

**TREK**

**HOW TO BUILD A VISUAL C++  
COMPUTATION**

**TUTORIAL**



**November 2012**

Approved for Public Release; Distribution is Unlimited.



## TABLE OF CONTENTS

<u>PARAGRAPH</u>	<u>PAGE</u>
<b>1 What you need to know before you read this document .....</b>	<b>1</b>
<b>2 Technical Support.....</b>	<b>1</b>
<b>3 Introduction.....</b>	<b>2</b>
<b>4 Step-By-Step .....</b>	<b>3</b>
<b>Appendix A Computation Source Code.....</b>	<b>8</b>
<b>Appendix B Glossary .....</b>	<b>13</b>
<b>Appendix C Acronyms .....</b>	<b>20</b>

FIGURES

<u>FIGURE</u>	<u>PAGE</u>
Figure 1 New Dialog Box.....	3
Figure 2 Win32 Console Application - Step 1 of 1 Dialog.....	4
Figure 3 New Project Information Dialog .....	5
Figure 4 Project Settings Dialog Link Tab .....	6

## 1 What you need to know before you read this document

This tutorial assumes the following:

- You are familiar with the material in the TReK Getting Started User Guide (TREK-USER-001) and the TReK Telemetry Tutorial (TREK-USER-002).
- You are familiar with the following material in the TReK Telemetry Application Programming Interface Reference Manual (TREK-USER-027):
  - \* Sections 1 – 8
  - \* GetOneNewestConvertedIntegerValue Function Description
  - \* CreatePseudo Function Description
  - \* UpdateOneIntegerPseudo Function Description
  - \* RemovePseudo Function Description
- You are an average C or C++ programmer.
- You have some experience with Microsoft Visual C++.
- You know how to start the TReK Telemetry Processing application, add a packet to the packet list, and activate the packet. (See TReK Telemetry Processing User Guide TREK-USER-003.)
- You know how to start the TReK Telemetry Trainer application, add a packet to the packet list, and send the packet. (See TReK Telemetry Trainer User Guide TREK-USER-004.)

If you are uncomfortable with any of the items listed above, some of the terminology and concepts presented in this tutorial may be difficult to understand.

## 2 Technical Support

If you are having trouble installing the TReK software or using any of the TReK software applications, please try the following suggestions:

Read the appropriate material in the manual and/or on-line help.

Ensure that you are correctly following all instructions.

Checkout the TReK Web site at <http://trek.msfc.nasa.gov/> for Frequently Asked Questions.

If you are still unable to resolve your difficulty, please contact us for technical assistance:

TReK Help Desk E-Mail, Phone & Fax:

E-Mail:            trek.help@nasa.gov  
Telephone:        256-544-3521 (8:00 a.m. - 4:30 p.m. Central Time)  
Fax:                256-544-9353

TReK Help Desk hours are 8:00 a.m. – 4:30 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

This tutorial will show you how to use the TReK Application Programming Interface (API) to get data into a Visual C++ Win32 console application. The Win32 console application that you will build performs the following tasks:

- A pseudo telemetry parameter is created to hold an integer value.
- A for loop is used to retrieve two telemetry values once every second, add the values together, and place the result in the pseudo telemetry value.
- When the loop is exited the pseudo telemetry parameter is deleted.

The Computation project files that match the finished version of this tutorial can be found in the TReK installation directory under \examples\VisualC++\Computation directory. These files can be a good resource if you want to copy and paste the source code instead of typing it in from scratch. These files also provide an easy way to verify that you have entered the correct information. For example if you run into a compile error, check the example files and compare them to your own.

Remember to perform incremental saves as you work through the tutorial. You never know when there's going to be a power outage. ☺

Well that's about it – Have Fun!

## 4 Step-By-Step

1. Start the Visual C++ Application.
2. Go to the **File** menu and select **New...**
3. In the New dialog select the **Projects** tab. On the left side of the Projects tab there is a list of project types. You will be creating a Win32 Console Application. **Select Win32 Console Application.** On the right side of the dialog you must enter the **Project name** (Computation) and **Location** (D:\Computation). You can choose a different directory if you wish. After you have entered this information your dialog should look like the one in Figure 1. Once you are finished push **OK**.

