TREK

CCSDS FILE DELIVERY PROTOCOL (CFDP) CONSOLE

USER GUIDE

November 2024

Approved for Public Release; Distribution is Unlimited.

TABLE OF CONTENTS

PARAGRAPH PAGE

1 Welcome 1

1.1 Getting Started 1

2 Technical Support 1

3 Introduction 1

4 Overview of the User Interface 8

4.1 Console Menu 8

5 Quick Start Guides 16

5.1 How to Configure the Application 16
5.2 How to Create a CFDP Dropbox 29
5.3 How to Create an Encrypt or Decrypt Dropbox 31
5.4 How to Create a Frag or Defrag Dropbox 32
5.5 How to Turn on Message Logging 34
5.6 How to Turn on Statistics Logging 35
5.7 How to Turn on Metrics Logging 38

6 Details 39

6.1 Configuration 39
6.2 Transaction 40
6.3 Messages and Message Logging 40

7 FAQ and Troubleshooting 40

7.1 Is There an Easy Way to Transfer the Contents of a Directory? 40
7.2 What is class1 and class2? 41
7.3 What is “////”? 41
7.4 Source and Destination Constraints 41
7.5 My File Starts to Transfer and Then Stops 42
7.6 Transfer Results When Item Exists at Destination 42
7.7 Important Things to Know When Using the Get Primitive 42
7.8 How Does Suspend Transactions Work? 43
7.9 CFDP Transactions in an AOS/LOS Environment 43
7.10 How Do I Include My Crypt User Passphrase in the CFDP Console App? 44

TABLES

TABLES PAGE

Table 1 ION CFDP Transmission Parameters 4
Table 2 CFDP Directives 5
Table 3 CFDP Directive Format 5
Table 4 TReK CFDP Configuration File Parameters 29
Table 5 Device Statistics 36
Table 6 Packet Statistics 38
Table 7 CFDP Metrics 39

1 Welcome

The Telescience Resource Kit (TReK) is a suite of software applications and libraries that can be used to monitor and control assets in space or on the ground.

The TReK CFDP console application provides the capability to transfer files using the Consultative Committee for Space Data Systems (CCSDS) File Delivery Protocol (CFDP).

1.1 Getting Started

Start with the Introduction which provides an application overview. Next, try the Quick Start Guides for "How Tos" for common functions. For help with details, reference the Details section. See the FAQ and Troubleshooting section for helpful hints and solutions to the common "gotchas".

2 Technical Support

If you are having trouble installing the TReK software or using any of the TReK software, please contact us for technical assistance:

E-Mail: trek.help@nasa.gov

Messages sent to this address are automatically forwarded to the TReK team.

The HOSC Help Desk (256-544-5066) can provide assistance as needed and is available 24x7.

3 Introduction

The TReK CFDP console application provides the capability to transfer files using the Consultative Committee for Space Data Systems (CCSDS) File Delivery Protocol (CFDP). The application uses the TReK CFDP library to provide CFDP functionality through a menu of console application command line primitives.

The CCSDS File Delivery Protocol (CFDP) was developed by the Consultative Committee for Space Data Systems (CCSDS). Official specifications are contained in a CCSDS document called the CFDP Blue Book (available at www.ccsds.org). The CFDP protocol provides reliable transfer of files from one computer (entity**)** to another, and has been designed to work well over space links. It can be used to perform space to ground, ground to space, and ground to ground file transfers.

CFDP requires a sender and a receiver. The sender and receiver must be configured and running at the same time to perform a file transfer. Each party is referred to as an

"Entity". The sender is an entity and the receiver is an entity. Each "Entity" must have a unique Entity ID. For example:

Figure 1 CFDP Sender and Receiver

To move a file from one computer to another, you will enter one or more CFDP Commands (primitives) to indicate the action to be taken. The syntax of a primitive is as follows:

[CFDP Directive] [Transmission] [source-path] [remote-EID] [destination-path]

The TReK CFDP console application provides support for CFDP over User Datagram Protocol (UDP) and CFDP over Bundle Protocol (BP) using the Jet Propulsion Lab (JPL) Interplanetary Overlay Network (ION) Disruption Tolerant Networking (DTN) software. The CFDP over UDP configuration option is referred to as Native CFDP. The CFDP over BP configuration option is referred to as ION CFDP. In the Native CFDP configuration, the application uses UDP sockets and a Goddard Space Flight Center (GSFC) CFDP library that performs the CFDP work. In the ION CDFP configuration, the TReK CFDP console application communicates with the ION software which performs the CFDP work.

There are differences in capabilities and syntax between the CFDP Native mode and the ION CFDP mode. These differences are summarized below.

Native CFDP Configuration
The syntax of a primitive is as follows:

[CFDP Directive] [Transmission] [source-path] [remote-EID] [destination-path]

Native CFDP Example: put class2 "D:\file1.txt" 2 "/home/kirk/file1.txt"

The TReK CFDP application currently supports multiple CFDP Directives for both Native and ION CFDP modes. These are defined in the Table 2 CFDP Directives.

Valid Transmission values are class1 or class2. Class1 does not guarantee reliable delivery of the file to the destination. Class2 does ensure reliable delivery.

The file you want to transfer is considered the "Source" and the location it should be transferred to is considered the "Destination".

When entering a Source or Destination in a CFDP command line, it must be encapsulated in double quotes.

ION CFDP Configuration

The syntax of a primitive is as follows:

[CFDP Directive] [Transmission] [source-path] [remote-EID] [destination-path]

ION CFDP Example: put //// "D:\file1.txt" 2 "/home/kirk/file1.txt"

The TReK CFDP application currently supports multiple CFDP Directives for both Native and ION CFDP modes. These are defined in the Table 2 CFDP Directives.

The Transmission entry for ION CFDP is composed of five properties. Once these properties are configured, a "////" string is used to indicate that the pre-configured values should be used. The "////" nomenclature is shorthand notation for the following combination of values:

Lifespan/Bundle Protocol Class of Service/Expedited Priority Ordinal/Transmission Mode/Criticality

When the values are empty it indicates that pre-configured values should be used for these properties.

A summary of the Transmission properties is provided in the following table. For details, please reference section 5.1 Table 4.

Property	**Description**
Lifespan	The lifespan is the bundle's "time to live" (TTL) in seconds. The bundle is destroyed if its TTL has expired and it has not reached its destination.
Bundle Protocol Class of Service	The Bundle Protocol Class of Service defines the transmission priority of outbound bundles from three ION priority queues corresponding to bulk, standard, and expedited priorities. The expedited priority queue must be empty before bundles in the standard or bulk queues are serviced by ION. Therefore, bundles with expedited priority should only be sent in critical/emergency situations.
Expedited Priority Ordinal	The expedited priority ordinal is only associated with the expedited priority class of service.
Transmission Mode	The transmission mode defines the reliability of bundle delivery to a destination. The three transmission modes supported are best effort, assured, and assured with custody transfer.
Criticality	A critical bundle is one that has to reach its destination as soon as is physically possible. For this reason, bundles flagged as critical may not

	include custody transfer and require an ION configuration with contact graph routing. In some cases, a critical bundle may be sent over multiple routes to ensure delivery to its final destination. Critical bundles are placed in the expedited priority queue and should only be used in emergency situations.

Table 1 ION CFDP Transmission Parameters

CFDP Directives

The TReK CFDP console application currently supports multiple CFDP directives for both Native and ION CFDP mode and includes put and get directives, filestore directives and message directives. These directives are defined in the following table:

Directive	**Description**
append_file	append a file at the remote entity to another file at the remote entity.
bit_rate	changes the aggregate file transfer bit rate, in real time, for local or remote entities hosting a TReK implementation of Native CFDP. The “affected EID” may be the local entity ID or a remote entity ID. The "bit_rate" primitive is delivered to a remote entity in the form of a TReK CFDP message.
close_rec_file	send a directive to a TReK Record library to close one or all open record files. If the record file name is not included in the primitive, all open record files associated with the TReK Record library are closed. The TReK Record library automatically opens a new record file after it closes a current record file. The “affected EID” may be the local entity ID or a remote entity ID. The "close_rec_file" directive is delivered to a remote entity in the form of a TReK CFDP message.
create_dir	create a directory at the remote entity.
create_file	create an empty file at the remote entity.
delete_file	delete a file at the remote entity.
deny_dir	delete a directory at the remote entity. (Like *remove_dir*, but does not fail if the directory does not exist. Directory must be empty.)
deny_file	delete a file at the remote entity. (Like *delete_file*, but does not fail if the file does not exist)
get	copy file(s) from the remote entity to the local entity. (File cannot be empty.)
message	send a text string to the remote entity.
put	copy file(s) from the local entity to the remote entity. (File cannot be empty.)

remove_dir	delete a directory at the remote entity. (Directory must be empty.)
rename_file	rename a file at the remote entity.
replace_file	replace a file (contents) at the remote entity with another file (contents) at the remote entity.

Table 2 CFDP Directives

Some directives only require a Source. The following table describes what is required for Source and Destination for each directive. Name of file, filename, and name of directory refer to an absolute path.

Directive	Source (first file)	Destination (second file)
append_file	name of file whose contents form first part of new file and name of the new file	name of file whose contents will form second part of new file
bit_rate	aggregate file transfer bit rate	
close_rec_file	name of the TReK record library device	name of the TReK record file to close (if not included all record files associated with the record device are closed)
create_dir	name of directory to be created	
create_file	filename to be created	
delete_file	filename to be deleted	
deny_dir	name of directory to be deleted	
deny_file	filename to be deleted	
get	file or directory name of file(s) to retrieve	file or directory name for file(s) retrieved
message	message	
put	file or directory name of file(s) to send	file or directory name for file(s) sent
remove_dir	name of directory to be deleted	
rename_file	old filename	new filename
replace_file	filename whose contents are to be replaced	filename whose contents will replace the contents of the first filename

Table 3 CFDP Directive Format

When entering a Source or Destination in a CFDP command line, it must be encapsulated in double quotes.

Note: The Get directive is not supported in all ISS CFDP Native and ION implementations. Filestore and message directives are not supported in all ISS CFDP Native implementations. They are supported by TReK when both the sender and receiver

are TReK CFDP implementations (TReK CFDP application, TReK CFDP console application, or TReK CFDP Library). In addition, the bit rate and close rec file directives are not part of the CFDP Blue Book. The bit rate directive is only supported by the TReK CFDP implementation of Native CFDP. The close rec file directive is only supported the by the TReK CFDP implementation of Native and ION CFDP in conjunction with the TReK Record library.

