once the maximum number of search strings is reached further searches cause the oldest search string to be removed so that the latest one can be added.
<b>NumberOfRememberedServers</b>	Maximum number of server names to store. These are used for auto-completing input in the <b>Database Server</b> dialog's <b>Host</b> field. Once the maximum number of server names is reached further entries cause the oldest name to be removed so that the latest one can be added.

## Spacing

---

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. <i>JSC-37494</i>	<i>Version 1.2.31</i>
	Date: <i>May 2018</i>	Page 275 of 295

<b>ButtonGap</b>	Number of pixels between each button in a dialog box.
<b>ButtonPad</b>	Minimum number of pixels around the perimeter of a dialog box's button grouping.
<b>CellHorizontalPadding</b>	Number of pixels added to either side of the text in a table cell.
<b>CellVerticalPadding</b>	Number of pixels added to above and below the text in a table cell.
<b>DialogBorderPadding</b>	Number of pixels between the dialog box contents and the dialog's frame.
<b>DialogIconPadding</b>	Number of pixels between the icon and text message in a message dialog box.
<b>HeaderHorizontalPadding</b>	Number of pixels added to the width of a table's column header text. The padding is split equally between each side of the header text. This padding provides room for the column sort arrow.
<b>HeaderVerticalPadding</b>	Number of pixels added to the height of a table's column header text. The padding is split equally between the top and bottom of the header text.
<b>InputFieldPadding</b>	Number of pixels added to the each side of the text in an input field.
<b>LabelHorizontalSpacing</b>	Defines the horizontal spacing between a text label and an adjacent component, in pixels.
<b>LabelVerticalSpacing</b>	Defines the vertical spacing between a text label and an adjacent component, in pixels.

## Other

---

<b>EDSSchemaLocationURL</b>	The URL for the EDS schema location. This value is used to set the JAXB_SCHEMA_LOCATION property in the marshalled XML output when exporting project data in EDS format.
<b>EnvironmentVariableOverride</b>	Environment variable override key/value pairs in the format <key=value<,...>>. The override values are used to expand variables in the script paths when executing script associations via the script manager or script execution dialogs; these overrides are not used when executing via the command line execute command.
<b>XTCESchemaLocationURL</b>	The URL for the XTCE schema location. This value is used to set the JAXB_SCHEMA_LOCATION property in the marshalled XML output when exporting project data in XTCE format.

## Appendix E.3. CCDD class files

Following is a list and description of the CCDD application's Java class files.

<b>CdddApplicationParameterDialog.java</b>	Dialog for assigning the application scheduling parameters. The dialog is built on the CdddDialogHandler class.
--	---

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. <i>JSC-37494</i> Date: <i>May 2018</i>	
		<i>Version 1.2.31</i>
		Page 276 of 295

<b>CcddApplicationParameterHandler.java</b>	Class that handles retrieval from and storage to the project database of the application scheduling parameter values.
<b>CcddApplicationSchedulerDialog.java</b>	Dialog for assignment of applications to time slots. The dialog is built on the CcddDialogHandler class and implements the CcddSchedulerDialogInterface class.
<b>CcddApplicationSchedulerInput.java</b>	Class for handling application selection in the application scheduler dialog. This class implements the CcddSchedulerInputInterface class.
<b>CcddApplicationSchedulerTableHandler.java</b>	Class for handling CFS application scheduler table output.
<b>CcddAssignmentTreeHandler.java</b>	Class that handles the variable assignment tree in the telemetry scheduler dialog. This class is an extension of the CcddInformationTreeHandler class.
<b>CcddAssignMessageIDDialog.java</b>	Dialog for automatic assignment of message IDs to data tables or telemetry messages. The dialog is built on the CcddDialogHandler class.
<b>CcddBackgroundCommand.java</b>	Class for generically handling execution of code in a background thread.
<b>CcddButtonPanelHandler.java</b>	Generic utility class for creating and handling button panels in the dialogs and frames created within the application.
<b>CcddClassesComponent.java</b>	Collection of common classes used by other CCDD classes. These classes, in general, override existing Java component classes or introduce new ones.
<b>CcddClassesDataTable.java</b>	Collection of common classes used by other CCDD classes. These classes, in general, are used to manipulate and contain information with respect to the data tables.
<b>CcddCommandLineHandler.java</b>	Class for reading and executing the command line options.
<b>CcddCommonTreeHandler.java</b>	Class containing tree handling methods common to all other trees used in the application. This class is an extension of the JTree class.
<b>CcddConstants.java</b>	Class containing constant values used by the other classes.
<b>CcddCopyTableHandler.java</b>	Class for handling copy table operations.
<b>CcddCSVHandler.java</b>	Class for handling import and export of data tables in CSV format. This class implements the CcddImportExportInterface class.
<b>CcddDataTypeEditorDialog.java</b>	Class for handling data type editing. The dialog is built on the CcddDialogHandler class.