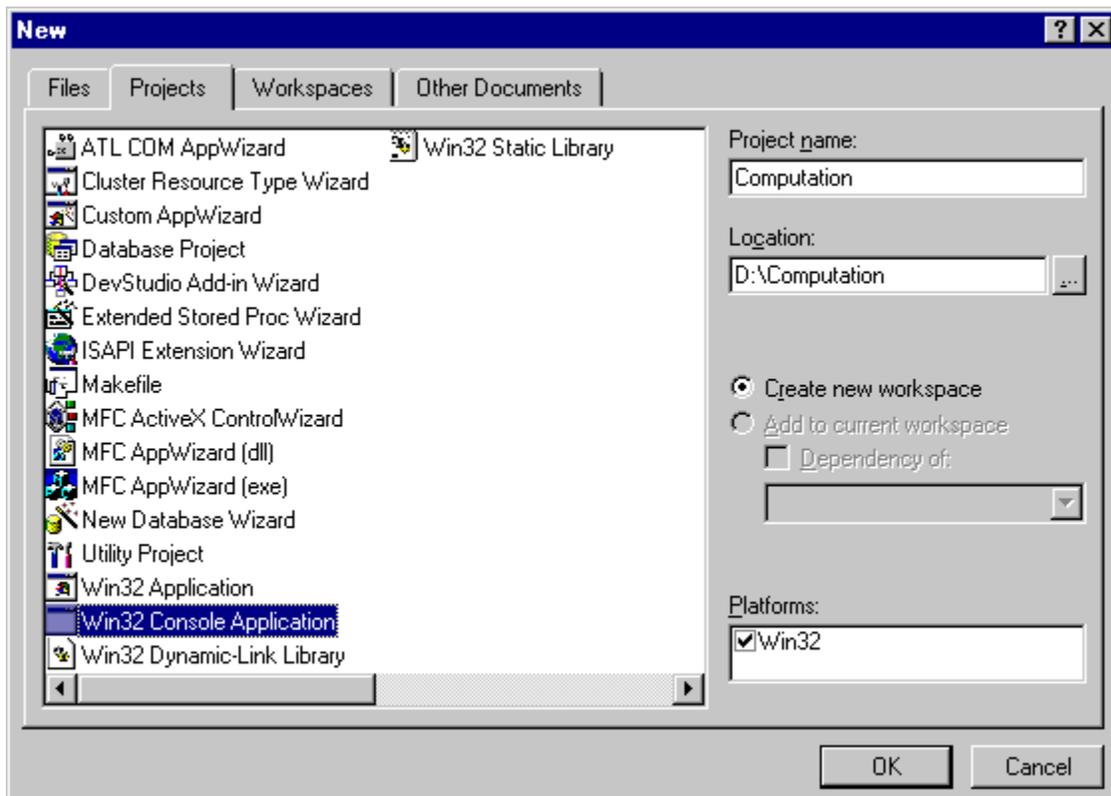


Figure 1 New Dialog Box

4. Select the option “An application that supports MFC” in the Win32 Console Application dialog shown in Figure 2. Push the **Finish** button.

