Satellite Control Software (SCS) EGSE Router Infrastructure ICD

Prepared by:
Florian George
Stephane Billeter

Swiss Space Center EPFL
Lausanne
Switzerland

06 November 2013

RECORD OF REVISIONS

ISS/REV	Date	Modifications	Created/modified by
1/0	27/04/2010	Initial issue.	Florian George
2/0	06/11/2013	Updated for QB50 and release ICD for CDR	Stéphane Billeter / Muriel Richard

ABBREVIATED TERMS

Ack	Acknowledgement
CCSDS	Consultative Committee for Space Data Systems
COM	Component Object Model
ECSS	European Cooperation for Space Standardization
EGSE	Electrical Ground-Support Equipment
ESA	European Space Agency
ESTEC	European Space Research and Technology Centre
MCS	Mission Control System
Nack	Negative Acknowledgment
OSI	Open Systems Interconnection
PUS	Packet Utilisation Standard (see [N1])
S/C	Spacecraft
SCOE	Special Checkout Equipment
TC	Telecommand
TM	Telemetry
VC	Virtual Channel

INTRODUCTION

This ICD describes the EGSE Router infrastructure and the EGSE Router Application Protocol used for interactions between the ground segment components such as the Mission Control System, the TM/TC Front End, ground stations, test equipment, etc. The EGSE Router routes the messages between the components using the EGSE Router Application Protocol on top TCP/IP connections.

The EGSE Router was originally a COM-based Windows application developed at ESA/ESTEC with a TCP/IP gateway added subsequently. The EGSE Router was later reimplemented to be TCP/IP-only. However the two layers were kept as separate protocol specifications. This specification brings them together making no breaking change to the protocol. In the process, some messages and fields were renamed for better clarity, but the protocol stays binary-compatible with the ESA/ESTEC TCP/IP EGSE Router implementation.

Figure 1 shows where the EGSE Router Application Protocol is situated in the OSI Model[1]:

OSI Model Layer	EGSE Router Infrastructure	
7. Application	Data Types	EGSE Router Application Protocol
6. Presentation	EGSE Router Messages	
5. Session		
4. Transport	TCP	
3. Network	IP	
2. Data Link	*Any supported by the upper layer*	
1. Physical	*Any supported by the upper layer*	

Figure 1 - EGSE Router Infrastructure alongside the OSI Model

As shown, the EGSE Router Application Protocol is composed of two layers: EGSE Router Messages and Data Types.

The first, EGSE Router Messages, defines the format of the messages and connection management functionalities such as registering/unregistering with the EGSE Router as well as transferring data to other EGSE Clients.

The second layer, Data Types, specifies the exact usage of the messages' data when transferring data between EGSE Client, depending on the data type (e.g. telecommand/telemetry packets, telemetry frames, ground commands, etc.).

[1] OSI Model: http://en.wikipedia.org/wiki/OSI_Model

1 GROUND SEGMENT ARCHITECTURE

Figure 2 shows a logical view of a conventional ground segment architecture. It includes:

- a Mission Control System,
- a TM/TC Front End,
- Ground Stations,
- other components such as test equipment and payload data archiving and processing.

These components are detailed afterwards. The EGSE Router Infrastructure enables them to interact by specifying a common communication protocol for their needs. Figure 3 shows that physically, in contrast to the logical view, the components are all only connected to the EGSE Router which takes care of the data routing. This simplifies both the development and deployment of the components as they only communicate through a single interface and need to know the network location of the EGSE Router alone. Therefore components located behind firewalls preventing inbound connections have no problem communicating with the others components.