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 277 of 295	
---	--	--

<b>CdddDataTypeHandler.java</b>	Class for handling data type operations.
<b>CdddDbCommandHandler.java</b>	Class for handling database commands.
<b>CdddDbControlHandler.java</b>	Class containing the methods for connecting to, creating, copying, renaming, and deleting project databases.
<b>CdddDbManagerDialog.java</b>	Dialog 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 CdddDialogHandler class.
<b>CdddDbTableCommandHandler.java</b>	Class containing the methods for creating, altering, copying, renaming, and deleting the database tables.
<b>CdddDbVerificationHandler.java</b>	Class that executes the database information consistency check.
<b>CdddDialogHandler.java</b>	Generic utility class for creating and handling all of the dialogs created within the application.
<b>CdddDuplicateMsgIDDialog.java</b>	Dialog displaying a table containing duplicate message IDs and their owners. The dialog is built on the CdddDialogHandler class.
<b>CdddEDSHandler.java</b>	Class for handling import and export of data tables in EDS XML format. This class implements the CddImportExportInterface class.
<b>CdddEventLogDialog.java</b>	Class for displaying and updating the session and stored event logs. The dialog is built on the CdddFrameHandler class.
<b>CdddFieldEditorDialog.java</b>	Class for handling data field operations. The dialog is built on the CdddDialogHandler class.
<b>CdddFieldHandler.java</b>	Class for handling the data field editor.
<b>CdddFieldTableEditorDialog.java</b>	Dialog for inspecting and assigning values to data input fields. The dialog is built on the CdddDialogHandler class.
<b>CdddFileIOHandler.java</b>	Class containing file input and output methods (project database backup and restore, table import and export, script storage and retrieval).
<b>CdddFrameHandler.java</b>	Generic utility class for creating and handling all of the frame windows created within the application.
<b>CdddGroupHandler.java</b>	Class for handling table grouping operations.
<b>CdddGroupManagerDialog.java</b>	Dialog for the user to create, alter, or delete table groups. The dialog is built on the CdddDialogHandler class.
<b>CdddGroupTreeHandler.java</b>	Class containing the methods for creating and manipulating a table group tree. This class is an extension of the CddInformationTreeHandler class.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
	<i>Version 1.2.31</i>	Page 278 of 295

<b>CcddImportExportInterface.java</b>	Class that defines the interface for data table import and export classes.
<b>CcddImportSupportHandler.java</b>	Class containing support methods for classes based on the CcddImportExportInterface class. The support methods handle validation and addition of table types and data fields, and for obtaining the user's response to a non-fatal error condition. Classes utilizing these support methods must extend this class.
<b>CcddInformationTreeHandler.java</b>	Generic utility class for manipulating information trees. This class is an extension of the CcddCommonTreeHandler class.
<b>CcddInputFieldPanelHandler.java</b>	Class for creating the table editor panel in which a table, description, and data fields are displayed.
<b>CcddJSONHandler.java</b>	Class for handling import and export of data tables in JSON format. This class implements the CcddImportExportInterface class.
<b>CcddJTableHandler.java</b>	Generic utility class for creating and handling all of the tables created within the application, including the data, type, and field tables. This class is an extension of the JTable class.
<b>CcddKeyboardHandler.java</b>	Class for controlling keyboard input and implementing special key sequence actions.
<b>CcddLinkHandler.java</b>	Class containing methods to manipulate variable linkages.
<b>CcddLinkManagerDialog.java</b>	Dialog 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.java</b>	Class for handling interactions with the variable links for a specific data stream.
<b>CcddLinkTreeHandler.java</b>	Class containing the methods for creating and manipulating a variable link tree. This class is an extension of the CcddInformationTreeHandler class.
<b>CcddMacroEditorDialog.java</b>	Dialog for the user to create, modify, or delete macros and macro values. The dialog is built on the CcddDialogHandler class.
<b>CcddMacroHandler.java</b>	Class for handling macro operations.
<b>CcddMain.java</b>	The CCDD main application class handles flow and execution of the menu bar items.
<b>CcddMathExpressionHandler.java</b>	Class for evaluating simple mathematical expressions.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. JSC-37494 Date: May 2018	
		<i>Version 1.2.31</i>
		Page 279 of 295

