TREK

MIGRATION

GUIDE

June 2019

TABLE OF CONTENTS

PARAGRAPH PAGE

FIGURES

FIGURE PAGE

1 Welcome

This document provides information about moving from previous versions of TReK software to the TReK 5.x series of software. This is intended to be general information. If you need specific assistance please contact the TReK team using one of the contact methods listed in the Technical Support section.

2 Technical Support

TReK Help Desk E-Mail, Phone & Fax:

E-Mail:	trek.help@nasa.gov
Telephone:	256-544-3521 (7:00 a.m. - 3:30 p.m. Central Time)
Fax:	256-544-9353

TReK Help Desk hours are 7:00 a.m. – 3:30 p.m. Central Time Monday through Friday. If you call the TReK Help Desk and you get a recording please leave a message and someone will return your call. E-mail is the preferred contact method for help. The e-mail message is automatically forwarded to the TReK developers and helps cut the response time.

3 Introduction

The purpose of this document is to assist in migrating from the TReK 3.x or TReK 4.x series of software to the TReK 5.x series of software. Section 4 addresses some of the key differences between the TReK 3.x software and the TReK 5.x software. Section 5 addresses some of the key differences between the TReK 4.x software and the TReK 5.x software. Section 6 identifies some of the new concepts in the TReK 5.x software. Section 7 discusses each TReK 3.x software application and application programming interface and describes where to find the associated capabilities in the TReK 5.x software. Section 8 provides a general set of Migration steps for moving from TReK 3.x to TReK 5.x. Section 9 discusses capabilities that have not been included in the TReK 5.x software and the future plans for each of those capabilities. Section 10 describes new capabilities that were not in the TReK 3.x software, but are included in the TReK 5.x software.

4 Key Differences Between TReK 3.x and TReK 5.x

This section identifies some of the key differences between the TReK 3.x software and the TReK 5.x software.

4.1 Operating Systems

The TReK 3.x software was only available for Windows. The TReK 5.x software is available for Windows and Linux. Please see the TReK Web Site for specific operating system details.

4.2 Databases

The TReK 3.x databases were Microsoft Access Databases. The TReK 5.x databases are SQLite databases. There are several ways to input data into TReK 5.x databases. These are discussed in section 7.7 and section 7.8.

4.3 Metadata

The TReK 5.x software provides support for metadata defined in databases and metadata files. For more information please see section 10.3.

4.4 HOSC Login

The HOSC Login capability is no longer embedded in telemetry and command applications. It is a stand-alone application.

4.5 Running Multiple Instances of TReK Applications

In the TReK 3.x series of software, only one copy of the Telemetry and Command applications could be run at a time. In the TReK 5.x series of software multiple instances of the applications can be run simultaneously.

4.6 Application Programming Interface

Most of the APIs included in the TReK 3.x series of software have been included in TReK 5.x with a goal of 90+% of users requiring no recoding of user developed programs. Some programs may run without recompiling, but it is recommended that you recompile and relink your application.

Release 5.x introduces C++ classes for telemetry and command APIs. The details about these classes can be found in the TReK Help application and the TReK website (https://trek.msfc.nasa.gov/trek_documents.htm).

For more detailed information on the release 3.x telemetry and command API see section 7.5.

4.7 Data Stores

In the TReK 3.x series of software, only one copy of the Telemetry and Command applications could be run at a time. These applications contained a single data store that was used to store telemetry and command information in support of the associated

Application Programming Interface. Since the TReK 5.x series of software supports running multiple instances of an application, each instance of an application has its own data store. The first instance of an application always uses a default data store name. Therefore, if only one instance is run there will only be one data store. When making API calls, the data store name is part of the API call. If the data store name is left empty, the default data store name will be assumed. The TReK 3.x API calls will work as is and data store information is handled internally by the API.

4.8 XML Configuration Files

The software applications in TReK 3.x saved configuration information in binary. The software applications in TReK 5.x save configuration information in XML. Some of the TReK 5.x APIs use configuration files that are in an ASCII format and others use XML.