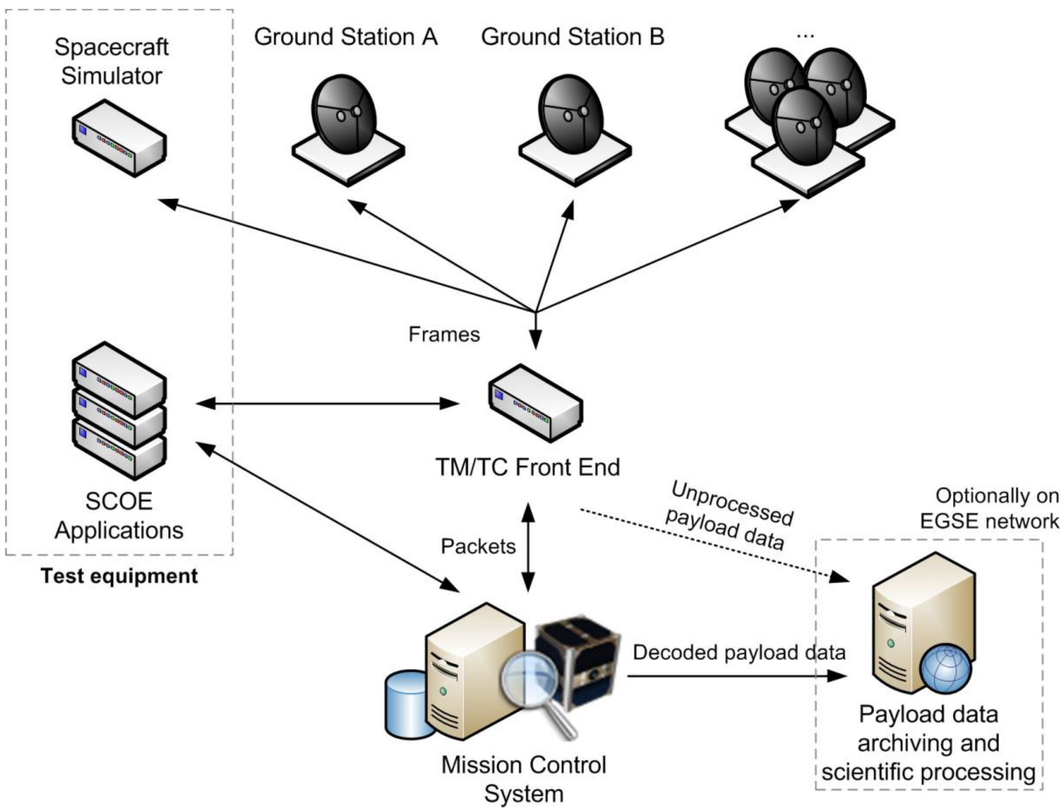


Figure 2 - Logical view of example Ground Segment

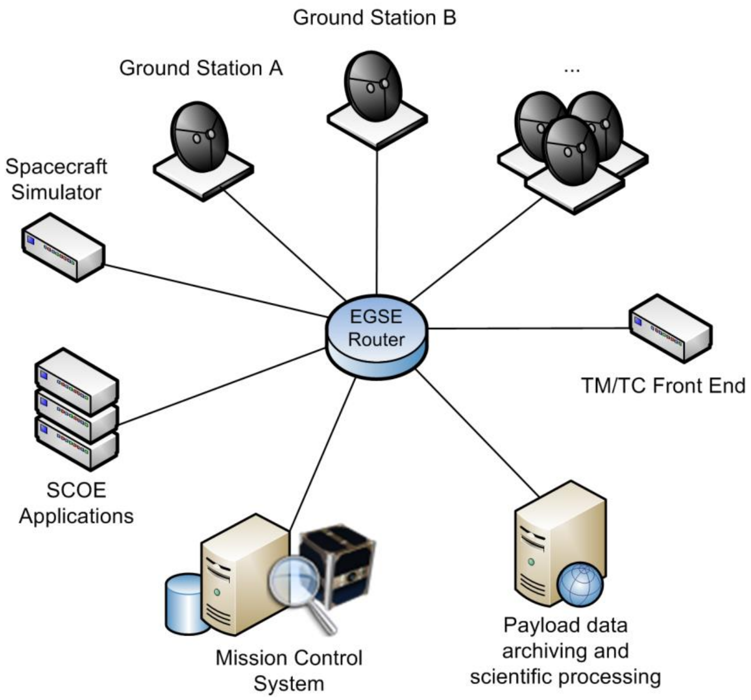


Figure 3 - Physical view of same Ground Segment interconnected through EGSE Router

1.1 Mission Control System

The Mission Control System is the central part of the ground segment. It performs all the packets data processing required to monitor the health of the spacecraft and ground segment, generate telecommand packets and track their execution progress, extract payload data, etc. It also stores all this processed data for archiving and/or post-processing.