TReK CFDP may be configured to encrypt and decrypt all Native CFDP transactions streams (e.g., "put", "get", "message", "create_file", "delete_file" ...). The Native CFDP stream encryption and decryption capability is implemented by encrypting and decrypting the CFDP Protocol Data Units (PDUs) that are exchanged between the CFDP source and destination platforms. In addition, Native CFDP stream encryption may be configured to encrypt a time stamp, Time To Live (TTL) value and sequence count to provide replay resistance time authentication in support of Safety Conditional KuIP Services (SCKIPS). The encrypted TTL value is in seconds and must be between 0-65535. The time authentication software compares the packet's decrypted time stamp with the operating system time (plus or minus the decrypted TTL) to see if the packet's decrypted time falls within the authentication time window. In addition, the time authentication software does not allow a packet to be processed if the decrypted time and sequence count stays the same or decreases when compared to the previous decrypted packet time and sequence count. Review the “remote entity ID” discussion in the CFDP configuration file description for further information on this Native CFDP encrypt/decrypt configuration option. This option is not available for ION CFDP because TReK CFDP does not have access to ION’s CFDP PDUs.

TReK provides the ability to create CFDP dropboxes to push files to a remote destination. The dropboxes support both Native and ION CFDP modes. CFDP dropboxes are created when the TReK application’s TReK CFDP library reads and processes the TReK CFDP configuration file. A TReK CFDP configuration file CFDP "dropbox" primitive defines a dropbox's operation parameters including where the dropbox is located and the destination of each file placed in the dropbox. A dropbox file is transferred to the dropbox destination immediately after the file is copied to the dropbox assuming a communication path exists between both sides of the transaction. Pre-existing dropbox files are immediately transferred after the creation of the dropbox.

TReK also provides the ability to create encryption and decryption dropboxes to encrypt and decrypt local files. The encrypt and decrypt dropboxes support both Native and ION CFDP modes. Encrypt and decrypt dropboxes are created when the TReK application’s TReK CFDP library reads and processes the TReK CFDP configuration file. A TReK CFDP configuration file encrypt/decrypt "dropbox" primitive defines a dropbox's operation parameters including where the dropbox is located and the directory location of the newly created encrypted or decrypted file. By chaining together encrypt and decrypt dropboxes with a CFDP dropbox, a completely automated encrypt, CFDP file transfer, decrypt chain may be created. This is the only method TReK provides to automate file encryption/decryption using ION CFDP.

TReK includes a CFDP Graphical User Interface (GUI) application and a CFDP library. If you need command line CFDP functionality onboard a spacecraft consider using the TReK CFDP console application. Source code for the console application is provided in the TReK example directory. If you need CFDP functionality without a command line interface, the CFDP destination application may meet your requirements. The source code for the CFDP destination application is also found in the TReK example directory. If you need CFDP functionality on the ground consider using the TReK CFDP GUI application. If you need to include CFDP functionality in your own application, consider using the TReK CFDP library.

TReK Encryption

The TReK encryption library uses OpenSSL's FIPS 140-2 validated cryptographic module and public/private key pairs to encrypt and decrypt files and packets. TReK encryption library support is provided on 32 bit and 64 bit Linux operating systems and 64 bit Windows operating systems. TReK encryption library support is not available on 32 bit Window operating systems. Both the flight platform and ground platform generate public/private key pairs using TReK's "trek_crypt" application. TReK's public key/private key encryption architecture is based on Elliptic Curve Cryptography (ECC) using curve P-256 providing 128-bit security with 128 or 256 bit keys. The cipher packages included with the TReK encryption library are the Advance Encryption Standard (AES) Galois/Counter Mode(AES GCM) and the AES Counter with CBC-MAC (AES CCM) ciphers offering confidentiality, authenticity and integrity. The library supports 128 and 256 bit cipher key sizes and provides AES 128 and 256 bit key-wrap/unwrap functions. Fresh Cipher Encryption Keys (CEK) are created for files and packets using a Password-Based Key Derivation Function 2 (PBKDF2).

TReK's "trek_crypt" application generates the public and private key pair using ECC. The private key is wrapped prior to storing in a file with a default passphrase or an optional user passphrase up to 63 characters in length. If a user passphrase is used to wrap the private key, the passphrase must be provided during runtime. Three methods are available to provide the user passphrase at runtime: enter the passphrase using the TReK CFDP GUI, include the passphrase as a seperate parameter after the path and filename of the CFDP configuration file when launching the TReK CFDP console application or provide the passphrase programmatically using the TReK API. The latter method requires recompilation of the TReK CFDP console application or user application. A shared secred key is generated using the private key and the remote/destination platform's public key referred to as the peer public key. The peer public key (i.e., the public key of the destination platform) must be exchanged manually prior to encryption or decryption, no automated key exchange mechanisms is implemented. The TReK encryption library generates a new CEK for every encrypted file and may be configured to generate a new CEK for every encrypted packet in a packet stream. The TReK encryption library may also be configured to generate a new CEK, for a packet stream, once every "x" seconds to support encryption of high rate packet stream.

No "encryption handshaking" is required between flight and ground hardware during the encryption and decryption of packets.

Files may be encrypted and decrypted using TReK encryption or decryption dropboxes. Native CFDP may be configured to encrypt and decrypt the CFDP packet streams associated with CFDP transactions. The native CFDP stream encryption configuration automatically encrypts and decrypts files as they are being transferred between the source platform and destination platform requiring no encryption or decryption dropbox. The CFDP packet stream encryption option is not available for ION CFDP. ION CFDP must use the encryption and decryption dropboxes to encrypt and decrypt files. Review the description of the CFDP configuration file's remote entity IDs for further information on packet stream encryption and native CFDP.

4 Overview of the User Interface

4.1 Console Menu

The CFDP console application command primitives are described below.

- To put a file on another platform:

  Native CFDP Configuration
  put <class1,class2> <"source pathname"> <destination entity id> <"destination pathname">
  (e.g., put class2 "/home/user/fileA.txt" 2 "/home/user/fileB.txt")

  ION CFDP Configuration
  put <life>/<cos>/<ord>/<mode>/<crit> <"src path"> <dest EID> <"dest path">
  (e.g., put 86400/STD_PRIORITY/0/ASSURED/NOT_CRITICAL "D:/test a" 100 "D:/test b")

  Executes a single "put" transaction by transferring a copy of a file from the local platform to the destination platform.

- To put a directory of files on another platform:

  Native CFDP Configuration
  put <class1,class2> <"source pathname"> <destination entity id> <"destination pathname">
  (e.g., put class2 "/home/user/" 2 "D:/")

  ION CFDP Configuration
  put <life>/<cos>/<ord>/<mode>/<crit> <"src path"> <dest EID> <"dest path">
  (e.g., put 86400/STD_PRIORITY/// "/home/user/" 100 "D:/")

  Executes a "put" transaction for all the files in the specified directory by transferring copies of the files from the local platform to the destination platform.

- To get a file from another platform:

  Native CFDP Configuration

get <class1,class2> <”source pathname”> <source entity id> <”destination pathname”>
(e.g., get class2 “/home/user/fileB.txt” 2 “D:/fileA.txt”)

ION CFDP Configuration
get <life>/<cos>/<ord>/<mode>/<crit> <"src path"> <src EID> <"dest path">
(e.g., get ///ASSURED / "D:/test a" 100 "D:/test b")

Executes a single “get” transaction by transferring a copy of a file from the remote platform to the local platform.

- To get a directory of files from another platform:

Native CFDP Configuration
get <class1,class2> <”source pathname” > < source entity id> <”destination pathname “>
(e.g., get class2 “D:/” 2 “/home/user/”)

ION CFDP Configuration
get <life>/<cos>/<ord>/<mode>/<crit> <"src path"> <src EID> <"dest path">
(e.g., get //// "/home/user/" 100 "D:/")

Executes a “get” transaction for all the files in the specified directory by transferring copies of the files from the remote platform to the local platform.

- To execute a filestore directive:

Native CFDP Configuration
<action> <class1,class2> <”1st path” > < dest EID >
(e.g., create_file class2 “D:/test a” 2)

Or

<action> <class1,class2> <”1st path” > < dest EID ><"2nd path">
(e.g., rename_file class2 "D:/test a" 2 "D:/test b")

ION CFDP Configuration
<action> <life>/<cos>/<ord>/<mode>/<crit> <"1st path"> <dest EID>
(e.g., create_file //// "D:/test a" 100)

Or

<action> <life>/<cos>/<ord>/<mode>/<crit> <"1st path"> <dest EID> <"2nd path">
(e.g., rename_file / "D:/test a" 100 "D:/test b")

Executes a filestore directive on remote platform. For a complete listing of the filestore directives see Section 3 Table 2.

- To send a message to a remote platform:

Native CFDP Configuration

message <class1,class2> <"message" > < dest EID >
(e.g., message class2 "Hello world" 2)

ION CFDP Configuration
message <life>/<cos>/<ord>/<mode>/<crit> <"message"> <dest EID>
(e.g., message / "Hello world" 100)

Sends a message to a remote platform.

- To change the aggregate file transfer bit rate:

Native CFDP Configuration
bit_rate <class1,class2> <aggregate file transfer bit rate > < affected EID >
(e.g., bit_rate class2 5000000 2)

Changes the aggregate file transfer bit rate, in real time, for local or remote entities hosting a TReK implementation of Native CFDP. The "affected EID" may be the local entity ID or a remote entity ID. The "bit_rate" directive is delivered to a remote entity in the form of a TReK CFDP message.