<b>CdddMessageIDDialog.java</b>	Dialog displaying all message ID names and associated message ID values. The dialog is built on the CdddDialogHandler class.
<b>CdddMessageIDHandler.java</b>	Class used to determine which message IDs are currently used in tables, data fields, and telemetry messages, and to determine if any references are duplicated.
<b>CdddPaddingAlignmentDialog.java</b>	Dialog displaying a slider control that allows selection of the variable padding byte alignment value. The dialog is built on the CdddDialogHandler class.
<b>CdddPaddingVariableHandler.java</b>	Class that adds/updates or removes the padding variables.
<b>CdddPatchHandler.java</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>CdddPreferencesDialog.java</b>	Class that creates and manages the Preferences dialog used for altering the application's look & feel, fonts, colors, size values, and spacing values. The dialog is built on the CdddDialogHandler class.
<b>CdddProjectFieldDialog.java</b>	Class that creates and manages project-level data fields. The dialog is built on the CdddDialogHandler class.
<b>CdddRateParameterDialog.java</b>	Dialog for assigning the telemetry sample rate parameters. The dialog is built on the CdddDialogHandler class.
<b>CdddRateParameterHandler.java</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>CdddReservedMsgIDEditorDialog.java</b>	Dialog for the user to create, modify, or delete reserved message ID and ID ranges and descriptions. The dialog is built on the CdddDialogHandler class.
<b>CdddReservedMsgIDHandler.java</b>	Class for handling reserved message ID operations.
<b>CdddSchedulerDbIOHandler.java</b>	Class for handling project database input and output operations for the applications and telemetry schedulers.
<b>CdddSchedulerDialogInterface.java</b>	Class that defines the interface for the application and telemetry scheduler dialog classes.
<b>CdddSchedulerEditorHandler.java</b>	Class that handles the Scheduler table within the application (for time slots) and telemetry (for messages) scheduler dialogs.

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide Doc. No. <i>JSC-37494</i> Date: <i>May 2018</i>	
		<i>Version 1.2.31</i>
		Page 280 of 295

<b>CcddSchedulerHandler.java</b>	Class that manages the application and telemetry scheduler dialogs, including transfer of information between the trees and lists.
<b>CcddSchedulerInputInterface.java</b>	Class that defines the interface for application and telemetry scheduler input.
<b>CcddScriptDataAccessHandler.java</b>	Class containing the methods whereby scripts can access the project database information.
<b>CcddScriptDataAccessHandlerStatic.java</b>	Class containing the static method references to the methods in the CcddScriptDataAccessHandler class.
<b>CcddScriptExecutiveDialog.java</b>	Dialog for the user to select script associations to execute. The dialog is built on the CcddDialogHandler class.
<b>CcddScriptHandler.java</b>	Class that handles obtaining the table data and executing the associated script.
<b>CcddScriptManagerDialog.java</b>	Dialog for the user to associate scripts and data tables. The dialog is built on the CcddDialogHandler class.
<b>CcddScriptStorageDialog.java</b>	Dialog 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.java</b>	Dialog for the user to perform text string searches of the project database data tables and stored scripts. The dialog is built on the CcddDialogHandler class.
<b>CcddSearchHandler.java</b>	Class that handles event log, table, and script searches.
<b>CcddServerPropertyDialog.java</b>	Dialog for changing the user name and password, and the PostgreSQL server host and port. The dialog is built on the CcddDialogHandler class.
<b>CcddTableEditorDialog.java</b>	Class for handling data table editing; displays instances of CcddTableEditorHandler. The dialog is built on the CcddEditorPanelHandler class.
<b>CcddTableEditorHandler.java</b>	Class that handles editing of a specific data table. This class is an extension of the CcddInputFieldPanelHandler class.
<b>CcddTableManagerDialog.java</b>	Dialog for the user create, edit, copy, rename, and delete data tables. The dialog is built on the CcddDialogHandler class.
<b>CcddTableTreeHandler.java</b>	Class containing the methods for creating and manipulating a data table tree. This class is an extension of the CcddCommonTreeHandler class.
<b>CcddTableTypeEditorDialog.java</b>	Class for handling table type editing; displays instances of CcddTableTypeEditorHandler. The dialog is built on the CcddEditorPanelHandler class.

