SPIRE-PAC-DOC-002293

EGSE Tools

User Manual

Document-ID: PICC-ME-MN-004
Issue: 2.1
Date: November 17, 2003
Author: Erich Wiezorrek

Table of Contents

0 General

For all the tools described below you must logged in as user "sops23e" or "hpops" in the host running SCOS2000.

Most of the tools are using a property file which is assigned with the environment variable EGSE_PROPS. This environment variable is set during login and usually must not be changed.

Properties specify kind of default parameters like router connection details. If for example a second router instance is used (which must use a different port number) a different property file must be assigned by the environment variable EGSE_PROPS.

1 Router and SCOS2000 Gateway

1.1 Standard Setup

To start the router and the SCOS2000 gateway in their standard configuration use the command:

```
% StartRouter
```

This will create two Xterminal windows on their default screen to log the console output of both applications. The router server port will be 9877 on the host running SCOS2000.

1.2 Special Router Usage

Sometimes another instance of the router might be necessary supplying a different router server port.

```
% Router portNumber
```

The *portNumber* argument specifies the new (additional) router server port.

2 TeiGateway

To start the test equipment interface type

```
% TeiGateway
```

This will create one Xterminal windows on its default screen to log the console output.

3 PacketRecorder

This is a tool to record PUS packets received from the router into a special archive file. Which PUS packets are requested from the router is selected via APIDs which are specified either in the command arguments or in the property file. To replay the PUS packet archive file the PacketPlayer tool can be used.

```
% PacketRecorder ArchiveFileName [TmApidList] [TcApidList]
```

Arguments:

ArchiveFileName: Filename of the PUS packet archive file

TmApidList: Optional list of comma separated APIDs for which TM packets shall be recorded. If not specified, the property file defines them.

TcApidList: Optional list of comma separated APIDs for which TC packets shall be

recorded. If not specified, the property file defines them.

Properties:

PacketRecorder.routerHostName: host name where the router resides

PacketRecorder.routerPortNumber: port number on which the router listens

PacketRecorder.routerClientName: client name used by the router for this application

PacketRecorder.TmList: List of comma separated APIDs for which TM packets shall be recorded

PacketRecorder.TcList: List of comma separated APIDs for which TC packets shall be recorded

The PUS packet archive file has following format:

Every file consist of a sequence of records. Each record consist of
8 bytes time field
4 bytes length
" length" bytes PUS packet

The time field holds the time when the packet was recorded. It is formatted as the number of milliseconds since midnight, January 1, 1970 (UTC).

The length field holds the number of bytes for the following PUS packet.

The PUS packet field is the unmodified PUS packet.

Please note that isn't guaranteed that the very last record is complete.

4 PacketPlayer

The PacketPlayer can be used to replay PUS packet archive files generated by the PacketRecorder tool.

```
% PacketPlayer ArchiveFileName
```

Arguments:

ArchiveFileName: Filename of the PUS packet archive file

Properties:

PacketPlayer.routerHostName: host name where the router resides

PacketPlayer.routerPortNumber: port number on which the router listens

PacketPlayer.routerClientName: client name used by the router for this application

5 TmPacketDumper

The TmPacketDumper prints a hexdump of PUS TM packets on the terminal window. The PUS packets are received from the router or read from a PacketRecorder file. For the router interface the APIDs selection is mandatory. If no APIDs are selected for the file interface all packets will be dumped. The -t option is only valid for the TM file

interface.

```
% TmPacketDumper [-v | -q | -h | -t | -c n | -f fileName | -a apids]
```

Arguments:

Apid: Pus packet apid of the packets to be dumped

Options:

-f name of PacketRecorder file to read TM packets from

-a comma separated list of APIDs to be dumped, ranges like 1152-1157 are allowed

-c n number of 16 bit words out of the data filed to be printed

-t dump record time stamp and size of TM packet

-v verbose output

-q be quiet

-h help, prints this message

Properties:

TmPacketDumper.router.hostname: host name where the router resides

TmPacketDumper.router.portnumber: port number on which the router listens

TmPacketDumper.router.clientname: client name used by the router for this application (defaults to "TmPacketDumper")

6 TcPacketDumper

The TcPacketDumper prints a hexdump of PUS TC packets on the terminal window. The PUS packets are received from the router or read from a PacketRecorder file. For the router interface the APIDs selection is mandatory. If no APIDs are selected for the file interface all packets will be dumped. The -t option is only valid for the TM file interface.

```
% TcPacketDumper [-v | -q | -h | -t | -c n | -f fileName | -a apids]
```

Arguments:

Apid: Pus packet apid of the packets to be dumped

Options:

-f name of PacketRecorder file to read TM packets from

-a comma separated list of APIDs to be dumped, ranges like 1152-1157 are allowed

-c n number of 16 bit words out of the data filed to be printed

-t dump record time stamp and size of TM packet

-v verbose output

-q be quiet

-h help, prints this message

Properties:

TcPacketDumper.router.hostname: host name where the router resides

TcPacketDumper.router.portnumber: port number on which the router listens

TcPacketDumper.router.clientname: client name used by the router for this application (defaults to "TcPacketDumper")

7 PacketGenerator

% PacketGenerator [-v | -q | -h] PusType Apid Type SubType Data

Arguments:

PusType: either "TC" for telecommand or "TM" for telemetry packets

Apid: APID of the PUS packet destination

Type: the packet type

SubType: the packet subtype

Data: a sequence of 16 bit words entered in hexadecimal notation

Options:

-*v* verbose output

-*q* be quiet

-*h* help, prints this message

Properties:

PacketGenerator.router.hostname: host name where the router resides

PacketGenerator.router.portnumber: port number on which the router listens

PacketGenerator.router.clientname: client name used by the router for this application (defaults to "PacketGenerator")

Example:

PacketGenerator -v TC 2016 8 2 100 23ff 0 abcd

This will send a (8,2) telecommand packet to the application 2016 with four 16 bit words (0x0100, 0x23ff, 0x0000, 0xabcd) in the packet application data field.

8 Data Rate Monitor

This is a tool to display the actual date rate of TM packets delivered by the router.

dataratemonitor apidList

Arguments:

a*pidList*: The list of susbsystems (specified by the APID) to be included in the data rate monitor list.

The apidList is a comma separated list of APIDs or APID ranges. An APID range is specified by the lower APID (inclusive), the dash ("-") characters and the upper APID (inclusive). All APIDs must be specified in decimal numbers.

Example:

```
datarateмonitor 1,4,5,1152-2017
```

This requests the display of the data rates for the APIDs 1, 4 and 5 and all APIDs from 1152 till 2017.

9 Cyclic Redundancy Check Code Calculator

This tools allows to calculate the CRC over a range of numbers. There are two modes of operation available:

a graphical interface mode
a command line mode.

In the command line mode the CRC calculator accepts a number of 16 bit words and returns the calculated CRC number on its standard output. In the GUI mode a window will pop to gather the user input and to return the CRC.

```
crccalc -gui
```

or

```
crccalc n1 [n2 ...]
```

Arguments:

n1...	A list of 16 bit integers specified in hexadecimal radix.

Options:

-gui	Run the CRC calculator in the graphical user interface mode