- To close a TReK record file:

Native CFDP Configuration
close_rec_file <class1,class2> <"rec lib dev name" > < affected EID >
(e.g., close_rec_file class2 "record_device1" 2)

Or

close_rec_file <class1,class2> <"rec lib dev name" > < affected EID > <"rec file name">
(e.g., close_rec_file class2 "record_device1" 2 " record_file1")

ION CFDP Configuration
close_rec_file <life>/<cos>/<ord>/<mode>/<crit> <"rec lib dev name" > < EID >
(e.g., close_rec_file //// "record_device1" 2)

Or

close_rec_file <life>/<cos>/<ord>/<mode>/<crit> <"rec lib dev name" > < EID >
<"rec file name">
(e.g., close_rec_file / "record_device1" 2 " record_file1")

Send a directive to a TReK Record library to close one or all open record files. If the record file name is not included in the primitive, all open record files associated with the TReK Record library are closed. The TReK Record library automatically opens a new record file after it closes a current record file. The "affected EID" may be the local entity ID or a remote entity ID. The

"close_rec_file" directive is delivered to a remote entity in the form of a TReK CFDP message. The "close_rec_file" directive is not part of the CCSDS CFDP Blue Book and will only succeed if both sides of the transaction are hosting the TReK CFDP library.

- To add a put primitive to a list:

  Native CFDP Configuration
  add put <class1 or 2> <"source pathname"> <destination EID> <"destination pathname">
  (e.g., add put class2" /home/user/fileA.txt" 2 "D:/fileB.txt")

  ION CFDP Configuration
  add put <life>/<cos>/<ord>/<mode>/<crit> <"src path"> <dest EID> <"dest path">
  (e.g., add put 86400/STD_PRIORITY/0/ASSURED/NOT_CRITICAL "D:/test a" 100 "D:/test b")

  Adds a "put" primitive to a list of "put" primitives. The "put" primitives are not executed until a "send" command is executed.

- To add a get primitive to a list:

  Native CFDP Configuration
  add get <class1 or 2> <"source pathname"> < source EID ><"destination pathname">
  (e.g., add get class2 "/home/user/fileA.txt" 2 "/home/user/fileB.txt")

  ION CFDP Configuration
  add get <life>/<cos>/<ord>/<mode>/<crit> <"src path"> <src EID> <"dest path">
  (e.g., add get //// "D:/test a" 100 "D:/test b")

  Adds a "get" primitive to a list of "get" primitives. The "get" primitives are not executed until a "send" command is executed.

- To add a filestore primitive to a list:

  Native CFDP Configuration
  add <action> <class1,class2> <"1st path" > < dest EID >
  (e.g., add create_file class2 "D:/test a" 2)

  ION CFDP Configuration
  add <action> <life>/<cos>/<ord>/<mode>/<crit> <"1st path"> <dest EID>
  (e.g., add create_file //// "D:/test a" 100)

  Adds a "filestore" primitive to a list of "filestore" primitives. The "filestore" primitives are not executed until a "send" command is executed.

- To add a message primitive to a list:

  Native CFDP Configuration

add message <class1,class2> <"message" > < dest EID >
(e.g., add message class2 "Hello world" 2)

ION CFDP Configuration
add message <life>/<cos>/<ord>/<mode>/<crit> <"message"> <dest EID>
(e.g., add message / "Hello world" 100)

Adds a "message" primitive to a list of "message" primitives. The "message" primitives are not executed until a "send" command is executed.

- To add a bit rate primitive to a list:

Native CFDP Configuration
add bit_rate <class1,class2> <aggregate file transfer bit rate > < affected EID >
(e.g., add bit_rate class2 5000000 2)

Adds a "bit rate" primitive to a list of "message" primitives. The "message" primitives are not executed until a "send" command is executed.

- To add a close rec file primitive to a list:

Native CFDP Configuration
add close_rec_file <class1,class2> <"rec lib dev name" > < affected EID >
(e.g., add close_rec_file class2 "record_device1" 2)

ION CFDP Configuration
add close_rec_file <life>/<cos>/<ord>/<mode>/<crit> <"rec lib dev name" > < EID >
(e.g., add close_rec_file //// "record_device1" 2)

Adds a "close rec file" primitive to a list of "filestore" primitives. The "filestore" primitives are not executed until a "send" command is executed

- To read a file of primitives and add to a list:

process <"primitive pathname">
(e.g., process "D:/toolkit_cfdp_primitives.txt")

Reads a file of primitives and adds them to the appropriate CFDP primitive lists. All valid primitive files must begin with the text string "primitive version X NATIVE_CFDP" or "primitive_version X ION_CFDP" on a single line (the "X" in the text is a version number that may be incremented in future releases). Files that do not contain the primitive version text string are considered invalid and will not be read. You may not mix Native CFDP primitives and ION CFDP primitives in the same primitive file.

- To remove all primitives from a list:

  remove

  Removes all the primitives from the primitive lists.

- To send/execute all primitives in a list:

  send

  Executes all the primitives from the primitive lists.

- To record all primitives in a list:

  record prim <"pathname">
  (e.g., record prim "D:/cfdp_prim.txt")

  Records the primitives from the primitive lists to a file.

- To suspend all CFDP transactions:

  windows os: ctrl-break or ctrl-fn-pause or ctrl-fn-right shift
  linux os: ctrl-c

  Suspends all the CFDP transactions.

- To resume all CFDP transactions:

  resume

  Resumes all the CFDP transactions.

- To cancel a CFDP transaction:

  cancel <transaction id>
  (e.g., cancel 1_1)

  Cancels a CFDP transaction by specifying the transaction ID assigned to the transaction.

- To cancel all CFDP transactions:

  cancel all

  Cancels all the CFDP transactions.

- To report on a CFDP transaction:

  report <transaction id>
  (e.g., report 1_1)

  Displays a status report on CFDP transaction by specifying the ID assigned to the transaction.

- To report on all CFDP transactions:

  report all
  or
  r

  Displays a status report on all the CFDP transactions.

- To display progress messages:

  prog

  Display progress messages on all the CFDP transactions.

- To stop displaying progress messages:

  stop prog

  To stop displaying progress messages on all the CFDP transactions.

- To log messages:

  log <"pathname"> <log debug messages (true or false)>
  (e.g., log "D:/log.txt" false)

  Logs CFDP transaction messages to a file. Debug messages may also be included in the log file for more detailed information about the transaction.

- To stop logging messages:

  stop log

  Stops logging CFDP transaction messages to file, closes the file and appends a GMT time stamp to the name of the file.

- To record statistics snapshot:

  stat <"pathname">
  (e.g., stat "D:/statistics.csv")

Records a snapshot of device statistics once a second and includes current statistics information on all packets that are being received or sent by the device.

- To stop recording statistics snapshot:

  stop stat

  Stops recording a snapshot of device statistics to a file, closes the file and appends to a GMT time stamp the name of the file.

- To reset statistics:

  reset stat

  Resets the device statistics information for all devices to zero.

- To record CFDP metrics snapshot:

  metric <"pathname">
  (e.g., metric "D:/metrics.csv")

  Records a snapshot of CFDP transaction metrics once a second and includes the completion status and transaction time of each CFDP transaction.

- To stop recording CFDP metrics snapshot:

  stop metric

  Stops recording a snapshot of CFDP transaction metrics to a file, closes the file and appends a GMT time stamp to the name of the file.

- To reset CFDP metrics:

  reset metric

  Resets all CFDP metrics information to zero.

- To reconfigure the CFDP console application:

  reconfig <"pathname">
  (e.g., reconfig "D:/cfdp_config.txt")

  Reconfigures the CFDP console application by cancelling all the current CFDP transactions and configuring the application with the new configuration file.

- To save the CFDP console configuration:

  save <"pathname">
  (e.g., save "D:/cfdp_config.txt")

  Saves the CFDP console configuration parameters to a file. This includes all "put" and "get" primitives in the "put" and "get" lists.

- To display the CFDP console configuration:

  display config

  Displays the list of CFDP console configuration parameters.

- To display the CFDP console command primitives:

  help

  Displays the list of CFDP console command primitives.

- To exit application:

  exit or quit or q

  Exits the CFDP console application.

5 Quick Start Guides

This section provides "How Tos" for common functions.

5.1 How to Configure the Application

When launching the CFDP console application, include the path and filename of a TReK CFDP configuration file. If no path and filename are provided in the command line, the application attempts to open a configuration file with the default path and filename equal to "./toolkit_cfdp_config.txt". If the CFDP console application is configured to perform encryption/decryption and the private key was wrapped/encrypted with a user passphrase, the user passphrase may be included in the command line after the configuration file path name by encapsulating the passphrase in double quotes. For example:

- trek_cfdp_console.exe "D:/tookit_cfpd_config.txt" "passphrase"

If the CFDP console application default path and filename of the configuration file is not appropriate and the console application is not configured to perform encryption/decryption, simple include the configuration file path and filename, in double quotes, after the console application executable. For example:

- trek_cfdp_console.exe “D:/my_cfpd_config.txt”

The format of a configuration file is a series of name value pairs that configure the CFDP console application to meet user requirements. One or more spaces separate individual parameters on each line in the file. Table 4 identifies and describes the individual configuration file parameters. The third column identifies the device mode that each parameter supports (the CFDP console application does not simultaneously support both Native CFDP and ION CFDP).