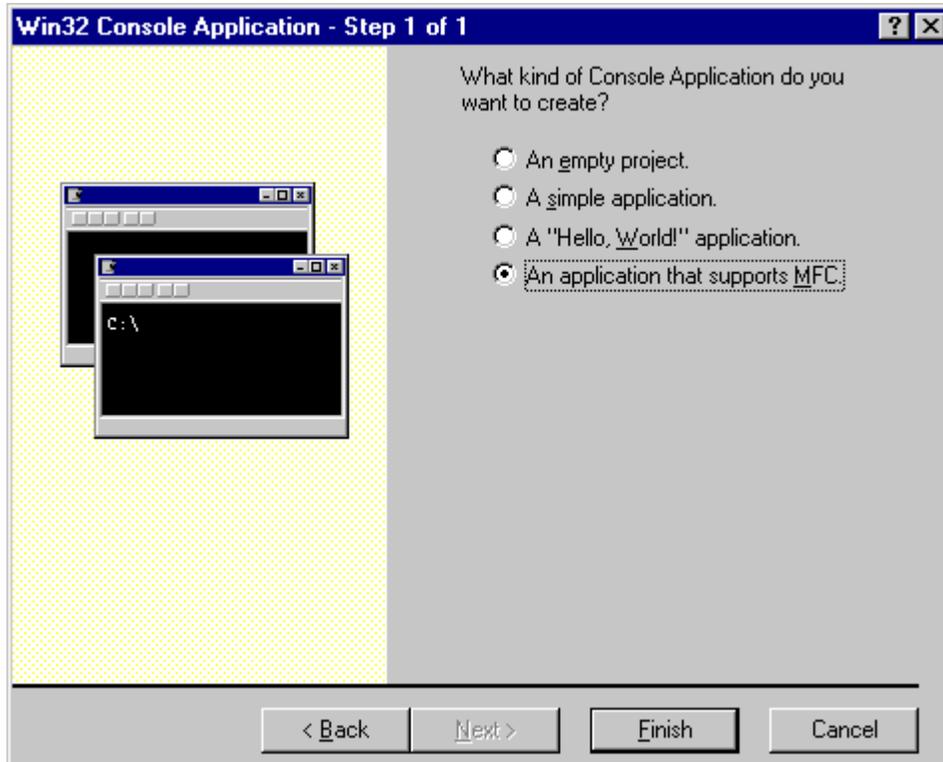
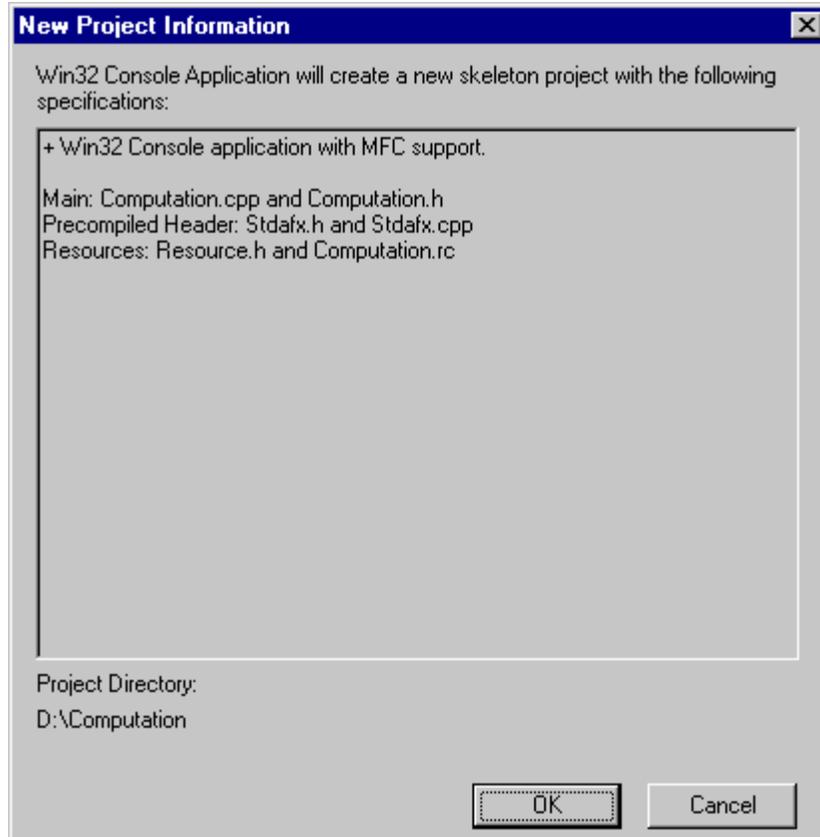