Johnson Space Center Engineering Directorate	<b>Core Flight System Command and Data Dictionary Utility User's Guide</b> Doc. No. <i>JSC-37494</i> Version <i>1.2.31</i> Date: <i>May 2018</i> Page 281 of 295	
---	--	--

<b>CdddTableTypeEditorHandler.java</b>	Class that handles the commands associated with a specific table type editor. This class is an extension of the CdddInputFieldPanelHandler class.
<b>CdddTableTypeHandler.java</b>	Class for handling interactions with table types.
<b>CdddTableTypeManagerDialog.java</b>	Dialog for the user to create, edit, copy, rename, and delete table types. The dialog is built on the CdddDialogHandler class.
<b>CdddTelemetrySchedulerDialog.java</b>	Dialog for assignment of variables to telemetry messages. The dialog is built on the CdddDialogHandler class and implements the CdddSchedulerDialogInterface class.
<b>CdddTelemetrySchedulerInput.java</b>	Class for handling variable selection in the telemetry scheduler dialog. This class implements the CdddSchedulerInputInterface class.
<b>CdddUndoHandler.java</b>	Class that manages GUI component undo and redo edit operations.
<b>CdddUndoManager.java</b>	Class that handles undo and redo of edit operations and the edit stack.
<b>CdddUtilities.java</b>	Class containing common utility methods used by other CCDD classes.
<b>CdddVariablesDialog.java</b>	Dialog for the user to view the project's variable (including the each variable's full path), and to display the variables and paths using user-specified separator characters for the data type and variable names, and each variable data type/name pair. The dialog is built on the CdddDialogHandler class.
<b>CdddVariableSizeAndConversionHandler.java</b>	Class for converting variable paths to unique path names. This is used by the script data access methods.
<b>CdddWebDataAccessHandler.java</b>	Class that accepts web access commands and provides JSON formatted output of the requested project data.
<b>CdddWebServer.java</b>	Class that handles set up and management of the embedded Jetty web server.
<b>CdddXTCEHandler.java</b>	Class for handling import and export of data tables in XTCE XML format. This class implements the CdddImportExportInterface class.
<b>Images.java</b>	Dummy class required for the images folder contents to be accessible.

## Appendix E.4. 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

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 282 of 295

(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 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. The tables, with their descriptions and formats, are described below:

<b>Table name:</b>	__app_scheduler						
<b>Description:</b> Contains the information produced by the application scheduler							
<b>Columns:</b>	<table> <tr> <td>time_slot</td> <td>Time slot to which the application belongs in the format &lt;<i>Time Slot #</i>&gt;, where # is the time slot index</td> </tr> <tr> <td>application_info</td> <td>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</td> </tr> </table>	time_slot	Time slot to which the application belongs in the format < <i>Time Slot #</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		
time_slot	Time slot to which the application belongs in the format < <i>Time Slot #</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>Comment:</b>	Application parameter values: maximum number of slots per message, maximum messages per second, maximum messages per cycle, maximum number of commands						
 <b>Table name:</b> __associations							
<b>Description:</b> Contains the script file and data table associations							
<b>Columns:</b>	<table> <tr> <td>description</td> <td>Description of the association; may be blank</td> </tr> <tr> <td>script_file</td> <td>Script file path and file name</td> </tr> <tr> <td>member_tables</td> <td>This column contains the name(s) of the table(s) associated with the script file name. If multiple tables are associated then each is separated by a semi-colon and line feed character. The table names are in the format &lt;<i>root table name</i>&gt;,&lt;<i>level 1 child table's prototype name</i>&gt;.&lt;<i>level 1 child table's variable name</i>&gt;[,&lt;<i>level 2 child table's prototype name</i>&gt;.&lt;<i>level 2 child table's variable name</i>&gt;[,&lt;<i>level 3, etc.</i>&gt;]]. Child tables of an associated table are automatically included when loading the data for script execution</td> </tr> </table>	description	Description of the association; may be blank	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 each is separated by a semi-colon and line feed character. The table names are in the format < <i>root table name</i> >,< <i>level 1 child table's prototype name</i> >.< <i>level 1 child table's variable name</i> >[,< <i>level 2 child table's prototype name</i> >.< <i>level 2 child table's variable name</i> >[,< <i>level 3, etc.</i> >]]. Child tables of an associated table are automatically included when loading the data for script execution
description	Description of the association; may be blank						
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 each is separated by a semi-colon and line feed character. The table names are in the format < <i>root table name</i> >,< <i>level 1 child table's prototype name</i> >.< <i>level 1 child table's variable name</i> >[,< <i>level 2 child table's prototype name</i> >.< <i>level 2 child table's variable name</i> >[,< <i>level 3, etc.</i> >]]. Child tables of an associated table are automatically included when loading the data for script execution						
<b>Comment:</b>	Unused						