CFDP Configuration File Parameter	Description	Device Mode
CFDP_configuration_version	The configuration file version number. The first parameter in the configuration file must be the version number or TReK CFDP initialization will fail.	NATIVE_CFDP ION_CFDP
cfdp_library_device_mode	A unique reference that may be used to communicate with other TReK library devices.	NATIVE_CFDP ION_CFDP
trek_device_mode	The TReK device mode parameter is set to NATIVE_CFDP if the TReK CFDP library is communicating with GSFC's CFDP library or to ION_CFDP if TReK is communicating with JPL's CFDP library.	NATIVE_CFDP ION_CFDP
log_messages_in_file	The log messages in file boolean controls message logging. If true, messages are recorded in a log file. The default value is false.	NATIVE_CFDP ION_CFDP
log_debug_messages	The log debug messages boolean controls logging debug messages. If true, debug messages are recorded in a log file. The default value is false.	NATIVE_CFDP ION_CFDP
log_file_path	The log file path is the absolute path to the directory where the log file should be written. If an empty string is provided, the default path is the user’s home directory.	NATIVE_CFDP ION_CFDP
log_file_name	The log file name is the name to use for the log file. The default value "toolkit_cfdp_log.txt".	NATIVE_CFDP ION_CFDP
record_stat_snapshot_in_file	The record stat snapshot in file boolean controls recording statistics. If "true", a statistic snapshot is recorded in a file. The default value is false.	NATIVE_CFDP ION_CFDP
record_packet_statistics	The record packet statistics boolean controls recording packet statistics in addition to device statistics. If "true", packet statistics are recorded in a file. The default value is false.	NATIVE_CFDP ION_CFDP

record_stat_file_path	The record stat file path is the absolute path to the directory where the statistics file should be recorded. If an empty string is provided, the default path is the user's home directory.	NATIVE_CFDP ION_CFDP
record_stat_file_name	The record stat file name is the name to use for the statistics file. The default value is "toolkit_cfdp_statistics.csv".	NATIVE_CFDP ION_CFDP
record_cfdp_metrics_snapshot_in_file	The record CFDP metrics snapshot in file boolean controls recording CFDP metrics. If "true", a CFDP metric snapshot is recorded in a file. The default value is false.	NATIVE_CFDP ION_CFDP
record_cfdp_metrics_file_path	The record CFDP metrics file path is the absolute path to the directory where the CFDP metrics file should be recorded. If an empty string is provided, the default path is the user's home directory.	NATIVE_CFDP ION_CFDP
record_cfdp_metrics_file_name	The record CFDP metrics file name is the name to use for the CFDP metrics file. The default value is "toolkit_cfdp_metrics.csv".	NATIVE_CFDP ION_CFDP
support_cfdp_status_requests	The support cfdp status requests boolean enables monitoring the status of CFDP transactions by a user application. If "true", CFDP transaction monitoring is enabled. The default value is false.	NATIVE_CFDP ION_CFDP
public_key_path_and_file_name	The public key path and filename is the absolute path and file name of the local entity's public key file. It is used to encrypt and decrypt files and CFDP PDU packets. The public key file is created by TReK's "trek_crypt" program.	NATIVE_CFDP ION_CFDP
private_key_path_and_file_name	The private key path and filename is the absolute path and file name of the local entity's private key file. It is used to encrypt and decrypt files and CFDP PDUs packets. The private key file is created by TReK's "trek_crypt" program.	NATIVE_CFDP ION_CFDP
packet_encryption_key_time_interval	The packet encryption key time interval determines how often the packet encryption key is changed while encrypting a stream of native CFDP PDU packets. The time interval is measured in seconds. If the packet encryption key time interval is set to zero, the TReK encryption library will generate a new packet encryption key for every packet in the stream. The TReK encryption library can support the encryption of high rate packet	NATIVE_CFDP

	streams by setting the packet encryption key time interval to a non-zero value. The default value is 10 seconds.	
cipher_class	The cipher class is the cipher package that the TReK encryption library will use to encrypt and decrypt files and streams of native CFDP PDU packets. The four cipher class values are AES_128_GCM, AES_256_GCM, AES_128_CCM and AES_256_CCM which support either a 128 bit or 256 bit symmetric key. An AES 256 cipher will require more CPU resources to encrypt and decrypt files and streams then an AES 128 cipher.	NATIVE_CFDP ION_CFDP
Primitives	Creates and configures TReK dropboxes using "dropbox" primitives and/or initializes the list of CFDP primitives using TReK CFDP primitives (e.g., "put", "get", "message", "create_file", "delete_file" ...). An additional CFDP library function (SendAllRequests) must be called prior to processing the list of CFDP primitives. The default primitive list is empty.	NATIVE_CFDP ION_CFDP
ack_timeout	The CFDP library sends positive acknowledgment on reception of the end-of-file packet and finished packet. This timeout defines the number of seconds the CFDP library will wait for the ACK packet to arrive prior to retransmitting the end-of-file or finished packet. Minimum value is 1, maximum value is 2,147,483,647 and the default value is 15 seconds.	NATIVE_CFDP
ack_limit	The ACK limit is the number of ack timeouts that may occur prior to cancelling the CFDP transaction. Minimum value is 1, maximum value is 2,147,483,647 and the default value is 25.	NATIVE_CFDP
nak_timeout	The CFDP library sends a NAK packet identifying the CFDP packets that were not received by the CFDP library. This timeout defines the number of seconds the CFDP library will wait for the retransmission of the requested CFDP packets. Minimum value is 1, maximum value is 2,147,483,647 and the default value is 15 seconds.	NATIVE_CFDP

nak_limit	The NAK limit is the number of Nak timeouts that may occur prior to cancelling the CFDP transaction. Minimum value is 1, maximum value is 2147483647 and the default value is 100.	NATIVE_CFDP
nak_max_pdu_size	The NAK maximum PDU size is the maximum size of a NAK PDU packet in bytes. If the CFDP transactions identifies multiple gaps in the data transmission and the gap information cannot fit in a single NAK PDU, the CFDP library will generate multiple NAK PDUs until all the data gaps are filled or the number of NAK PDU's surpasses the NAK limit. Providing a limit on the maximum size of the NAK PDU ensures the PDU will not be dropped because its size exceeds the limits of its transport medium. Minimum value is 50, maximum value is 64042 and the default value is 16000.	NATIVE_CFDP
inactivity_timeout	The inactivity timeout is the length of time, in seconds, the CFDP library is required to wait between CFDP packet receptions prior to cancelling the CFDP transaction. Minimum value is 1, maximum value is 2,147,483,647 and the default value is 300 seconds.	NATIVE_CFDP
outgoing_file_chunk_size	The outgoing file chunk size is the maximum size, in bytes, of the data zone of the CFDP packets created by the CFDP library. Minimum value is 1, maximum value is 65,200 and the default value is 1,300 bytes.	NATIVE_CFDP
aggregate_file_transfer_bit_rate	The aggregate file transfer rate represents the maximum transmission rate, in bits per second, of the CFDP packets created by the CFDP library. Minimum value is 1, maximum value is 2,147,483,647 and the default value is 10,000,000 bits/second.	NATIVE_CFDP
socket_queue_size	The UDP socket that is created to receive CFDP packets may store CFDP packets in a queue prior to the packets being processed by the CFDP library. This queue minimizes the chances of a CFDP packet being dropped due to packet transmission bursts or a temporary CPU spike on the receiving platform. In general, a larger queue size is needed for higher transmission rates. If an unacceptable number of CFDP packet retransmissions is	NATIVE_CFDP

	occurring, increasing the queue size or decreasing the file transfer rate may help decrease or eliminate the CFDP packet retransmissions. Minimum value is 0, maximum value is 1,000,000 and the default value is 1000.	
transaction_cycle_time_interval	The transaction cycle time interval, in milliseconds, controls the processing rate of CFDP library transactions. Minimizing the cycle time, increases the transaction speed or processing rate. The default value is 1 milliseconds. The minimum value is 0 millisecond and the maximum value is 2,147,483,647 milliseconds. This value should only be incremented if CPU usage on the host platform is unexpectedly high while idling or while processing a transaction.	NATIVE_CFDP
steps_per_transaction_cycle	The step per transaction cycle defines how many steps or transaction cycles are performed prior to delaying the prescribed transaction cycle time. Increasing the steps per transaction cycle, increases the transaction speed or processing rate. The default value is 10. The minimum value is 1 and the maximum value is 2,147,483,647. This value should be incremented if the CFDP library is not able to achieve the aggregate file transfer rate. This value should be decremented if CPU usage on the host platform is unexpectedly high while idling or while processing a transaction.	NATIVE_CFDP
class_of_service	The class of service defines the CFDP level of service for the file transfer. The two CFDP levels of service are class1 and class2. CFDP class1 service is a "send and forget" level of service that sends files without any acknowledgement of their receipt by the recipient. CFDP class2 service requires file delivery acknowledgements in the form of ACKs and NAKs from the recipient. The default value is class2.	NATIVE_CFDP
auto_resize_nak_max_pdu_size	The auto_resize_nak_max_pdu_size boolean enables the automatic detection of dropped NAK PDUs and the resizing of the NAK maximum PDU packet size. The auto resize capability is advantageous when a CFDP	NATIVE_CFDP

	transaction experiences a very large number of dropped packets or data gaps and the size of the associated NAK PDU packet that is identifying the gaps exceeds the capability of the transport medium (i.e., the NAK PDU packet is heavily fragmented and cannot successfully reach its destination). Under these conditions, CFDP's best option is to reduce the size of NAK PDU by splitting it up across multiple NAK PDUs. The CFDP auto resize code attempts to determine an acceptable value for the NAK maximum PDU size given the limitations of the transport medium and the understanding that the CFDP library's best performance is achieved by setting the NAK maximum PDU size to the largest value supported by the transport medium. If auto resize of the NAK maximum PDU size has been enabled and a CFDP transaction determines no NAK PDU packet has been successfully transmitted to its destination, the auto resize code automatically halves the NAK maximum PDU size and retransmits the PDU. The auto resize code continues to halve the NAK maximum PDU size and retransmit a smaller NAK PDU until the NAK PDU reaches its destination or the NAK maximum PDU size drops below 1280 bytes. Once a NAK PDU packet is successfully transmitted to its destination, the auto resize code is no longer exercised for that transaction and subsequent dropped NAK PDUs are not resized. Each new CFDP transaction initiates a new instance of the auto resize code. Therefore, the value for the maximum NAK PDU size may be different with different CFDP transactions. The auto resize code initialize the maximum NAK PDU size to the value provided by the nak_max_pdu_size CFDP parameter. By resetting the nak_max_pdu_size CFDP parameter in the CDFP configuration file or CFDP GUI to the value determined by the auto resize code, the CFDP application may avoid dropping NAK PDU packets that are too	

	large to be supported by the transport medium. The default value is true.	
auto_suspend_and_resume	The auto suspend and resume boolean enables the automatic suspension or resumption of all CFDP transactions associated with a remote entity ID when a network connection to that remote entity ID has been lost or found. The TReK CFDP library creates a UPD socket that sends and receives four byte packets to confirm network connectivity. This capability may be used to detect Acquisition Of Signal (AOS) and Loss Of Signal (LOS) events enabling native CFDP to successfully transfer files across multiple AOS/LOS windows without manual intervention. Auto suspend and resume is only supported by the TReK CFDP library. Therefore, the TReK CFDP library software must be running on both the local and remote nodes. The default value is false.	NATIVE_CFDP
auto_suspend_and_resume_mode	The auto suspend and resume mode identifies the auto suspend and resume relationship between the local node and the remote nodes. The three auto suspend and resume mode parameter values are PEER_TO_PEER_MODE, CLIENT_OR_GROUND_MODE and SERVER_OR_FLIGHT_MODE. A peer to peer configuration allows all peer to peer nodes to perform CFDP transactions with each other. A client server configuration restricts CFDP transactions. A client may only perform CFDP transactions with a server and a server may only perform CFDP transactions with a client. A client may perform CFDP transactions with multiple server nodes and a server may perform CFDP transactions with multiple client nodes. The four byte connectivity packet is always being transmitted by all nodes in a peer to peer configuration regardless of network connectivity or AOS/LOS periods. In a client/ground and server/flight configuration the connectivity packet is always being transmitted by the client/ground node but the server/flight node only transmits the	NATIVE_CFDP