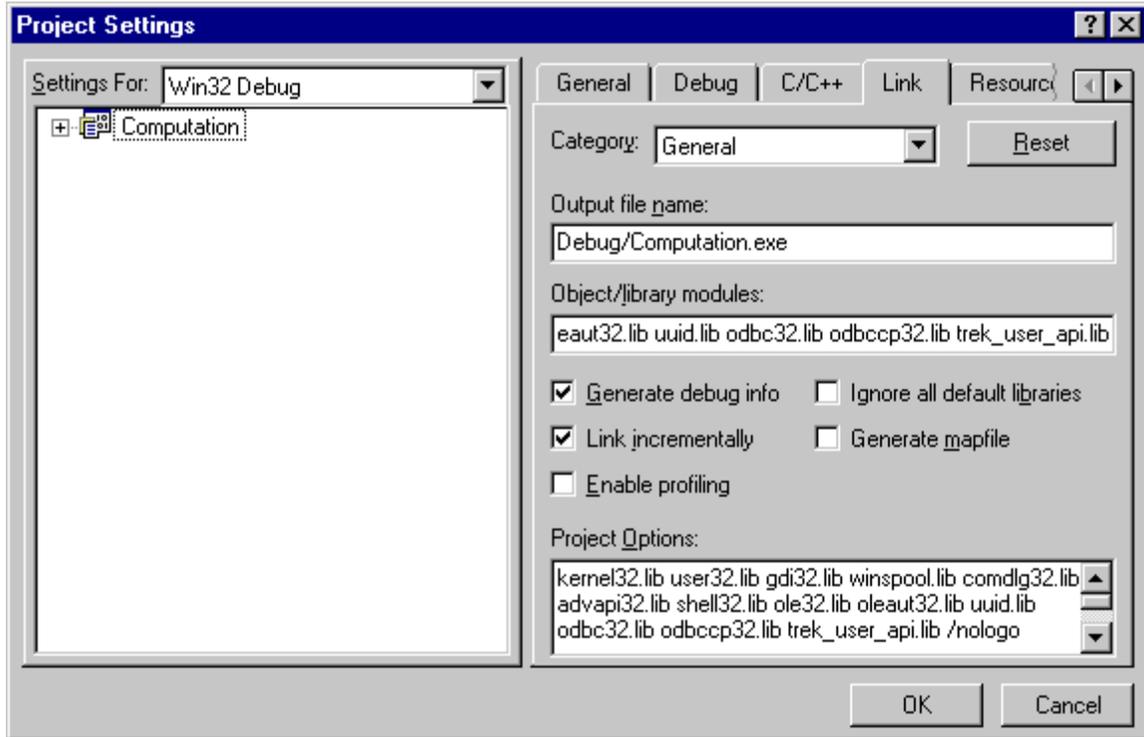
Figure 2 Win32 Console Application - Step 1 of 1 Dialog

5. You will see New Project Information dialog as shown in Figure 3. Push the **OK** button.



**Figure 3 New Project Information Dialog**

6. In order to use the TReK Application Programming Interface (API) library you need to tell Visual C++ that you want this library to be linked into your application. The next few steps you perform will tell Visual C++ that you want to link in the TReK API library.
7. First you need to copy the `trek_user_api.lib` file into your project directory. This file is located in the TReK installation directory under `lib`. Copy the `trek_user_api.lib` file into your Computation project directory (`D:\Computation` or the directory you chose). If you have worked with libraries before then you know that there is a corresponding `trek_user_api.dll` file. This file was installed in your `winnt\system32` directory when you installed the TReK software. Visual C++ knows how to find it so you don't need to do anything about the `trek_user_api.dll` file.
8. In Visual C++, go to the **Project** menu and choose **Settings...** The dialog shown in Figure 4 will appear. In the Project Settings dialog select the **Link** tab. In the **Object/library modules** field enter the location of the `trek_user_api.lib` file at the end of the edit box (click the Object/library modules edit box and push the "End" key on your keyboard). Since you copied this file into your local directory you should enter **`trek_user_api.lib`**. Figure 4 shows an example of what your dialog should look like now. When you're done push the **OK** button to exit the dialog.