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 283 of 295

**Table name:** \_\_data\_types

**Description:** Contains the information for the data type definitions. The table is automatically populated by default data types, which can be altered or deleted

<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

**Comment:** Unused

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 284 of 295

**Table name:** \_\_fields

**Description:** Contains the definitions and values for all of the project's data fields. 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, group, or project

<b>Columns:</b>	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 <root table name>,<level 1 child table's prototype name>.<level 1 child table's variable name>[,<level 2 child table's prototype name>.<level 2 child table's variable name>[,level 3, etc.]]. Default data fields (i.e., those applied to each table of a given table type when created) are denoted by having an owner name in the format <i>Type:&lt;table type name&gt;</i> . Group data fields (i.e., those assigned to a group of data tables) are denoted by having an owner name in the format <i>Group:&lt;group name&gt;</i> . Project data fields (i.e., those assigned at the project level) are denoted by having an owner name <i>Project::</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

**Comment:** Unused

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 285 of 295

**Table name:** \_\_groups

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

**Columns:**

group_name	Group name
member_tables	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 <root table name>,<level 1 child table's prototype name>.<level 1 child table's variable name>[,<level 2 child table's prototype name>.<level 2 child table's variable name>[,level 3, etc.]]

**Comment:** Unused

**Table name:** \_\_links

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

**Columns:**

rate_name	Name of the rate column from which the rate for the variables in this link are taken
link_name	Link name
member_variables	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 <root table name>,<level 1 child table's prototype name>.<level 1 child table's variable name>[,<level 2 child table's prototype name>.<level 2 child table's variable name>[,level 3, etc.]],<data type>.<variable name>

**Comment:** Unused

**Table name:** \_\_macros

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

**Columns:**

macro_name	Macro name
value	Macro value

**Comment:** Unused

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 286 of 295

<b>Table name:</b> __orders
<b>Description:</b> Contains the information for the table column orders, based on user
<b>Columns:</b>
user_name                          User name for which this column ordering applies
table_path                         Path to a table in the format:  <i>&lt;root 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>  <i>root table name</i> is the name of 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 (root table name) followed by structure name and variable name pairs leading to the target child table, separated by commas
column_order                       Contains the column numbers, as defined in the __types table, separated by colons (:), in the order in which the columns are displayed when the user, <i>user_name</i> , is viewing the table specified by <i>table_path</i>
<b>Comment:</b> Unused
<b>Table name:</b> __reserved_msg_ids
<b>Description:</b> Contains the reserved message IDs and ID ranges with their descriptions. By default the range 0x0800 - 0x08FF is reserved for cFE telemetry IDs and the range 0x1800 - 0x18FF is reserved for cFE command IDs (these default values may be altered or deleted)
<b>Columns:</b>
msg_id                             Message ID or ID range in hexadecimal format. ID range values are separated by a hyphen (-)
description                       User-defined text describing the reserved ID or ID range
<b>Comment:</b> Unused
<b>Table name:</b> __script_<script name>
<b>Description:</b> Contains the contents of the script file <script name>
<b>Columns:</b>
line_number                       Script file line number
line_text                         Line of text from the script file
<b>Comment:</b> script name with capitalization intact, script description