	connectivity packet over a confirmed network connection during AOS periods. The default value is PEER_TO_PEER_MODE.	
auto_suspend_and_resume_port	The auto suspend and resume port is used to create the UDP socket that sends and receives the four byte connectivity packet. The default port value is 45600 (minimum value 0 and maximum value 65535).	NATIVE_CFDP
auto_suspend_and_resume_connection_timeout	The auto suspend and resume connection timeout value is the length of time, in seconds, that must pass between the receipt of connectivity packets before a connection between two entity IDs or nodes is declared lost. Connectivity packets are sent once every half second. Minimum value is 1, maximum value is 2,147,483,647 and the default value is 5.	NATIVE_CFDP
local_entity_id local_ip_address local_port	The pre-assigned local entity ID integer value and its associated local IP address and local port. Only one local EID entry is supported by the CFDP library. The default local_ip_address value is 127.0.0.1. The default local_port value is 4560 (minimum value 0 and maximum value 65535).	NATIVE_CFDP
remote_entity_id remote_ip_address remote_port	The pre-assigned remote entity ID integer value and its associated remote IP address and remote port. Multiple remote entity ID entries are supported by the CFDP library.	NATIVE_CFDP
remote_entity_id remote_ip_address remote_port peer_pub_key_path_and_file_name	The pre-assigned remote entity ID integer value and its associated remote IP address, remote port and the absolute path and name of the file containing the peer public key to encrypt and decrypt the native CFDP PDU packets. The peer public key is the public key of the remote/destination platform and is created by TReK's "trek_crypt" program. A peer public key path and file name must be provided to enable encryption and decryption of all CFDP transactions with the remote entity. Multiple remote entity ID entries are supported by the CFDP library.	NATIVE_CFDP
remote_entity_id remote_ip_address remote_port peer_pub_key_path_and_file_name time_to_live	The pre-assigned remote entity ID integer value and its associated remote IP address, remote port and the absolute path and name of the file containing the peer public key to encrypt and decrypt the native CFDP PDU	NATIVE_CFDP

	packets. The peer public key is the public key of the remote/destination platform and is created by TReK's "trek_crypt" program. A peer public key path and file name must be provided to enable encryption and decryption of all CFDP transactions with the remote entity. If the encrypted CFDP session requires an encrypted timestamp, sequence count and time to live value to provide replay resistance time authentication in support of SCKIPS, include the time to live value, in seconds, after the peer public key path and file name. Multiple remote entity ID entries are supported by the CFDP library. Minimum TTL value is 0, maximum TTL value is 65535 and the default TTL value is 60.	
lifespan	The lifespan is the bundle's "time to live" (TTL) in seconds. The bundle is destroyed if its TTL has expired and it has not reached its destination. Minimum value is 1, maximum value is 2,147,483,647 and the default value is 86400.	ION_CFDP
bp_class_of_service	The BP class of service defines the transmission priority of outbound bundles from three ION priority queues corresponding to bulk, standard and expedited priorities. The three BP class of service parameter values are BULK_PRIORITY, STD_PRIORITY and EXPEDITED_PRIORITY. The expedited priority queue must be empty before bundles in the standard or bulk queues are serviced by ION. Therefore, bundles with EXPEDITED_PRIORITY should only be sent in critical/emergency situations. The default value is STD_PRIORITY.	ION_CFDP
expedited_priority_ordinal	The expedited priority ordinal is only associated with the EXPEDITED_PRIORITY class of service. Ordinal values range from 0 (lowest priority) to 254 (highest priority). The default value is 0.	ION_CFDP
transmission_mode	The transmission mode defines the reliability of bundle delivery to a destination. The three transmission mode parameter values are BEST_EFFORT, ASSURED and ASSURED_WITH_CUSTODY_TRANSFER. BEST_EFFORT relies upon the underlying	ION_CFDP

	convergence-layer protocol (e.g., Transmission Control Protocol or TCP) to retransmit missing bundles. ASSURED is a step up in reliability and includes BP support in detecting a lost TCP connection and re-forwarding of bundles assumed aborted by the convergence-layer protocol failure. ASSURED_WITH_CUSTODY_TRANSFER requires the reception, by the sending node, of a custody acceptance or refusal signal (packaged in a bundle) from the receiving node. The default value is ASSURED.	
criticality	A critical bundle is one that has to reach its destination as soon as is physically possible. For this reason, bundles flagged as critical may not include custody transfer and require an ION configuration with contact graph routing. In some cases, a critical bundle may be sent over multiple routes to ensure delivery to its final destination. Critical bundles are placed in the expedited priority queue and should only be used in emergency situations. The two criticality parameters are NOT_CRITICAL and CRITICAL. The default value is NOT_CRITICAL.	ION_CFDP
support_transaction_result_message	The support transaction result boolean enables the generation and transmission of a CFDP transaction result message to the source node. If the source node receives the transaction result message within a designated time window, it will update its transaction status with the transaction result (e.g., success or fail). If the support transaction result boolean is set to "true" and the result message is not received within a designated time window, the source node's transaction status is set to "unknown". If this boolean is set to "false" and the source node did not experience any problems while transmitting the CFDP transaction request, the final transaction status is set to "finished". This capability is only supported by the TReK CFDP library. Therefore, the TReK CFDP library software must be running on both the source and destination nodes. The default value is true.	ION_CFDP

transaction_result_message_timeout	The transaction result message timeout is the length of time, in seconds, the TReK CFDP library will wait for a transaction result message prior to setting the final status of the transaction to "unknown". Choosing the proper transaction result message timeout is problematic. ION CFDP processes CFDP transactions sequentially so careful considerations must be made when setting this value. If a large number of files are being uplinked and downlinked simultaneously, a larger timeout value may be necessary. In addition, the timeout value should include LOS windows if the file transfer will span LOS periods (the result message timer is not paused during an LOS). Too small a value will unnecessarily set the final status of a transaction to "unknown", too large a value will introduce an unnecessary wait prior to setting the final status of the transaction to "unknown" if a final status message is never received. It is best to choose too large a value versus too small a value. Minimum value is 1, maximum value is 2,147,483,647 and the default value is 300.	ION_CFDP
add_tmp_cfdp_filename_extension	The "tmp_cfdp" file name extension boolean should be set to "true" if ION is transferring one or more files to a TReK dropbox directory. The most common scenario is if an encryption dropbox has been created and configured to place an encrypted file in an ION CFDP dropbox. If the ION CFDP dropbox is configured to transfer the encrypted file to a decryption dropbox on the destination platform, the "tmp_cfdp" file name extension boolean must be "true" on the source or sending platform to properly decrypt the file on the destination platform. If this boolean is set to "true", a temporary file name is created for all file transfers by adding a ".tmp_cfdp" file name extension to the original file name on the destination platform. Upon successful completion of the file transfer, the ".tmp_cfdp" extension is removed from the file name on the destination platform. If the destination is a dropbox, the dropbox	ION_CFDP

	will use the file name to determine both when the file has completed its transfer and when the file may safely be decrypted by the dropbox. If this boolean is set to "false", no temporary file name is used during the file transfer and ION "put" transfers to a TReK dropbox directory are not supported. Native CFDP uses a temporary "tmp_cfdp" file name for all file transfers and does not require this boolean flag when transferring files to a dropbox directory. Setting the "tmp_cfdp" file name extension boolean to "true" even if a file transfer destination is not a TReK dropbox directory is supported and does not impact performance. However, the destination platform must be hosting TReK version 5.2.0 or higher to properly remove the ".tmp_cfdp" file name extension. The default value is true.	
display_console_menu	The display console menu boolean controls displaying the console command primitive menu during startup of the console application. The default value is true.	NATIVE_CFDP ION_CFDP
display_error_messages	The display error messages boolean controls displaying error messages by the CFDP GUI and console applications. If "true", error messages are displayed by the CFDP GUI or console applications. The default value is true.	NATIVE_CFDP ION_CFDP
display_warning_messages	The display warning messages boolean controls displaying warning messages by the CFDP GUI and console applications. If "true", warning messages are displayed by the CFDP GUI and console applications. The default value is false.	NATIVE_CFDP ION_CFDP
display_info_messages	The display info messages boolean controls displaying information messages by the CFDP GUI and console applications. If "true", information messages are displayed by the CFDP GUI and console applications. The default value is true.	NATIVE_CFDP ION_CFDP
display_progress_messages	The display progress messages boolean controls displaying progress messages by the CFDP GUI and console applications. If "true", progress messages are displayed by the CFDP GUI and console applications. The default value is false.	NATIVE_CFDP ION_CFDP

display_debug_messages	The display debug messages boolean controls displaying debug messages by the CFDP GUI and console applications. If "true", debug messages are displayed by the CFDP GUI applications. The default value is false.	NATIVE_CFDP ION_CFDP
default_remote_entity_id	The default remote entity ID is used by the CFDP GUI application to save a default value for the remote EID. The default value is blank.	NATIVE_CFDP ION_CFDP
default_destination_command_line	The default destination command line is used by the CFDP GUI application to save a selected default command line destination path from the list of default destination paths. The default value is blank.	NATIVE_CFDP ION_CFDP
default_destination_command_list	The default destination command list is used by the CFDP GUI application to save a selected default command list destination path from the list of default destination paths. The default value is blank.	NATIVE_CFDP ION_CFDP
default_destination_path	The default destination path is used by the CFDP GUI application to save the list of default destination paths. The default value is blank.	NATIVE_CFDP ION_CFDP
gui_command_line_primitive	Used by the CFDP GUI application to save the command line primitive. The default value is blank.	NATIVE_CFDP ION_CFDP

Table 4 TReK CFDP Configuration File Parameters

5.2 How to Create a CFDP Dropbox

This section describes how to create a CFDP dropbox. The CFDP console application may be configured to create a CFDP dropbox by including a “dropbox” primitive in the TReK CFDP configuration file. One or more “dropbox” primitives may be added to the “Dropbox CFDP Primitives” section of the configuration file. The “dropbox” primitive is not supported by the CFDP console application’s command line interface.