**Figure 4 Project Settings Dialog Link Tab**

9. There is one more step you need to perform in order to use the TReK API. The API relies on several header files. You need to copy these header files into your Computation project directory. These files are located in the TReK installation directory under include. You should copy the `trek.h`, `trek_error.h`, and `trek_user_api.h` files into your Computation project directory (`D:\Computation`).
10. Now you need to add the source code for your computation. You can either replace the source code in your `Computation.cpp` file with the code in Appendix A or you can type in the code yourself. If you choose to type in the code there are two sections you will need. The first is at the beginning of the file and is marked with a comment called “TReK Include Files.” The second section is marked with a comment called “TReK Computation Example Code Begins Here.”
11. Now you are ready to compile and build your computation. Go to the Visual C++ **Build** menu and select **Build Computation.exe**. If you run into any computation errors that you cannot easily solve, compare your code to the finished product in the examples directory in the TReK installation to locate any inconsistencies. If you encounter any Linking errors this may mean Visual C++ is having trouble locating the `trek_user_api.lib` file. Make sure this file is in your Computation project directory.

12. Before you run your computation, you need to set up your TReK system so Packet ID 7 is being sent and received. This is because Packet ID 7 contains MSID053 and MSID055. These are the two MSIDs that your computation will retrieve once a second and add together. To set up your TReK system for Packet ID 7, perform the following steps:
  - a. Start the Telemetry Processing application.
  - b. Add Packet ID 7 to the packet list.
  - c. Activate Packet ID 7.
  - d. Start the Telemetry Trainer application.
  - e. Add Packet ID 7 to the packet list. Set the Run Time to 120 seconds.
  - f. Send Packet ID 7.

After you have performed the steps above, go to the Visual C++ **Build** menu and choose **Execute Computation.exe** to run your computation. Your computation will loop 20 times and retrieve MSID053 and MSID055 during each loop. If the API returns any return code other than SUCCESS, then a message will be printed. The computation will print out the API return code as a string so you can see why the API call failed. Some of the reasons this can occur are: there was an error calling one of the API functions, there was no data available for MSID053 or MSID055, or the API could not find MSID053 or MSID055.

When you write a program, you will probably add quite a bit more error checking than was shown in this tutorial. For instance you would probably want to check to make sure the telemetry pseudo parameter was created successfully and also removed successfully. In this tutorial, these checks were not performed in order to make the program as simple as possible. Please see the TReK Application Programming Interface Reference Manual for more information about API return codes and what they mean.

## Appendix A Computation Source Code

```

// Computation.cpp : Defines the entry point for the console
// application.

#include "stdafx.h"
#include "Computation.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

//
//   TReK Include Files
//

#include "trek.h"
#include "trek_error.h"
#include "trek_user_api.h"

//   End TReK Include Files

////////////////////////////////////
/////
// The one and only application object

CWinApp theApp;

using namespace std;

int _tmain(int argc, TCHAR* argv[], TCHAR* envp[])
{
    int nRetCode = 0;

    // initialize MFC and print and error on failure
    if(!AfxWinInit(::GetModuleHandle(NULL),NULL,::GetCommandLine(), 0))
    {
        // TODO: change error code to suit your needs
        cerr << _T("Fatal Error: MFC initialization failed") << endl;
        nRetCode = 1;
    }
    else
    {
        // TODO: code your application's behavior here.

        //
        //   TReK Computation Example Code Begins Here
        //

```

```

char comp_pseudo[20];           // Integer Pseudo Name
long msid053_value;           // MSID053's value
long msid055_value;           // MSID055's value
long msid053_token[TOKEN_SIZE]; // MSID053's token
long msid055_token[TOKEN_SIZE]; // MSID055's token
long pseudo_value;           // Pseudo's value
char status[STATUS_LEN];     // Status string
int return_code;             // Return code from TReK API
int msid053_return_code;     // Return code from TReK API
int msid055_return_code;     // Return code from TReK API
char message[70];           // API return code as string
int i;                       // Counter Variable

//
//   Initialize the comp_pseudo variable and create an integer
//   pseudo parameter.
//

strcpy( comp_pseudo, "comp_pseudo" );

return_code = CreatePseudo( comp_pseudo, INTEGER_DATATYPE,1,0);

//
//   Initialize the tokens
//

for( i = 0; i < TOKEN_SIZE; i++ )
{
    msid053_token[i] = 0;
    msid055_token[i] = 0;
}

//
//   Loop 20 times.  Get the two parameters add them together
//   and update the pseudo parameter's value with the result.
//

for( i = 0; i < 20; i++ )
{
    msid053_return_code = GetOneNewestConvertedIntegerValue(
                                                PDSS_PAYLOAD,
                                                "MSID053",
                                                "",
                                                REAL_TIME,
                                                NO_SENSE,
                                                msid053_token,
                                                &msid053_value,
                                                status );

    msid055_return_code = GetOneNewestConvertedIntegerValue(
                                                PDSS_PAYLOAD,
                                                "MSID055",
                                                "",
                                                REAL_TIME,