4.9 Data

TReK 3.x used the terms telemetry and commanding to refer to data sent from the spacecraft and from the ground respectively. The term data is meant to be somewhat abstract. It doesn't matter if something is telemetry or commanding at the bit and byte level. It's just data. In TReK 5.x, three simple concepts cover most of what data is in TReK: packets, parameters, and parameter collections. Please reference section 6.5 for details about how TReK 5.x looks at packets and parameters. If you have used the TReK 3.x Pass-Thru packet capability, please reference section 7.1 and 7.7 for information about how this works in TReK 5.x.

5 Key Differences Between TReK 4.x and TReK 5.x

This section identifies some of the key differences between the TReK 4.x software and the TReK 5.x software.

TReK HOSC Login

In the TReK 4.x software the HOSC Login was embedded in the TReK HPEG application. In TReK 5.x the TReK Command application was added. This application also uses a HOSC Login. The HOSC Login capability is now in the TReK HOSC Login application. This application provides a way to create a HOSC Login session that can be shared across TReK applications. The HPEG application has been modified to provide the capability to enter the name of the HOSC Login session created with the TReK HOSC Login application. The TReK Command application also provides the capability to enter the name of the HOSC Login session.

TReK IONizer

In the TReK 4.x software the IONizer application would automatically detect when connecting to the HOSC DTN ground gateway and display a dialog box to ask if a proxy should be used. TReK 5.x adds a third option to run with a series of scripts to

reconfigure ION as needed. See Section 5.1 in the TReK IONizer User Guide for more details.

CCSDS Telemetry and Commanding
CCSDS Telemetry and Commanding capabilities are available in TReK 5.x.

New Capabilities
There are many new capabilities available in TReK 5.x. This includes enhancements to existing software like CFDP, and new capabilities like data recording and data playback. Please reference the TReK 5.x Getting Started guide for more information.

6 New Concepts

This section addresses some of the new concepts in the TReK 5.x software.

6.1 TReK Workspace

The TReK Workspace is a directory structure created by the TReK 5.x software in your home directory. It is similar to the directory structure that was created by the TReK 3.x software in C:\Users\<username>AppData\Roaming\TReK. The TReK Workspace will be created by TReK 5.x applications that use this area such as the TReK Data, TReK Command, and TReK Metadata applications.

6.2 Packet Keys

TReK 5.x introduces you to the premier of the packet key, a new methodology for identifying, filtering and routing packets in TReK Data and TReK Command. The packet key replaces the classic TReK 3.x Packet ID (APID), Packet Type and Data Mode identification scheme with a modernized key that travels with the packet as it traverses through the various TReK devices that compose TReK Data and TReK Command. A TReK device is a generic term used to describe various software constructs including sockets, dynamic link libraries, shared objects, pipes, shared memory, etc. The packet key is "glued" to the packet as soon as it is received, read or created by a TReK device. The packet key gives you the power to tell packets what devices they need to visit (or not visit) much like parents and their directions to their teenagers about friends only in the packet key case, the packets will actually follow directions.

So, how does a packet key get created? From a software object called a Packet Header Processor (PHP). A PHP defines the location of a variety of packet header fields. A PHP has the ability to process or retrieve the values in these individual packet header fields using the field's PHP definition. These fields may be associated with the size of the packet, the sequence count of the packet or the embedded time of the packet. The fields may also be associated with the generation of a packet's packet key.

So how does a PHP get defined or created? There are two answers to that question. The first answer is by the user. A PHP can be defined and initialized by reading a PHP ASCII configuration file of name value pairs. A template for this file may be found in the config directory of your TReK Workspace. The template includes a brief description of the various PHP parameters. A more detailed discussion about the PHP configuration file may be found in the TReK Record API. The second and most popular way a PHP is defined is by metadata from the database or metadata from a metadata file. By attaching a PHP to a TReK device, you enable TReK Data or TReK Command to generate and glue packet keys to the packets that a TReK device receives and forwards. TReK devices downstream process or ignore packets based upon the packet's key.