1.2 TM/TC Front End

The role of the TM/TC Front End is to provide the layer between the Mission Control System (MCS) which is packet-based and the ground stations working at frame-level. This layer contains the handling of all the features of the frame-based protocol used on the space link; such as packet reconstruction and encapsulation, telecommand onboard arrival detection, telecommand transmission retrial, etc. This layer may also include the archiving of raw telemetry and replay functionality of data (e.g. send the payload data on-demand after the end of the pass not to overload he MCS during the pass with data that it doesn't need in real-time).

1.3 Ground Stations

The ground stations are the edge of the ground segment interfacing directly to the space segment using radio frequencies as physical link. These ground stations can be on-site or located anywhere on the globe.

A ground station is composed of two parts. The first is the hardware (antennas, transceivers, etc.) and its control software which can be specific to each ground station. The second is the interfacing software which interfaces the ground station with the rest of the ground segment. This part is very important as it lets ground stations expose a standard interface allowing new ground stations to be easily added to the ground segment.

1.4 Other equipment

Many other components can be part of the ground segment and connected though the EGSE Router infrastructure. This can include simulators (e.g. spacecraft simulator) and SCOE Applications during testing or gateway to other networks or ground segments (e.g. GENSO) during operations.

2 ESGE Router Application Protocol

The EGSE Router Application Protocol defines the types and format of the exchanged messages between the EGSE Router and the EGSE Clients. Figure 4 shows the complete life-cycle of an EGSE connection in nominal operations. That includes establishing the TCP/IP connection to the EGSE Router, registering the EGSE Client with a *RegisterClient* command, sending data to other clients using *SendData* messages, receiving data with *ReceiveData* messages and finally unregistering the client before disconnecting.