```

```
NO_SENSE,  
msid055_token,  
&msid055_value,  
status );
```

```

//
//   If both calls succeeded, add the values and update
//   the pseudo parameter.  Otherwise, write an error
//   message.
//
//
if( (msid053_return_code == SUCCESS) &&
    (msid055_return_code == SUCCESS) )
{
    pseudo_value = msid053_value + msid055_value;
    printf( "pseudo_value = %d\n", pseudo_value );
    return_code = UpdateOneIntegerPseudo( comp_pseudo,
                                          pseudo_value );
}
else
{
    if( msid053_return_code != SUCCESS )
    {
        GetAPIReturnCodeAsString( msid053_return_code,
                                  70,
                                  message );
        printf( "Error getting value for MSID053:  %s\n",
               message );
    }
    if( msid055_return_code != SUCCESS )
    {
        GetAPIReturnCodeAsString( msid055_return_code,
                                  70,
                                  message );
        printf( "Error getting value for MSID055:  %s\n",
               message );
    }
}

//
//   Wait one second before getting next value
//

Sleep(1000);

}

//
//   Delete the pseudo parameter.
//

return_code = RemovePseudo( comp_pseudo );

//   End of TReK Computation Example Code
}

return nRetCode;
}

```



## Appendix B Glossary

Note: This Glossary is global to all TReK documentation. All entries listed may not be referenced within this document.

Application Programming Interface (API)	A set of functions used by an application program to provide access to a system's capabilities.
Application Process Identifier (APID)	An 11-bit field in the CCSDS primary packet header that identifies the source-destination pair for ISS packets. The type bit in the primary header tells you whether the APID is a payload or system source-destination.
Calibration	The transformation of a parameter to a desired physical unit or text state code.
Communications Outage Recorder	System that captures and stores payload science, health and status, and ancillary data during TDRSS zone of exclusion.
Consultative Committee for Space Data Systems (CCSDS) format	Data formatted in accordance with recommendations or standards of the CCSDS.
Consultative Committee for Space Data Systems (CCSDS) packet	A source packet comprised of a 6-octet CCSDS defined primary header followed by an optional secondary header and source data, which together may not exceed 65535 octets.
Conversion	Transformation of downlinked spacecraft data types to ground system platform data types.
Custom Data Packet	A packet containing a subset of parameters that can be selected by the user at the time of request.
Cyclic Display Update Mode	A continuous update of parameters for a particular display.
Decommutation (Decom)	Extraction of a parameter from telemetry.
Discrete Values	Telemetry values that have states (e.g., on or off).

Dump	During periods when communications with the spacecraft are unavailable, data is recorded onboard and played back during the next period when communications resume. This data, as it is being recorded onboard, is encoded with an onboard embedded time and is referred to as dump data.
Enhanced HOSC System (EHS)	Upgraded support capabilities of the HOSC systems to provide multi-functional support for multiple projects. It incorporates all systems required to perform data acquisition and distribution, telemetry processing, command services, database services, mission support services, and system monitor and control services.
Exception Monitoring	A background process capable of continuously monitoring selected parameters for Limit or Expected State violations. Violation notification is provided through a text message.
Expected State Sensing	Process of detecting a text state code generator in an off-nominal state.
EXPRESS	An EXPRESS Rack is a standardized payload rack system that transports, stores and supports experiments aboard the International Space Station. EXPRESS stands for EXpedite the PRocessing of Experiments to the Space Station.
File transfer protocol (ftp)	Protocol to deliver file-structured information from one host to another.
Flight ancillary data	A set of selected core system data and payload health and status data collected by the USOS Payload MDM, used by experimenters to interpret payload experiment results.