Acceptable formats of the CFDP "dropbox" primitive string for Native CFDP are as follows:

- dropbox <class1/class2> <"dropbox path"> <dest EID> <"dest path"> <retry limit> <"successful transaction path">
  (e.g., dropbox class2 "D:/db_dest1/" 200 "D:/dest1/" 1 "D:/success/")

- dropbox <class1/class2> <”dropbox path"> <dest EID> <"dest path"> <retry limit> <"">
  (e.g., dropbox class2 "/home/user/dropbox_dest1" 200 "/home/user/dest1" 1 "")

The dropbox primitive for Native CFDP includes class 1 or class 2 service, the dropbox path, the destination entity ID, the destination path, the retry limit and the successful transaction path. The retry limit defines the number of additional attempts at transferring a file before declaring the transaction unsuccessful. The successful transaction path is the path to a directory, on the dropbox source platform, where successfully transferred files are stored upon completion of a transaction. If the successful transaction path is empty, as shown in the second example, the dropbox will delete the file, on the source platform, if the file is successfully transferred to its destination. For class 1 service, files are simply moved or deleted from the dropbox directory when the transaction has completed the number of retry attempts.

Acceptable formats of the CFDP "dropbox" primitive string for ION CFDP are as follows:

- dropbox <life>/<cos>/<ord>/<mode>/<crit> <"dropbox path"> <dest EID> <"dest path"> <retry limit> <" successful transaction path ">
  (e.g., dropbox 86400/STD_PRIORITY/0/ASSURED/NOT_CRITICAL "/home/user/dropbox_dest1/" 200 "/home/user/dest1/" 0 "/home/user/success/")

- dropbox //// <"dropbox path"> <dest ID> <"dest path"> <retry limit> <"">
  (e.g., dropbox ///ASSURED/ "/home/user/db_dest1/" 200 "/home/user/dest1/" 0 "")

- dropbox / <"dropbox path"> <dest EID> <"dest path"> <retry limit> <"">
  (e.g., dropbox / "/home/user/dropbox_dest1" 100 "D:/dropbox_dest1" 0 "")

The CFDP dropbox primitive for ION CFDP is identical to Native CFDP except "class1" or "class2" is replaced by the ION CFDP parameter values in the TReK CFDP configuration file (TTL, priority, mode and criticality) if the values were not specified in the primitive string. Another important distinction between ION and Native CFDP dropboxes is associated with the retry limit. An ION CFDP dropbox ignores the retry limit value in the dropbox primitive and resets the value to zero in the TReK CFDP library. There are two important reasons why the ION CFDP dropbox does not attempt to retransmit failed CFDP transactions:

1. ION CFDP uses the original filename when populating the destination file. Any attempt to retransmit a file must also include a "delete" filestore directive to ensure no file with the original filename exists at the destination.
2. If transaction result messages are being processed and an incorrect "transaction_result_message_timeout" is chosen, a successfully transferred file will be incorrectly deleted.

If transaction result messages are not being processed by the TReK ION CFDP library (i.e., class 1 service), files are simply moved or deleted from the dropbox directory when the transaction has completed the transfer from the dropbox.

Dropbox files are renamed with a ".dropbox" extension while they are being processed by the dropbox. If a dropbox fails to successfully transfer a file to the destination directory, a class 2 Native CFDP dropbox will initiate additional transfer attempts up to the "retry limit" designated in the dropbox primitive. A class 1 Native CFDP dropbox will blindly repeat the file transfer up to the "retry limit". If the final status message of a file transaction identifies an unsuccessful file transfer, the file is renamed with an ".unsuccessful" extension. If the CFDP library fails to receive the final status of a file transaction, the file is renamed with an ".unknown" extension. If an error occurred during the file transfer, the file is renamed with a ".droperror" extension. Only successfully transferred files are moved or deleted from the dropbox directory.

5.3 How to Create an Encrypt or Decrypt Dropbox

This section describes how to create an encrypt or decrypt dropbox. The CFDP console application may be configured to create an encrypt or decrypt dropbox by including a dropbox primitive in the TReK CFDP configuration file. One or more dropbox primitives may be added to the “Dropbox Encrypt or Decrypt Primitives” section of the configuration file. The dropbox primitive is not supported by the CFDP console application’s command line interface. A TReK CFDP configuration file dropbox primitive defines an encrypt or decrypt dropbox's operation parameters including where the dropbox is located and the local destination directory of each newly created encrypted or decrypted file. Encrypt and decrypt dropboxes are created during initialization of the TReK CFDP library when InitToolkitCfdp function reads the TReK CFDP configuration file. An encrypt or decrypt dropbox file is encrypted or decrypted prior to being transferred to a local destination directory on the dropbox platform. Pre-existing dropbox files are immediately encrypted or decrypted after the creation of the dropbox. If the local destination directory of an encrypt dropbox is a CFDP dropbox, the encrypted file will automatically be transferred to the CFDP dropbox's remote destination directory. If the CFDP dropbox's remote destination directory is a decrypt dropbox the encrypted file will automatically be decrypted and placed in the decrypt dropbox's destination directory. By chaining together encrypt and decrypt dropboxes with a CFDP dropbox, a completely automated encrypt, CFDP file transfer, decrypt chain may be created and set in motion by placing a file in the local encrypt dropbox. The encrypt, decrypt, CFDP dropbox chain is currently the only method TReK provides to automate file encryption/decryption using ION CFDP. Unlike ION CFDP, native CFDP provides access to the CFDP PDUs, making it possible to configure the TReK native CFDP application to encrypt and decrypt all CFDP transactions (e.g., "put", "get", "message", "create_file", "delete_file" ...) and avoid creating encrypt and decrypt dropboxes. Simply add a peer public key path and_file name to the end of the remote entity line in the native section of the CFDP configuration file. Review the description of the CFDP configuration file's remote entity IDs for further information on this native CFDP encrypt/decrypt configuration option.

Acceptable formats of the encrypt or decrypt dropbox primitive string are as follows:

- dropbox <encrypt/decrypt> <"dropbox path"> <" peer public key path and filename"> <"destination path"> <crypt block size> <"successful transaction path">
  (e.g., dropbox encrypt "D:/dropbox_dest1/" "D:/ peer_public.key" "D:/dest1/" 1000000 "D:/success/")

- dropbox <encrypt/decrypt> <"dropbox path"> <" peer public key path and filename"> <"destination path"> <crypt block size> <"">
  (e.g., dropbox decrypt "/home/user/dropbox_dest1" "/home/user/ peer_public.key" "/home/user/dest1" 1000000 "")

The encrypt/decrypt dropbox primitive includes the encrypt or decrypt service, the dropbox path, the peer public key path and filename, the destination path, the crypt block size and the successful transaction path. The encrypt or decrypt service identifies whether the dropbox is encrypting or decrypting files. The dropbox path defines the location of the encrypt or decrypt dropbox while the peer public key path and filename define the location and name of the peer public key file. The peer public key is the public key of the destination platform. The encrypt/decrypt dropbox primitive includes a destination path to the local directory where the new encrypted or decrypted file is created and stored. The crypt block size is an unsigned 32 bit value identifying the number of bytes that are read and encrypted or decrypted with every file read. A large crypt block size improves encryption and decryption performance but may also tax a CPU. If the successful transaction path is defined, as shown in the first example, the dropbox will move the original file placed in the dropbox to the successful transaction directory if and only if a new encrypted or decrypted file is successfully created and stored in the dropbox's destination directory. If the successful transaction path is empty, as shown in the second example, the dropbox will delete the original file placed in the dropbox if and only if a new encrypted/decrypted file is successfully created and stored in the dropbox's destination directory. If the encrypt or decrypt dropbox fails to encrypt or decrypt a file, the file will be renamed with a time tagged ".droperror" extension and remain in the dropbox. The encrypt or decrypt dropbox will not attempt to encrypt or decrypt a file with a ".droperror" extension in its filename. The TReK encryption architecture uses OpenSSL's FIPS 140-2 validated cryptographic module.

5.4 How to Create a Frag or Defrag Dropbox

This section describes how to create a fragmentation or defragmentation dropbox. The CFDP console application provides the ability to transfer very large, multi-Gigabyte, files by splitting the files apart using a fragmentation dropbox, transferring the file fragments, using a CFDP dropbox, to a defragmentation dropbox where the file fragments are put back together producing the original very large, multi-Gigabyte, file. A TReK CFDP configuration file frag or defrag dropbox primitive defines a frag or defrag dropbox's operation parameters including where the dropbox is located, the size of the file fragments and the frag or defrags destination directory and successful transaction directory. Frag and defrag dropboxes are created during initialization of the TReK CFDP library when InitToolkitCfdp function reads the TReK CFDP configuration file. Pre-existing dropbox files are immediately fragmented after the creation of the frag dropbox.

If the local destination directory of a frag dropbox is a CFDP dropbox, the file fragment will automatically be transferred to the CFDP dropbox's remote destination directory. If the CFDP dropbox's remote destination directory is a defrag dropbox, the fragmented file will automatically be put back together and moved to the defrag dropbox's destination directory when all the file fragments have been received by the defrag dropbox. By chaining together frag and defrag dropboxes with a CFDP dropbox, a completely automated file fragmentation, CFDP file transfer, file defragmentation chain may be created and set in motion by placing a file in the local frag dropbox. In addition, encrypt and decrypt dropboxes may be chained to the frag and defrag dropboxes producing an automated sequence of file encryption, file fragmentaion, CFDP file transfer, file defragmentation and file decryption. The fragmentation dropbox comes in two flavors: "frag" or "frag_cfdp". A "frag" dropbox creates a series of file fragments and immediately places the fragments in the dropbox's destination directory which may or may not be a CFDP dropbox directory. If the "frag" dropbox destination directory is a CFDP dropbox, the file fragments are downlinked simultaneously in a series of independent CFDP transactions. A "frag_cfdp" dropbox creates a series of fragments but only after the successful CFDP transfer of the previous file fragment to the remote destination of CFDP dropbox. In other words, a "frag_cfdp" dropbox's destination directory must be a CDFP dropbox directory and the "frag_cfdp" dropbox will only create the next file fragment after the previous file fragment has been successfully transferred by the CFDP dropbox. For this reason, a "frag_cfdp" dropbox's destination directory must be CFDP dropbox directory. If multiple files are added to a "frag_cfdp" dropbox simultaneously, the "frag_cfdp" dropbox will process the files one at a time, fragmenting and transferring all the fragments from one file prior to fragmenting and transferring all the fragments from the next file in the "frag_cfdp" dropbox. A "frag_cfdp" dropbox will require more time to transfer a very large file but has the advantage of an orderly and immediate cancellation of file fragmentation if there are problems transferring a file fragment using CFDP.