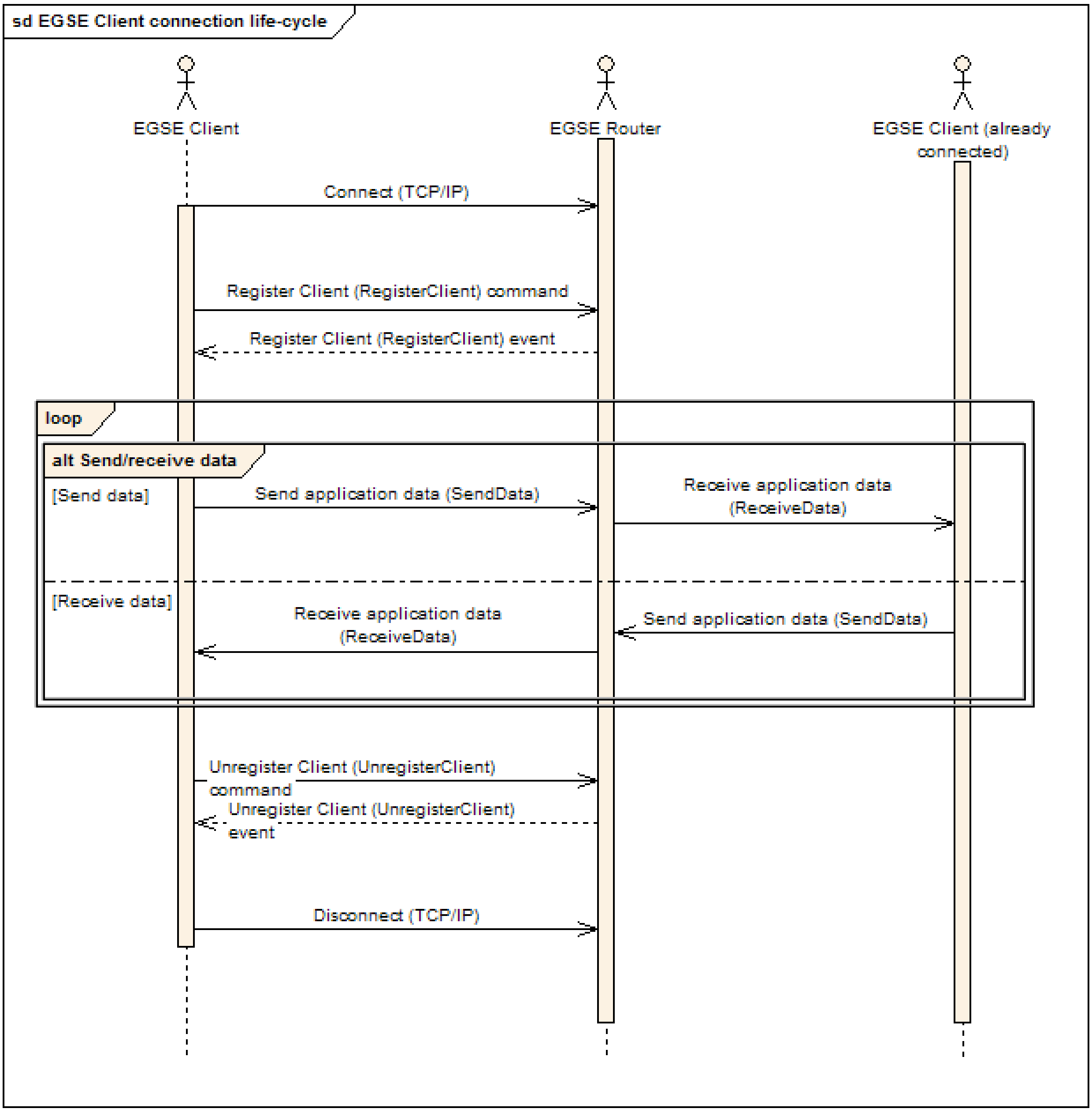


Figure 4 - Complete life-cycle of an EGSE Client using the EGSE Router

2.1 Connection to the EGSE Router

The EGSE Router listens on a TCP/IP socket, by default on port 9876, for incoming connections. The specifics on how to establish the TCP/IP connection to the EGSE Router are outside the scope of this document and specific to the client platform.

Once connected, as many Clients as required may be registered with the EGSE Router on the same connection (usually a single one); with the constraint that both the Client ID and Client Name must be unique across all connections. It is the project's responsibility to define a policy which satisfies this constraint. A Client **must** be registered to the EGSE Router (using a *RegisterClient* message) before it can transfer data to other Clients (i.e. use *SendData*).

2.2 Client ID on the EGSE Router

To communicate through the EGSE Router, each element of the ground segment must have a unique ID. A Client ID is 16-bit number with the first bit reserved for the EGSE Router itself and future expansions of the protocol; therefore leaving the useable part of the Client ID as a 15-bit number.

EGSE Client	Client ID	
User-defined range	0x0001 - 0xEFFF	(1 to 61439)
ESGE Router	0xF000	(61440)
Reserved	0xF001- 0xFFFE	(61440 to 65534)
Broadcast	0xFFFF	(65535)

Table 1 - Client ID usage

The Client ID *0xF000* is the ID of the EGSE Router. The control messages (ex: *RegisterClient*) have this Client ID in their *Destination ID* field. The errors from the EGSE Router have this Client ID in their *Source ID* field.

The Client ID *0xFFFF* is used for broadcast messages when used as destination. Every client connected to the EGSE Router will receive them. If needed, clients can detect broadcasted messages by checking the *Destination ID* field value.

2.3 Messages categories

Two categories of messages exist depending upon the purpose of the message; the specifications of the messages are found in the following chapters.

2.3.1 Command Messages

Command Messages are transmitted by the EGSE Client to the EGSE Router to instruct to the later to perform a specific task.

2.3.2 Event Messages

Event Messages are passed to the Client whenever the EGSE Router needs to inform the Client about the result of execution of a command, or when data arrive from another Client. Two types of Event Messages exist:

Data Event Messages

Data Event Messages are used to pass the result of a successful command or when data is available from another Client. The formats of the Data Event Message are defined with the description of each Message Type in the following chapters.