So what does a packet key look like? The packet key is simply alphanumeric dotted string notation of one or more packet key field values (packet header parameters) and zero or more character strings formatted as follows:

<packet type>.<packet header field value(s)>.<source ID>.<trailer>

All fields are optional though a packet key with no field definitions does not make much sense. Packet type, source ID and trailer are simply alphanumeric strings that may be added to every packet associated with the TReK device. For instance, common packet type string notations in the TReK metadata files are "PdssPayload" and "IssCcsds". Packet header field values are where all the work is done in generating the packet key. These field values are the specific parameters in the packet header. The PHP processes the specific parameters from the packet's header using the parameter's description in the PHP configuration file. The parameter's description information includes its location, byte order, value offset and precision. Packet header field values in a packet key may also be replaced by an enumerator name by specifying the field value and its enumerated name in the PHP configuration file.

Some simple examples of packet keys with enumerated values include: PdssPayload.RT.PL.7 and IssCcsds.7. The first example is a PDSS payload packet with a real-time data mode and an APID value of 7. The second example is an ISS CCSDS packet with and an APID value of 7. One and only one PHP may be associated with a TReK device and all data processed by a TReK device should match the TReK device's PHP stream type. In other words, you should not flow ISS CCSDS data to a TReK device configured to receive PDSS payload data because the PHP will generate incorrect packet keys.

6.3 Colors and Data Flow

If you used the TReK Telemetry Processing application you are probably used to looking for the packet to turn green to indicate you are receiving data. TReK 5.x still uses color to provide information about data flow, but there is additional information available and it's important to understand what it means when something turns green. Please reference the TReK Data User Guide for details.

6.4 Data Store

Applications that have an associated API will contain a data store. In most cases you will never need to know that the data store exists, but it can be important to understand a little bit about it for some users. Currently there are two applications that use a data store: TReK Data (replaces Telemetry Processing) and TReK Command (replaces Command Processing).

For TReK Data the data store is created when the application initializes. If you only run a single instance of TReK Data, it will use the creative name "DefaultDataStore". If another instance of TReK Data is started, you will be prompted to enter a name for the data store. It is anticipated that most users migrating from TReK Release 3.x will use a single instance of TReK Data.

The Release 3.x compatible telemetry APIs that are included in Release 5.x will always use the "DefaultDataStore". If you plan on using applications that incorporate the Release 3.x APIs, you should just use a single instance of TReK Data. If you want to run multiple instances of TReK Data, you will need to incorporate the C++ API into your applications.

For TReK Command the data store is created when you activate a destination. Each TReK Command application supports a single destination. If you must connect to multiple command destinations, you will run multiple instances of the TReK Command application.

The Release 3.x compatible command APIs that are included in Release 5.x will work with the multiple instances of the TReK Command application. Each of those API functions included a destination name which is the same in the Release 5.x software.

6.5 Data

This section describes some of the new concepts when working with data in TReK 5.x.

Previous versions of TReK used the terms telemetry and commanding to refer to data sent from the spacecraft and from the ground respectively. The term data is meant to be somewhat abstract. It doesn't matter if something is telemetry or commanding at the bit and byte level. It's just data. Three simple concepts cover most of what data is in TReK: packets, parameters, and parameter collections.

Packets are most often the data that are sent from one system to another system. Commands and telemetry are just packets. Command data is packets that tell another system to do something. Telemetry data is packets that supply information about the system sending the packet.

Parameters are the individual data values that contain information about the state of the sending system or actions to be taken by the receiving system. Parameters have a value

and are either placed in the outgoing data or pulled from the incoming data. TReK uses the terms “build” to describe placing parameters in a packet and “extract” to describe pulling parameters from a packet.

Parameters are grouped with related parameters into collections named parameter collections. Parameter collections are the basic building blocks of packets which are the data sent from one system to another. Figure 1 has four views of the same packet. The first row shows a packet as a single entity that could be sent between systems. The second row shows that the packet is composed of parameter collections and another packet. The third row shows that eventually a packet will break down into a series of parameter collections. The final row shows that all parameter collections are a series of parameters. Each row is a different view of the same data.