Grayed out	Refers to a menu item that has been made insensitive, which is visually shown by making the menu text gray rather than black. Items that are grayed out are not currently available.
Greenwich Mean Time (GMT)	The solar time for the meridian passing through Greenwich, England. It is used as a basis for calculating time throughout most of the world.
Ground ancillary data	A set of selected core system data and payload health and status data collected by the POIC, which is used by experimenters to interpret payload experiment results. Ground Ancillary Data can also contain computed parameters (pseudos).
Ground receipt time	Time of packet origination. The time from the IRIG-B time signal received.
Ground Support Equipment (GSE)	GSE refers to equipment that is brought in by the user (i.e. equipment that is not provided by the POIC).
Ground Support Equipment Packet	A CCSDS Packet that contains data extracted from any of the data processed by the Supporting Facility and the format of the packet is defined in the Supporting Facility's telemetry database.
Huntsville Operations Support Center (HOSC)	A facility located at the Marshall Space Flight Center (MSFC) that provides scientists and engineers the tools necessary for monitoring, commanding, and controlling various elements of space vehicle, payload, and science experiments. Support consists of real-time operations planning and analysis, inter- and intra-center ground operations coordination, facility and data system resource planning and scheduling, data systems monitor and control operations, and data flow coordination.

IMAQ ASCII	A packet type that was added to TReK to support a very specific application related to NASA's Return to Flight activities. It is not applicable to ISS. It is used to interface with an infrared camera that communicates via ASCII data.
Limit Sensing	Process of detecting caution and warning conditions for a parameter with a numerical value.
Line Outage Recorder Playback	A capability provided by White Sands Complex (WSC) to play back tapes generated at WSC during ground system communication outages.
Measurement Stimulus Identifier (MSID)	Equivalent to a parameter.
Monitoring	A parameter value is checked for sensing violations. A message is generated if the value is out of limits or out of an expected state.
Parameter	TReK uses the generic term parameter to mean any piece of data within a packet. Sometimes called a measurement or MSID in POIC terminology.
Payload Data Library (PDL)	An application that provides the interface for the user to specify which capabilities and requirements are needed to command and control his payload.
Payload Data Services Systems (PDSS)	The data distribution system for ISS. Able to route data based upon user to any of a number of destinations.
Payload Health and Status Data	Information originating at a payload that reveals the payload's operational condition, resource usage, and its safety/anomaly conditions that could result in damage to the payload, its environment or the crew.
Payload Operations Integration Center (POIC)	Manages the execution of on-orbit ISS payloads and payload support systems in coordination/unison with distributed International Partner Payload Control Centers, Telescience Support Centers (TSC's) and payload-unique remote facilities.

Payload Rack Checkout Unit (PRCU)	The Payload Rack Checkout Unit is used to verify payload to International Space Station interfaces for U.S. Payloads.
Playback	Data retrieved from some recording medium and transmitted to one or more users.
Pseudo Telemetry (pseudo data)	Values that are created from calculations instead of directly transported telemetry data. This pseudo data can be created from computations or scripts and can be displayed on the local PC.
Remotely Generated Command	A command sent by a remote user whose content is in a raw bit pattern format. The commands differ from predefined or modifiable commands in that the content is not stored in the POIC Project Command Database (PCDB).
Science data	Sensor or computational data generated by payloads for the purpose of conducting scientific experiments.
Subset	A collection of parameters from the total parameter set that is bounded as an integer number of octets but does not constitute the packet itself. A mini-packet.
Super sampled	A parameter is super sampled if it occurs more than once in a packet.
Swap Type	A flag in the Parameter Table of the TReK database that indicates if the specified datatype is byte swapped (B), word swapped (W), byte and word swapped (X), byte reversal (R), word reversal (V) or has no swapping (N).
Switching	A parameter's value can be used to switch between different calibration and sensing sets. There are two types of switching on TReK: range and state code.