Acceptable formats of the frag/defrag dropbox primitive string are as follows:

- dropbox <frag/frag_cfdp> <"dropbox path"> <“destination path"> <file fragmentation size> <“successful transaction path">
(e.g., dropbox frag "D:/dropbox_dest1/" "D:/dest1/" 100000000 "D:/success/")

- dropbox <frag/frag_cfdp> <"dropbox path"> <“destination path"> <file fragmentation size> <“successful transaction path">
(e.g., dropbox frag_cfdp "D:/dropbox_dest1/" "D:/dest/" 100000000 "")

- dropbox <defrag> <"dropbox path"> <“destination path"> <“successful transaction path">
(e.g., dropbox defrag "D:/dest1" "D:/final_destnation" "")

The frag/frag_cfdp dropbox primitive includes the frag or frag_cfdp service, the dropbox path, the destination path, the 32 bit file fragmentation size, in bytes, and the successful transaction path. The defrag dropbox primitive includes the defrag service, the dropbox

path, the destination path and the successful transaction path. The frag, frag_cfdp or defrag service identifies whether the dropbox is fragmenting or defragmenting/reconstructing files. The frag or frag_cfdp dropbox breaks up a file into fragments sized to match the desired file fragmentation size, in bytes, and creates a new file fragment name by adding the current file fragment count and total fragment count to the fragmented file's dropbox extension. The defrag dropbox parses the file fragment name to identify the file's current and total fragment count prior to reconstructing the original file.The frag and defrag dropbox primitives also include a destination path to the local directory where the file fragments or reconstructed files are stored. If the successful transaction path is defined, as shown in the first example, the dropbox will move the original file or file fragments placed in the dropbox to the successful transaction directory if and only if new file fragments or reconstructed files are successfully created and stored in the dropbox's destination directory. If the successful transaction path is empty, as shown in the second example, the dropbox will delete the original file or file fragments placed in the dropbox if and only if new file fragments or reconstructed files are successfully created and stored in the dropbox's destination directory. If the frag or defrag dropbox fails to fragment or reconstruct the file, the file will be renamed with a time tagged ".droperror" extension and remain in the dropbox. The frag or defrag dropbox will not attempt to fragment or defragment/reconstruct a file with a ".droperror" extension in its filename.

5.5 How to Turn on Message Logging

This section describes how to log messages to a file. Message Logging will only capture messages generated after Message Logging is turned on. Any messages generated before message logging was turned on will not appear in the log.

1. Launch the CFDP console application and enter the log command primitive as follows: log <”pathname”> < log debug messages (true or false)>
2. The log command primitive includes:
    a. A pathname with the path to the directory of the log file as well as the name for the log file.
    b. A true or false Boolean identifying whether debug messages are written to the log file.

Alternatively, the CFDP console application may be configured to log messages using the log parameters found in the TReK CFDP configuration file. These parameters include:

1. log_messages_in_file
2. log_debug_messages
3. log_file_path
4. log_file_name

Defintions for each of these TReK CFDP configuration file parameters may be found in section 5.1 Table 4.

5.6 How to Turn on Statistics Logging

This section describes how to record a snapshot of device and packet statistics to a user specified file. The snapshot of device and packet statistics is updated once a second with current statistics information at both the device and packet level. Device statistics provides information on all packets that are being received or sent by the device. Packet statistics provides information on the individual packet groups that are being received or sent by the device. The TReK CFDP library does not divide packets into groups so packet statistics mirror device statistics.

1. Launch the CFDP console application and enter the statistics command primitive as follows: stat <"pathname">
2. The statistics command primitive includes:
    a. A pathname with the path to the directory of the statistics file as well as the name for the statistics file.

Alternatively, the CFDP console application may be configured to record a snapshot of statistics using the record statistics parameters found in the TReK CFDP configuration file. These parameters include:

1. record_stat_snapshot_in_file
2. record_packet_statistics
3. record_stat_file_path
4. record_stat_file_name

Definitions for each of these TReK CFDP configuration file parameters may be found in section 5.1 Table 4.

Table 5 identifies and describes the device statistics parameters. Note, ION CFDP relies upon the bundle protocol to assure delivery of file segments therefore, device and packet statistics are not relevant.

Device Statistics Parameter	Description
Device Key	A character string that uniquely identifies each device.
IP Address	The IP address of the device if it is a socket.
Port (C/L/S)	The port number of the device if it is a socket. If the socket is a client socket then the port number will be followed by two '/'. If the client socket is connected to a listener socket, the listener's port number is also listed. If the socket is a server socket then the client port number that is connected to the server is listed first, followed by two '/' and

	the server's listener port number. If the socket is a listener socket the listener's port number is listed between two '/'.
Protocol	The IP transportation protocol, either TCP or UDP, if the device is a socket.
Segments Rcvd	The number of segments received by the device if the device is a TCP socket.
Pkts Rcvd	The total number of packets received by the device.
Pkts Sent	The total number of packets sent by the device.
Pkt Rcv Rate	The number of packets received by the device in the last second.
Max Pkt Rcv Rate	The maximum packet receive rate experienced by the device.
Kbit Rcv Rate	The number of kilobits received by the device in the last second.
Max Kbit Rcv Rate	The maximum kilobit receive rate experienced by the device.
Pkt Send Rate	The number of packets sent by the device in the last second.
Max Pkt Send Rate	The maximum packet send rate experienced by the device.
Kbit Send Rate	The number of kilobits sent by the device in the last second.
Max Kbit Send Rate	The maximum kilobit send rate experienced by the device.
Pkts Dropped	The total number of packets that were dropped because they could not be temporarily stored in a queue or buffer. The most likely cause of dropped packets is packets arriving at very high packet rates and/or a queue size that is too small.

Table 5 Device Statistics

Table 6 identifies and describes the packet statistics parameters for a device.

Packet Statistics Parameter	Description
Packet Key	A character string that uniquely identifies each packet type.

Pkts Rcvd	The total number of packets that are received and identified as this packet type.
Pkts Sent	The total number of packets that are sent and identified as this packet type.
Pkt Rcv Rate	The number of packets received and identified as this packet type in the last second.
Max Pkt Rcv Rate	The maximum packet receive rate experienced by this packet type.
Kbit Rcv Rate	The number of kilobits received and identified as this packet type in the last second.
Max Kbit Rcv Rate	The maximum kilobit receive rate experienced by this packet type.
Pkt Send Rate	The number of packets of this packet type sent in the last second.
Max Pkt Send Rate	The maximum packet send rate experienced by this packet type.
Kbit Send Rate	The number of kilobits of this packet type sent in the last second.
Max Kbit Send Rate	The maximum kilobit send rate experienced by this packet type.
Pkts Dropped	The total number of packets, of this type, that were dropped because they could not be processed by another device. The most likely cause of dropped packets is packets arriving at very high packet rates.
Pkt Seq Errors	The total number of packet sequence errors identified for this packet type. For example, the primary header of the CCSDS packet contains a 14-bit number that is used as a sequence count. For each packet that arrives, the sequence count is compared to the sequence count of the previous packet. If the count is not the next in the sequence, the packet sequence error value is incremented.
Max Pkt Seq Error	The maximum packet sequence error experienced by this packet type.

Table 6 Packet Statistics

5.7 How to Turn on Metrics Logging

This section describes how to record a snapshot of CFDP metrics to a user specified file. The snapshot of CFDP metrics is updated once a second with the completion status of each CFDP transaction. The CFDP metrics are divided into sending and receiving categories and grouped by file size ranging from less than a one megabyte to over a gigabyte. The metrics include calculations on the number and percent of files sent or received, the minimum, maximum, and average file transfer time and the number and percent of files that required packet retransmission.

1. Launch the CFDP console application and enter the metrics command primitive as follows: metric <"pathname">
2. The metrics command primitive includes:
    a. A pathname with the path to the directory of the metrics file as well as the name for the metrics file.

Alternatively, the CFDP console application may be configured to record a snapshot of metrics using the record metrics parameters found in the TReK CFDP configuration file. These parameters include:

1. record_cfdp_metrics_snapshot_in_file
2. record_cfdp_metrics_file_path
3. record_cfdp_metrics_file_name

Definitions for each of these TReK CFDP configuration file parameters may be found in section 5.1 Table 4.

Table 7 identifies and describes the CFDP metrics parameters. Note, ION CFDP relies upon the bundle protocol to assure delivery of file segments therefore, NAK metrics are not relevant.

CFDP Metrics Parameter	Description
File Size (MB)	The minimum and maximum file size, in megabytes, for the group.
Success Count	The number of successful file transfers for the group.
Success %	The percentage of successful file transfers for the group.
Cancel Count	The number of canceled file transfers for the group.
Abandon Count	The number of abandoned file transfers for the group.
Fail Count	The number of failed file transfers for the group.

Unknown Count	The number of unknown file transfers for the group.
Min Trans Time (sec)	The minimum successful file transfer time in seconds.
Max Trans Time (sec)	The maximum successful file transfer time in seconds.
Avg Trans Time (sec)	The average successful file transfer time in seconds.
Success W/ NAK Cnt	The number of successful file transfers that required one or more NAK packets.
Success W/ NAK %	The percentage of successful file transfers that required one or more NAK packets.
Min NAK CNT/Trans	The minimum number of NAK packets that had to be transferred for a successful file transfer requiring NAK packets.
Max NAK Cnt/Trans	The maximum number of NAK packets that had to be transferred for a successful file transfer requiring NAK packets.
Avg NAK Cnt/Trans	The average number of NAK packets that had to be transferred for a successful file transfer requiring NAK packets.