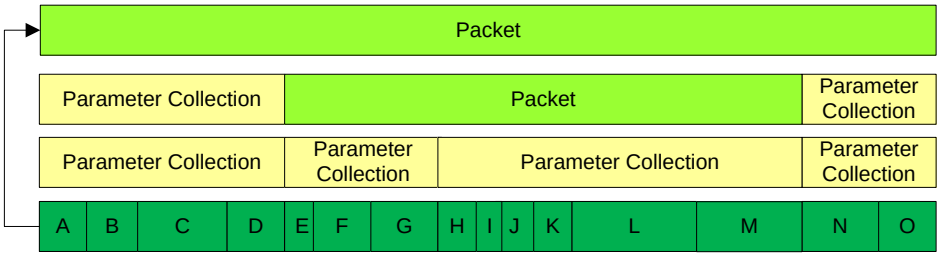

Figure 1 Packets, Parameter Collections, and Parameters

The packet is the largest aggregation of data in TReK. Packets are divided into three zones: header, data, and trailer. One or more zones must be defined in a packet for it to be considered valid. Figure 2 shows each of the three zones and their relative locations. Each zone of a packet contains either another packet or a parameter collection.

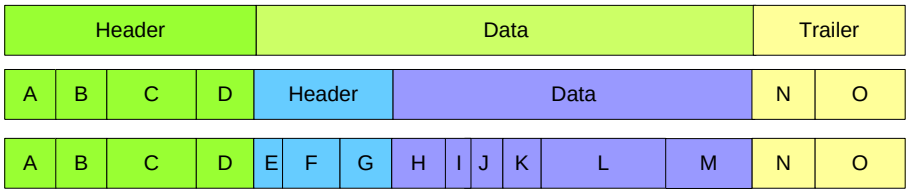

Figure 2 A Packet and Its Zones

The first line in the figure above shows a packet that has all three zones defined. The second level shows that the header and trailer zone are composed of parameter collections which contain one or more parameters. The data zone is composed of another packet which only has the header and data zones defined. The third line shows that the packet in the data zone of the top level packet is composed of two parameter collections and that all of the data in a packet will eventually break down into a series of parameters.

For more information about working with data in TReK 5.x, please reference the TReK Concepts document.

6.6 Database

As described in the previous section, packets can be embedded within another packet. So when you add or convert a packet into a TReK 5.x telemetry database, sometimes more than one packet will get added to the database. For instance when you import an EHS GSE definition file into the database, two packets will get added to the database. There will be a GSE CCSDS type packet and then that packet will get embedded within a PDSS GSE packet. The first packet only has the CCSDS headers on it while the latter packet has the EHS headers which get wrapped around the first. When you go to process that packet in the Data application you would add the PDSS GSE packet, because you would be receiving the data with the EHS headers and CCSDS headers. The same is true for PDSS Payload type packets. The way TReK looks at it, it is a CCSDS type packet embedded in a PDSS Payload type packet as shown in Figure 3.

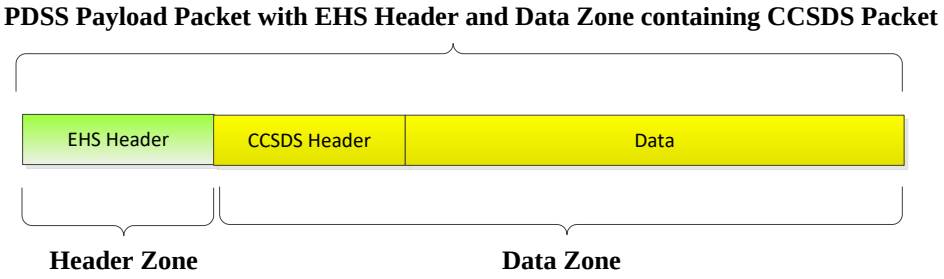

Figure 3 PDSS Payload Packet Example

7 Migration Information By Application and API

This section addresses migrating from each TReK 3.x application.

7.1 Telemetry Processing

The Telemetry Processing application in TReK 3.x provides access to various telemetry capabilities. The corresponding applications in TReK 5.x are the TReK Data application and the TReK Playback application. The TReK Data application provides the capability to receive, process, record, forward, and display data. The TReK Playback application provides the capability to play back data from recorded data files. The capability to view recorded data is available by using both the TReK Data and TReK Playback applications together. The TReK Playback application can be used to read the recorded data files and forward the data to the TReK Data application where the packets can be viewed in a text/hexadecimal format. Some capabilities in the Telemetry Processing application have not been included in the TReK 5.x software. Please reference section 9 for details.