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 287 of 295

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

**Description:** Contains the table type definitions for the project's data tables. Structure and command table types are created by default (these can be altered or deleted)

<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
	allow_structure	'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
	allow_pointer	'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

**Comment:** Unused

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 288 of 295

**Table name:** \_\_temp\_table

**Description:** Temporary table created by the CCDD PostgreSQL functions

**Columns:** Varies

**Comment:** Unused

**Table name:** \_\_tlm\_scheduler

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

**Columns:** rate\_name      Rate name

message\_name      Message name in the format <Message #.#> where the first number is the message index and the second is the sub-index for the message

message\_id      Message ID number, in hexadecimal

member\_variable      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 <root table name>,<level 1 child table's prototype name>.<level 1 child table's variable name>[,<level 2 child table's prototype name>.<level 2 child table's variable name>[,level 3, etc.]],<data type>.<variable name>

**Comment:** Rate parameters: maximum seconds per message, maximum messages per second, include uneven rates (true or false), rate 1 column name, rate 1 data stream name, rate 1 maximum messages per cycle, rate 1 maximum bytes per second [, ..., rate n column name, rate n data stream name, rate n maximum messages per cycle, rate n maximum bytes per second]

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 289 of 295

<b>Table name:</b>	<u>values</u>						
<b>Description:</b> Contains the description and individual data table cell values for all of the project's data tables							
<b>Columns:</b>	<table> <tr> <td>table_path</td> <td>Path to a table in the format:   <math>\langle\text{root table name}\rangle,\langle\text{level 1 child table's prototype name}\rangle.\langle\text{level 1 child table's variable name}\rangle[, \langle\text{level 2 child table's prototype name}\rangle.\langle\text{level 2 child table's variable name}\rangle[, \text{level 3, etc.}]]</math>   <math>\text{root table name}</math> is the top-level table name. 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 (root table name) followed by structure name and variable name pairs leading to the target child table, separated by commas           </td></tr> <tr> <td>column_name</td> <td>Column name as seen by the user (versus the database version, which is all lower case and has any spaces replaced with underscores (_))</td></tr> <tr> <td>value</td> <td>If the column_name 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</td></tr> </table>	table_path	Path to a table in the format:  $\langle\text{root table name}\rangle,\langle\text{level 1 child table's prototype name}\rangle.\langle\text{level 1 child table's variable name}\rangle[, \langle\text{level 2 child table's prototype name}\rangle.\langle\text{level 2 child table's variable name}\rangle[, \text{level 3, etc.}]]$  $\text{root table name}$ is the top-level table name. 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 (root table name) followed by structure name and variable name pairs leading to the target child table, separated by commas	column_name	Column name as seen by the user (versus the database version, which is all lower case and has any spaces replaced with underscores (_))	value	If the column_name 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
table_path	Path to a table in the format:  $\langle\text{root table name}\rangle,\langle\text{level 1 child table's prototype name}\rangle.\langle\text{level 1 child table's variable name}\rangle[, \langle\text{level 2 child table's prototype name}\rangle.\langle\text{level 2 child table's variable name}\rangle[, \text{level 3, etc.}]]$  $\text{root table name}$ is the top-level table name. 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 (root table name) followed by structure name and variable name pairs leading to the target child table, separated by commas						
column_name	Column name as seen by the user (versus the database version, which is all lower case and has any spaces replaced with underscores (_))						
value	If the column_name 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						
<b>Comment:</b>	Unused						
<b>Table name:</b> <i>prototype table name</i>							
<b>Description:</b> Contains the cell values for the table							
<b>Columns:</b>	Table type dependent						
<b>Comment:</b>	Table name with capitalization intact, table type name						

## Appendix E.5. PostgreSQL Functions

The CCDD application creates a number of special functions within the project's database to optimize certain table searches and updates. A description of these functions is provided below. The input(s) and output(s) are also described; the 'type' (shown in parentheses) is the PostgreSQL data type.

<b>Function name:</b>	delete_function
<b>Description:</b>	Deletes a function from the project database
<b>Input (type):</b>	function_name (text) Name of the function to delete
<b>Output (type):</b>	none

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 290 of 295

**Function name:** find\_columns\_by\_name

**Description:** Search the data tables in the specified column for non-empty cells. For structure tables include child table references in the search (i.e., references in the \_\_values table).