Table 7 CFDP Metrics

6 Details

This section covers various application details.

6.1 Configuration

The CFDP console application may be configured to hide or not display all console application messages by setting the "display" parameters to false. This configuration might be preferable if running as a flight application.

The GUI and default parameters referenced in the configuration file are applicable to the TReK CFDP GUI application and are ignored by the TReK CFDP console application.

The configuration file "support_cfdp_status_requests" parameter should be set to "false" to disable actively monitoring transactions with the TReK CFDP library's transaction monitoring functions. The TReK CFDP console application does not actively monitor transactions but example code that actively monitors transactions may be found in the TReK CFDP library examples.

The console application may initialize its list of primitives by adding them to the configuration file. Alternatively, the primitive lists may be initialized using the "process" command primitive in the console application.

6.2 Transaction

The CFDP console application uses the TReK CFDP library and TReK Device Service library to provide CFDP functionality. The CFDP console application's ION CFDP mode may be configured to send a final transaction result message (e.g., success or fail) to the source by setting the "support_transaction_result_message" flag in the console application's configuration file (see section 5.1 Table 4). Detailed information about CFDP transactions may be found in the TReK CFDP library's on-line help documentation.

6.3 Messages and Message Logging

The CFDP console application generates a variety of messages throughout the file transfer activity. The messages are categorized by their message severity. Message categories include error messages, warning messages, information messages, progress messages and debug messages. Progress messages provide transaction status information including the size of the file, bytes transferred, percentage complete and transaction state (e.g., sending, receiving, suspend, resume…). Information messages include information on the start of a transaction as well the success or failure of the transaction. The console application may display error, warning, information, progress and debug messages. The display of these messages is controlled by the "display" parameters in the configuration file. The console application may also be configured to log messages. Message logging is controlled by the "log" command primitive or parameters in the configuration file. Logging may be configured to include or exclude debug messages. When logging is turned off using the "stop log" command primitive, the log file name is appended with a time tag to produce a unique log file name.

7 FAQ and Troubleshooting

This section addresses Frequently Asked Questions and provides tips for troubleshooting common gotchas.

7.1 Is There an Easy Way to Transfer the Contents of a Directory?

Yes. To transfer the contents of a directory, enter the absolute path to the directory. The CFDP application will transfer all the files in the first level of the directory. Subdirectories will not be transferred. Be sure to include a forward slash '/' at the end of the directory path when entering the absolute path into the Source and Destination fields.

7.2 What is class1 and class2?

Class1 and class2 are transmission properties used with Native CFDP. Note: Special thanks to NASA/GSFC for the following user friendly definitions:

"CFDP provides three **Service Classes**. Service **Class 1** simply sends each file; there are no replies from the receiver, nor is there any guarantee of reliable delivery. Service **Class 2** ensures reliable file delivery; any required retransmissions are requested and performed by CFDP. Service **Class 3** provides **Proxy Operations** (e.g. Entity 'A' tells Entity 'B' to make a request of Entity 'C')."

The TReK CFDP application supports Class 1 and Class 2 when configured for Native CFDP. When typing in one of these choices please use **class1** and **class2**.

7.3 What is "////"?

"////" is a value used to designate the default set of transmission properties defined in the Configure dialog for ION CFDP. "/" can also be used. When configured for ION CFDP, transmission properties are specified using the Configure dialog and will apply to all CFDP transactions. For more information about transmission properties please reference section 5.1 Table 4.

7.4 Source and Destination Constraints

Source and Destinations must be identified using an absolute path. The absolute path name consists of the full path and the file name. The absolute path must meet the following criteria:

- The ION CFDP library works with a source absolute path limited to 255 bytes and a destination absolute path limited to 246 bytes.

- The Native CFDP library works with primitive strings. The "put" primitive string format is as follows:

  put –class2 "source absolute path" eid "destination absolute path"

  All Native CFDP library primitives are limited to 255 bytes.

The file size must meet the following criteria:

- The size of the file to be transferred must be greater than 0 Bytes.
- The size of the file to be transferred cannot exceed 4.2 Giga Bytes for Native CFDP.
- The size of the file to be transferred cannot exceed 2.1 Giga Bytes for ION CFDP when transferring from a device hosting the Windows Operating System (OS) to a device hosting the Windows OS.

- The size of the file to be transferred cannot exceed 2.1 Giga Bytes for ION CFDP when transferring from a device hosting the Windows OS to a device hosting the Linux OS.

Consider using the fragmentation and defragmentation dropboxes when transferring files that exceed CFDP's file size limitations.

7.5 My File Starts to Transfer and Then Stops

Chances are the remote entity is unavailable or is not configured as you expected. Check both the Local and Remote entity configurations and ensure the EIDs are correct, the IP address and port information is correct, and both entities are up and running.

7.6 Transfer Results When Item Exists at Destination

Transfer results when an item exists at a destination differ based on configuration.

When configured for Native CFDP

If you attempt to "put" an item to a destination and the item already exists at the destination, you will see a "cancelled (Filestore rejection)" error message and the existing item will not be overwritten.

When configured for ION CFDP

If you attempt to "put" an item to a destination and the item already exists at the destination, you will see a "failed" error message and the existing item will not be overwritten.

7.7 Important Things to Know When Using the Get Primitive

The TReK CFDP software provides the capability to "get" or retrieve one or more files from a remote destination. It is important to note that the CFDP Blue Book describes implementation of a "get" as a proxy "put". TReK Native CFDP implements the CFDP Blue Book defined proxy "put" function using proxy request and response messages. TReK ION CFDP does not implement proxy "put" function using proxy request and response messages. In addition, the "get" directive is not supported in all ISS CFDP Native and ION implementations. Therefore, the "get" request will only succeed if both sides of the file delivery transaction are using TReK software. The TReK ION CFDP "get" function initiates the file transfer process by delivering an equivalent "put" primitive character string to the remote platform's CFDP software using a "get" request message. In addition, a "get" response message is generated providing transaction status information to the initiator. There is an error scenario in which the initiator of the ION CFDP "get" receives no feedback. If an ION CFDP "get" request message or "get" response message never reaches its target platform, the initiator will receive no status describing the result of the "get" request.

.

7.8 How Does Suspend Transactions Work?

When configured for Native CFDP

CFDP suspending transactions by suspending both data transmission and timeout clocks associated with the local platform's file transfer transactions. The remote platform is not notified of the suspension of file transfer transactions on the local platform and may exceed its timeout limits if the local platform does not resume its file transfer transactions for an extended period of time. If the local file transfer suspension is for an extended period of time, the remote platforms should receive a separate suspend transaction command to avoid exceeding its timeout limits. Both platforms may resume file transfer transactions when they receive separate resume file transfer transaction commands.

When configured for ION CFDP

The ION CFDP application completes its file transfer responsibilities when it hands off to ION's BP application. The handoff may be relatively quick depending on the size of the file. The suspend transaction request will not suspend a file's transfer after the ION CFDP application hands off a transaction to ION's BP application. The suspend transaction request does not affect ION CFDP file reception. The lifespan of the packet bundles must also be considered when suspending for an extended period of time.

7.9 CFDP Transactions in an AOS/LOS Environment

Maintaining CFDP transactions across extended LOS periods is problematic. ION CFDP solves this issue by relying on DTN's store and forward infrastructure. A contact plan that predicts AOS/LOS periods may be used to support ION CFDP transactions between a flight node and a ground node if the two nodes are communicating directly with each other. If the ION CFDP transaction communication path includes the Huntsville Operations Support Center (HOSC) DTN2 node, the DTN2 node will store and forward the ION CFDP transaction during AOS/LOS periods. Native CFDP is not supported by an underlying store and forward DTN infrastructure. A user must configure timeouts or manually suspend and resume CFDP at both ends of a transaction to maintain the transaction across a LOS period. The TReK CFDP library solves this native CFDP issue by providing an automatic suspend and resume capability during LOS and AOS periods. When the TReK CFDP library senses a LOS event, it automatically suspends all CFDP transactions. When the TReK CFDP library senses an AOS event, it automatically resumes all CFDP transactions. TReK identifies AOS and LOS events by sending and receiving a four byte connectivity packet over a predefined UDP socket. TReK's automatic suspend and resume design maintains native CFDP transactions across multiple LOS periods without adjusting timeout values or requiring user intervention. If new CFDP transactions are requested during a LOS period, the pending transaction requests are placed in a queue and are submitted to the TReK CFDP library at the beginning of the next AOS period. The automatic suspend and resume parameters in the TReK CFDP configuration file configure and control TReK's automatic suspend and resume feature. For more information about automatic suspend and resume parameters please reference section 5.1 Table 4.

7.10 How Do I Include My Crypt User Passphrase in the CFDP Console App?

If you created a public/private key pair using the "trek_crypt" application and included a user passphrase, you must include the user passphrase in the command line that launches the CFDP console application or you must modify the CFDP console application source code and rebuild the application.

To launch the CFDP console application with a user passphrase, include the passphrase, in double quotes, after the configuration file path and filename as follows:

- trek_cfdp_console.exe "D:/toolkit_cfdp_config.txt" "passphrase"

or

- ./start_trek_cfdp_console.sh "/home/username/toolkit_cfdp_config.txt" "passphrase"

To modify the CFDP console application source code to use a passphrase to unwrap/decrypt the private key, change "InitToolkitCfdpAndCryptPassphrase" in main() to include your passphrase as the second argument in the function call. The TReK CFDP console source code is located in the install directory on Windows under /example/trek_toolkit_cfdp_api/trek_cfdp_console/. On Linux the source code is located in the install directory under /example/ trek-deviceservices /trek_toolkit_cfdp_api/trek_cfdp_console/. You will have to recompile the CFDP console application after making the change. You may use the Windows trek_cfdp_console.vcxproj file or Linux makefile that is included with the source code when recompiling the TReK CFDP console application. Finally, you will need to rename the original trek_cfdp_console.exe (always keep a copy of the original executable) located in TReK's bin directory prior to moving the new trek_cfdp_console.exe to TReK's bin directory.