Additional Notes

- In the TReK 3.x Telemetry Processing application, the Add Packet dialog default for Processing was “Process Entire Packet”. In the TReK 5.x Data application, the Add Service dialog default for Processing is Off.
- If you were using the Pass-Thru Packet capability in Telemetry Processing, please reference section 7.7 for information about how to add a pass-thru packet to a TReK 5.x database. The TReK Data application does not have a specific pass-thru packet option since the metadata selected, along with the on-demand processing requested, drives the level of processing that is performed.

Recommended Reading

- TReK Data Tutorial
- TReK Playback Tutorial
- TReK Data User Guide
- TReK Metadata User Guide
- TReK Playback User Guide
- TReK Telemetry API

7.2 Command Processing

The Command Processing application in TReK 3.x provides access to various command capabilities. The corresponding applications in TReK 5.x are the TReK HOSC Login application and the TReK Command application. The TReK HOSC Login application provides the capability to log into the HOSC. The TReK Command application provides the capability to establish a command destination, update commands, send commands, track commands, and record command history information. The TReK Command application works with a single command destination. You can run multiple instances of the application if you need to communicate with multiple command destinations.

Recommended Reading

- TReK Command Tutorial
- TReK Command User Guide
- TReK HOSC Login User Guide
- TReK Command API

7.3 Remote Services

There is no corresponding application for the Remote Services application. Please reference section 9 for details.

7.4 Trainer Tools

The trek_eris_sim_console application provides a training interface for logging into the HOSC. The Telemetry Trainer and Command Trainer applications do not yet have corresponding applications. Please reference section 9 for details.

7.5 Application Programming Interfaces

The following sections describe what functionality is being included for Release 3.x APIs. There may not be 100% function support for each API. Please review the information below and the API documentation included in the Release 5.x. If you need a capability that is currently shown as not supported for Release 5.x or have any questions about transitioning user programs, please contact the TReK Help Desk.

Note: In TReK 3.x the API DLLs were placed in the C:/Windows/system32 directory which allowed programs to automatically pick up the DLLs. TReK 5.x allows multiple versions to be installed at the same time which prevents adding the TReK API DLLs to a globally accessible directory. You should copy the DLLs you need from the installation directory to your application directory. The required DLLs are listed in the main help page for each API.

7.5.1 TReK User API

The trek_user_api includes all of the functions included in the Release 3.x software with the exception of those related to pseudo telemetry. Details for all of the included functions and examples are available in TReK Help. There are a few slight differences that you need to be aware of:

The API no longer uses Windows events to signal for packet arrival. To ease transition to Release 5.x two new APIs have been added: WaitForPacketArrival() and UnregisterPacketArrivalEvent(). These can be used in addition to the GetPacketArrivalEventName() function if you want to be signaled when a new packet arrives.

The InitializeMultipleParameters() function immediately returns success and takes no action. This function was used to allow on demand processing of data in Release 3.x. For Release 5.x all processing of data is on demand and this function has no practical use. It is included to allow user programs to be used without modification.

The GetOverallStatus() function is under consideration for removal. In Release 3.x this function allowed users to access the status of a GSE parameter. Release 5.x makes this parameter available through a regular API call.

The following packet types are available in Release 5.x: PDSS Payload, GSE, GSE Merge, CCSDS, SCSC, and PRCU.

7.5.2 TReK User API.NET

The trek_telemetry_dotnet_api wrapped both the TReK User API and TReK Telemetry Processing API for use in .NET applications. The mapping of functions will be the same as the TReK User API.

7.5.3 TReK Command API

The trek_cmd_user_api includes all of the functions included in the Release 3.x software with the exceptions of those requiring remote services. The functions that are not included are: UplinkTReKCommand(), UpdateDestinationDatabase(), ChangeUplinkPacket(), GetValidationEventName(), GetUplinkPacketForValidation(), and SendValidationResults().