Transmission Control Protocol (TCP)	TCP is a connection-oriented protocol that guarantees delivery of data.
Transmission Control Protocol (TCP) Client	A TCP Client initiates the TCP connection to connect to the other party.
Transmission Control Protocol (TCP) Server	A TCP Server waits for (and accepts connections from) the other party.
Telemetry	Transmission of data collected from a source in space to a ground support facility. Telemetry is downlink only.
Telescience Support Center (TSC)	A TSC is a NASA funded facility that provides the capability to plan and operate on-orbit facility class payloads and experiments, other payloads and experiments, and instruments.
User Application	Any end-user developed software program that uses the TReK Application Programming Interface software. Used synonymously with User Product.
User Data Summary Message (UDSM)	Packet type sent by PDSS that contains information on the number of packets sent during a given time frame for a PDSS Payload packet. For details on UDSM packets, see the POIC to Generic User IDD (SSP-50305).
Uplink format	The bit pattern of the command or file uplinked.
User Datagram Protocol (UDP)	UDP is a connection-less oriented protocol that does not guarantee delivery of data. In the TCP/IP protocol suite, the UDP provides the primary mechanism that application programs use to send datagrams to other application programs. In addition to the data sent, each UDP message contains both a destination port number and a fully qualified source and destination addresses making it possible for the UDP software on the destination to deliver the message to the correct recipient process and for the recipient process to send a reply.

User Product	Any end-user developed software program that uses the TReK Application Programming Interface software. Used synonymously with User Application.
Web	Term used to indicate access via HTTP protocol; also referred to as the World Wide Web (WWW).

## Appendix C Acronyms

Note: This acronym list is global to all TReK documentation. Some acronyms listed may not be referenced within this document.

AOS	Acquisition of Signal
API	Application Programming Interface
APID	Application Process Identifier
ASCII	American Standard Code for Information Interchange
CAR	Command Acceptance Response
CAR1	First Command Acceptance Response
CAR2	Second Command Acceptance Response
CCSDS	Consultative Committee for Space Data Systems
CDB	Command Database
CDP	Custom Data Packet
COR	Communication Outage Recorder
COTS	Commercial-off-the-shelf
CRR	Command Reaction Response
DSM	Data Storage Manager
EHS	Enhanced Huntsville Operations Support Center (HOSC)
ERIS	EHS Remote Interface System
ERR	EHS Receipt Response
EXPRESS	Expediting the Process of Experiments to the Space Station
ES	Expected State
FAQ	Frequently Asked Question
FDP	Functionally Distributed Processor
FSV	Flight System Verifier
FSV1	First Flight System Verifier
FSV2	Second Flight System Verifier
FPD	Flight Projects Directorate
FTP	File Transfer Protocol
GMT	Greenwich Mean Time
GRT	Ground Receipt Time
GSE	Ground Support Equipment
HOSC	Huntsville Operations Support Center
ICD	Interface Control Document
IMAQ ASCII	Image Acquisition ASCII
IP	Internet Protocol
ISS	International Space Station
LDP	Logical Data Path
LES	Limit/Expected State
LOR	Line Outage Recorder
LOS	Loss of Signal
MCC-H	Mission Control Center – Houston
MOP	Mission, Operational Support Mode, and Project
MSFC	Marshall Space Flight Center
MSID	Measurement Stimulus Identifier

NASA	National Aeronautics and Space Administration
OCDB	Operational Command Database
OS	Operating System
PC	Personal Computer, also Polynomial Coefficient
PCDB	POIC Project Command Database
PDL	Payload Data Library
PDSS	Payload Data Services System
PGUIDD	POIC to Generic User Interface Definition Document
POIC	Payload Operations Integration Center
PP	Point Pair
PRCU	Payload Rack Checkout Unit
PSIV	Payload Software Integration and Verification
RPSM	Retrieval Processing Summary Message
SC	State Code
SCS	Suitcase Simulator
SSP	Space Station Program
SSCC	Space Station Control Center
SSPF	Space Station Processing Facility
TCP	Transmission Control Protocol
TReK	Telescience Resource Kit
TRR	TReK Receipt Response
TSC	Telescience Support Center
UDP	User Datagram Protocol
UDSM	User Data Summary Message
URL	Uniform Resource Locator
USOS	United States On-Orbit Segment
VCDU	Virtual Channel Data Unit
VCR	Video Cassette Recorder
VPN	Virtual Private Network