**Input (type):** column\_name\_user (text) Name of the column as seen by the user

column\_name\_db (text) Name of the column as used in the database

table\_types (text[]) Name(s) of the table types to which the search is constrained

**Output (type):** owner\_name (text) Name of the table containing the search column name

column\_value (text) Value of the table's cell in the search column

**Function name:** find\_prototype\_columns\_by\_name

**Description:** Search the prototype data tables in the specified column for non-empty cells.

**Input (type):** column\_name\_db (text) Name of the column as used in the database

table\_types (text[]) Name(s) of the table types to which the search is constrained

**Output (type):** owner\_name (text) Name of the table containing the search column name

column\_value (text) Value of the table's cell in the search column

**Function name:** get\_def\_columns\_by\_index

**Description:** Search a structure table for the values in the default columns (as used in the database) that define the table as a structure. These columns are data\_type, variable\_name, bit\_length, all rate columns, and all enumeration columns. Return the references sorted by the \_index\_ column, which defines the order of the variables relative to each other.

**Input (type):** name (text) Name of the prototype structure table

**Output (type):** data\_type (text) Data type (structure reference or primitive data type)

variable\_name (text) Variable name

bit\_length (text) Bit length; blank if the variable has no bit length assigned

rate (text) Rate value for each rate column defined in the table's type, in the order the rate columns are assigned in the type definition, separated by commas. The rate value is blank if the variable does not have a value assigned for a given rate

enumeration (text) Enumeration value for each enumeration column defined in the table's type, in the order the enumeration columns are assigned in the type definition, separated by backslashes (\). The enumeration value is blank if the variable does not have a value assigned for a given enumeration

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 291 of 295

**Function name:** get\_def\_columns\_by\_name

<b>Description:</b>	Search a structure table for the values in the default columns (as used in the database) that define the table as a structure. These columns are data_type, variable_name, bit_length, all rate columns, and all enumeration columns. Return the references sorted alphabetically by the variable_name column.	
<b>Input (type):</b>	name (text)	Name of the prototype structure table
<b>Output (type):</b>	data_type (text)	Data type (structure reference or primitive data type)
	variable_name (text)	Variable name
	bit_length (text)	Bit length; blank if the variable has no bit length assigned
	rate (text)	Rate value for each rate column defined in the table's type, in the order the rate columns are assigned in the type definition, separated by commas. The rate value is blank if the variable does not have a value assigned for a given rate
	enumeration (text)	Enumeration value for each enumeration column defined in the table's type, in the order the enumeration columns are assigned in the type definition, separated by backslashes (\). The enumeration value is blank if the variable does not have a value assigned for a given enumeration

**Function name:** get\_table\_members\_by\_index

<b>Description:</b>	Get the table member information for all prototype structure tables, sorted by the table's _index_column, which defines the order of the variables relative to each other. The member information is used primarily to construct the table trees, so it includes the data types (both for structure references and primitives) and variable names for each prototype structure table. Also included are each variable's bit length (if any), rate values (for each rate column in the structure's type definition), and enumerations (for each enumeration column in the structure's type definition).	
<b>Input (type):</b>	none	
<b>Output (type):</b>	tbl_name (text)	Prototype structure table name
	data_type (text)	Data type (structure reference or primitive data type)
	variable_name (text)	Variable name
	bit_length (text)	Bit length; blank if the variable has no bit length assigned
	rate (text)	Rate value for each rate column defined in the table's type, in the order the rate columns are assigned in the type definition, separated by commas. The rate value is blank if the variable does not have a value assigned for a given rate
	enumeration (text)	Enumeration value for each enumeration column defined in the table's type, in the order the enumeration columns are assigned in the type definition, separated by backslashes (\). The enumeration value is blank if the variable does not have a value assigned for a given enumeration

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 292 of 295

**Function name:** get\_table\_members\_by\_name

**Description:** Get the table member information for all prototype structure tables, sorted alphabetically by the variable\_name column. The member information is used primarily to construct the table trees, so it includes the data types (both for structure references and primitives) and variable names for each prototype structure table. Also included are each variable's bit length (if any), rate values (for each rate column in the structure's type definition), and enumerations (for each enumeration column in the structure's type definition).