Note: The Command GUI can be used to set a flag to allow TReK to behave the same as TReK 3.x with respect to the AddHeaderAndUplinkCommand() function. In TReK 3.x this function required two extra bytes for the checksum. TReK 5.x no longer requires those bytes. The new flag (available on the Preferences dialog in the Command GUI) when turned on will remove those two extra bytes to mimic the TReK 3.x behavior. This will also change the TReK 5.x behavior for the InsertDataAndUplinkCommand() method of the CommandApi class. It should only be used if you are exclusively using the TReK 3.x compatible API.

7.5.4 TReK Command API.NET

The trek_command_dotnet_api wraps the TReK Command API for use in .NET applications. The mapping of functions will be the same as the TReK Command User API.

7.5.5 TReK Telemetry Processing API

The TReK Telemetry Processing API is not included in Release 5.x.

7.6 Command Bridge

The Command Bridge capability provided in the TReK 3.x software is available in the TReK Command application in the TReK 5.x software.

Recommended Reading

- TReK Command User Guide

7.7 Telemetry Database

The TReK 5.x Telemetry Database is a SQLite database. The TReK 5.x Metadata application provides access to interact with a TReK 5.x database. To directly access Telemetry Database tables you can use third party database tools available as applications or SQLite plugins that work with browsers. There is no direct translation available from Microsoft Access. You can create a new database using the TReK Metadata application and populate it using a TReK 5.x ASCII file or by using files from a HOSC partial database download. The TReK 5.x ASCII file format is described in the TReK Telemetry Database Definition document. An ASCII file can be used to add a fully

defined packet or a pass-thru packet. The TReK Metadata application also supports adding an EHS PDSS Payload Pass-Thru Packet definition and importing a HOSC GSE Packet definition into a TReK 5.x database. When a GSE packet definition is imported the parameter names are changed. In order to keep parameter names unique, characters representing the type of processing are added to the name. For example, suppose the name of the parameter is UGZG20RT2004J. If the parameter is unprocessed, the parameter will be named UGZG20RT2004J=UN where =UN was added to indicate that it is an unprocessed parameter. A converted value will have =CO added to the name, and a calibrated value will have =CA added to the name. Additional characters are also added to indicate Overall Status (=OStatus), Number of Samples (=Samples), and Sample Status Values (=Status). The following shows an example for a GSE parameter named UGZG20RT2004J:

UGZG20RT2004J	Original Parameter name in GSE Packet Definition
UGZG20RT2004J=UN	Unprocessed Value Name
UGZG20RT2004J=UNOStatus	Unprocessed Overall Status Name
UGZG20RT2004J=UNSamples	Unprocessed Number of Samples Name
UGZG20RT2004J= UNStatus	Unprocessed Sample Status Values Name

Recommended Reading
- TReK Metadata Tutorial
- TReK Metadata User Guide
- TReK Telemetry Database Definition Document

7.8 Command Database

The TReK 5.x Command Database is a SQLite database. The TReK 5.x Metadata application provides access to interact with a TReK 5.x database. To directly access Command Database tables you can use third party database tools available as applications or SQLite plugins that work with browsers. You can create a new database using the TReK Metadata application and populate it using a TReK 5.x ASCII file or by using files from a HOSC partial database download. The TReK 5.x ASCII file format is described in the TReK Command Database Definition document.

Recommended Reading
- TReK Metadata Tutorial
- TReK Metadata User Guide
- TReK Command Database Definition Document

8 General Steps to Migrate

1. Review the following TReK 5.x Documents

    - TReK Getting Started Guide

- TReK Concepts Document
- TReK Data Tutorial
- TReK Command Tutorial
- TReK On-Line Help for specific details on API Calls.

2. Recreate data definitions currently located in the TReK 3.x databases in a TReK 5.x database or TReK 5.x metadata file. See section 7.7 Telemetry Database and section 7.8 Comnand Database for more information.

3. Use the TReK Data and TReK Command applications to re-create telemetry and command configurations used in Telemetry Processing and Command Processing.

4. Recompile software programs that used the TReK API.

9 Additional TReK 3.x Capabilities

This section addresses TReK 3.x capabilities that have not been included in the TReK 5.x software. These capabilities are either planned for a future release of TReK 5.x or under consideration for removal. If there are any capabilities below that are currently planned not to be carried forward and you need the capability please send an e-mail to trek.help@nasa.gov.

9.1 Converting EHS Data Storage Manager Files

The capability to read EHS Data Storage Manager files is available in the TReK 5.x software. There is no longer a need to convert these files to TReK recorded data files. The TReK Playback application can be used to read files retrieved from the EHS Data Storage Manager.

9.2 Parameter Recording and Extraction

The parameter recording and extraction capabilities will be included in a later release of the TReK 5.x software.

9.3 Recorded Data Viewer

The capability to view recorded data is available. The TReK Playback application can be used to read the recorded data files and forward the data to the TReK Data application where the packet contents can be viewed in a text/hexadecimal format.

9.4 Telemetry Trainer

The Telemetry Trainer application will be included in a later release of the TReK 5.x software.

9.5 Command Trainer

The Command Trainer application will be included in a later release of the TReK 5.x software.

9.6 Custom Data Packet

Several years ago the HOSC Customer Support team polled the user community to determine how many customers were using the Custom Data Packet service. The results showed no customers were using this service. Due to the amount of work required to remove this capability from the TReK 3.x software, a decision was made to leave the TReK 3.x software as is but not to carry this capability forward in the TReK 5.x software.

9.7 Remote Services

Since the period of time in which the Remote Services capability was added to the TReK 3.x software, there have been many IT Security mandates. It is anticipated that security requirements will continue to grow more stringent making it difficult to maintain this type of capability as it was designed in the TReK 3.x software. Currently there are no plans to carry forward the Remote Services capability into the TReK 5.x software. However, if you are using this capability please send an e-mail to trek.help@nasa.gov and we will work with you to determine what can be added to the TReK 5.x software to meet your needs.

9.8 TReK Telemetry Processing API

The capabilities provided by the TReK Telemetry Processing API are available through new TReK Application Programming Interfaces. These API calls can be integrated to perform the same type of functionality provided through the Telemetry Processing API. However, these API calls are at a lower level and require multiple calls to achieve the same functionality. An integrated API similar to the Telemetry Processing API is planned for a later release of the TReK 5.x software.

10 New Capabilities since TReK 3.x

This section discusses some of the new capabilities that were not in the TReK 3.x software. Many of these are related to Ku IP Services.

10.1 CCSDS File Delivery Protocol (CFDP)

Many TReK 3.x user programs were written that used custom telemetry formats to downlink files. TReK 4.x introduced a file transfer option that alleviates the need for custom, payload-specific file transfer: CFDP. You can read more about the CFDP capability in the following documents:

- TReK CFDP Tutorial

- TReK CFDP User Guide
- TReK CFDP Console User Guide
- TReK CFDP API

10.2 Delay Tolerant Networking

Delay Tolerant Networking (DTN) was introduced in the TReK 4.x series. DTN allows users to transfer data between flight and ground assets and provides guaranteed delivery. DTN is a store and forward network that automatically handles loss of signal. For more information on DTN read the following:

- TReK DTN Tutorial
- TReK IONconfig User Guide
- TReK IONizer User Guide
- TReK Device Services API

10.3 Metadata Support

TReK 3.x metadata was always contained in a database. Entry of the data into the database was through simple user generated files, partial database downloads from the HOSC, or manual entry.

Release 5.x includes these methods of metadata entry and also includes some new features. In addition to storing data as part of a database, you can save individual commands and packets to XML files on your file system. The XML files can be used just like the database to setup the Data and Command applications.

Additional metadata generation capabilities are also provided as part of Release 5.x. Metadata can be generated from a C structure or defined using a drag and drop capability in the TReK Metadata application. It is also possible to generate metadata with user code by using the Data API.

For more information on metadata in TReK 5.x please reference the following:

- TReK Concepts Document
- TReK Metadata Tutorial
- TReK Metadata User Guide
- TReK Telemetry Database Definition
- TReK Command Database Definition
- TReK Data API