**Input (type):** none

<b>Output (type):</b>	tbl_name (text)	Prototype structure table name
	data_type (text)	Data type (structure reference or primitive data type)
	variable_name (text)	Variable name
	bit_length (text)	Bit length; blank if the variable has no bit length assigned
	rate (text)	Rate value for each rate column defined in the table's type, in the order the rate columns are assigned in the type definition, separated by commas. The rate value is blank if the variable does not have a value assigned for a given rate
	enumeration (text)	Enumeration value for each enumeration column defined in the table's type, in the order the enumeration columns are assigned in the type definition, separated by backslashes (\). The enumeration value is blank if the variable does not have a value assigned for a given enumeration

**Function name:** reset\_link\_rate

**Description:** Set the rate to 0 for any links in the \_\_links table for links containing no member variables.

**Input (type):** none

**Output (type):** none

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 293 of 295

**Function name:** search\_tables

**Description:** Search the database tables for the specified text string. The search may be modified to ignore case, to use a regular expression in the search string, to search only the data tables (as opposed to including the internal tables), and to limit searching to specific columns in the data tables.

<b>Input (type):</b>	search_text (text)	Text to search for in the tables. Interpreted as a literal string unless the <i>allow_regex</i> flag is true, in which case the search text is considered a regular expression
	no_case (boolean)	true to ignore case when determining matching text; false to preserve the text case
	allow_regex (boolean)	true to interpret the search string as a regular expression; false to treat the search string as literal text
	selected_tables (text)	'ALL' to search all tables (data, internal, and script), 'DATA' to limit the search to data tables and entries in the custom values table, 'PROTO' to limit the search to the data tables only (i.e., prototype tables), or 'SCRIPT' to search only script files
	columns (name[])	one or more column names, separated by commas, to which the match is limited. Blank to include any column
	all_schema (name[])	Database schema(s) in which to limit the search. Defaults to 'public'
<b>Output (type):</b>	search_result (text)	Information on each match found. The text consists of multiple parts separated by the backslash (\) character:
	schema_name	Schema in which the match is found
	table_name	Name of the table
	column_name	Name of the column in the table
	table_description	Comment text for this table which includes the table's name with case preserved and the table's type, separated by a comma
	column_value	Complete contents of the table cell where the match occurs

**Function name:** update\_data\_type\_names

**Description:** Replace all references to a structure table found in every table containing a column with the default database name for the data type ('data\_type') with the new structure table name.

<b>Input (type):</b>	oldtype (text)	Name of the structure data type to replace
	newtype (text)	New name for the structure data type

**Output (type):** none

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. JSC-37494	Version 1.2.31
	Date: May 2018	Page 294 of 295

## Appendix E.6. Known Issues

- Concurrent operation is not supported. Simultaneously interacting with the same project from more than one instance of the CCDD application or via another database access application can result in unexpected results or corruption of the project database.
- 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.

- 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.
- The HTML-formatted table columns cease character wrapping below a certain width (which is dependent on the cell contents), causing the text to appear truncated. Character and word wrapping work as expected when the column width exceeds this 'limit'.
- If 32-bit Java 7 is used in a 64-bit Linux environment then the 32-bit compatibility libraries must be installed. The specific libraries are Linux version dependent. As an example, the user's guide cannot be displayed in 64-bit CentOS 6 using the command menu unless the Gnome 32-bit library, `libgnome.i686`, is installed.
- 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 this type of mouse 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):
  - Control Panel → Mouse**
  - Mouse Properties → Hardware tab**
  - Select the problematic mouse from the list ("HID-compliant mouse")
  - Click the **Properties** button
  - Go to the **Details** tab
  - Select "Device Instance Path" from the combo box
  - 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
  - Open the registry editor and navigate to:  
  
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\<value noted in step 7>\Device Parameters`
  - In Device Parameters, add the following DWORD (32 bit) registry keys:

Johnson Space Center Engineering Directorate	Core Flight System Command and Data Dictionary Utility User's Guide	
	Doc. No. <i>JSC-37494</i>	<i>Version 1.2.31</i>
	Date: <i>May 2018</i>	Page 295 of 295

HScrollHighResolutionDisable = 1

VScrollHighResolutionDisable = 1

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
- 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