Error Event Messages

Error Event Messages – with *Result Code* not Success (0) – are passed when a Command fails execution. A failing command will leave the state of the EGSE Router unchanged (especially no data has been sent if a *SendData* fails). See *Appendix B – Result Codes* for a description of the possible results codes.

3 EGSE ROUTER MESSAGES

In this chapter the EGSE Router Command and Event Messages are defined.

All the exchanged messages are formatted in conformance to the Message Format specified bellow. The descriptions in chapters 3.2 to 3.8 are semantic and only significant message fields are specified. The value of all the other message fields is ignored but **the fields of the Header must always be present whether they are used or not**.

3.1 Message Format

The *EGSE Router Message* contains two separated parts:

- The *Header,* composed of a fixed structure of fields,
- The *Data,* a structure depending on the type of message (*Message Type*) and data (*Data Type*)

The format of the *EGSE Router Messages* is as following:

Part		Field	Size
Header		Message Length	4 octets
		Message Type	1 octets
		Result Code	4 octets
		Destination ID	2 octets
		Source ID	2 octets
		Token	4 octets
		Time	8 octets
	Data Header	Data Type	1 octet
		Spare	1 octet
		Spacecraft ID	2 octets
Data		Message Type specific data	any

Table 2 - EGSE Router Message format

IMPORTANT NOTE: As with most network protocols, all fields are transmitted in **big-endian format** (see chapter 9 *Appendix A – Conventions*).

3.1.1 Fields explanation

Message Length Size in octets of the message, the *Message Length* field not included (= 25 octets + the length of the *Data* part). If an implementation has a maximum message length, it must allow the transfer, using the *Data Type 4,* of the largest PUS packet that has a size of 65 542 octets (see [N1]).

Message Type Specifies the type of message being transferred. The possible type of messages and their associated *Message Type* value are specified in chapter *3.1.2 Message Types*.

Result Code Result Code of a Command Message previously sent. Only used in Error Event Messages to specify a non-successful result (see chapter 3.2); set to 0 (Success) in Command Messages and Data Event Messages. See *Appendix B – Result Codes* for a description of the possible results codes.

Destination ID Identifies the Destination Client to which the message is sent.

Source ID Identifies the Source Client which has sent the message. As multiple Clients can be registered on one connection, it is the responsibility of the host managing the connection to correctly identify the Source Client. It is a protocol violation to send messages with a Source ID of a Client not registered on the current connection.

Token Identifier of the message given by the sending application and can be used, whenever required, to associate Command and Event Messages together. The value of this token is always copied from a Command Message to the generated Event Message. For better traceability, it is recommended that applications always generate a sequential unique number for each Command Message they send, even if they don't need the messages association.

Time Time at which the message has been issued. The format of this field is compliant with the UNIX systems time format. It is made of 2 32-bit unsigned integers encoded as following:

- Bytes 0 to 3 containing an unsigned integer equal to the cumulative number of seconds since the 1970-01-01T00:00:00 UTC epoch (equivalent to a 32-bit *time_t* value).
- Bytes 4 to 7 containing the number of remaining microseconds.

In Client to Client communication, this field is forwarded untouched by the EGSE Router.

The three following fields compose the *Data Header* of the message. They are only of use for *SendData* and *ReceiveData* messages and must be set to 0 otherwise.

Data Type A unique identifier of the type of data that identifies the structure of the *Data* part. See chapter *4 Data Types* for a full description.

Spare Field reserved for future extension. It must be set to 0.

Spacecraft ID Identifies the Spacecraft concerned by this message to the Destination Client.

3.1.2 Message Types

Here are the Message Types defined in the protocol, grouped by validity in Command or Event Messages:

Command Messages (Client to Router)

#	Name	Data
0	Register Client (RegisterClient)	ID, Name
1	Unregister Client (UnregisterClient)	-
2	Send application data to Client (SendData)	Application data
3*	Request Client ID of named Client (RequestClientId)	Client ID
4	Request Client Name with ID (RequestClientName)	Client Name

**Not implemented in current versions of EGSE Router*

Table 3 - Command and Data Messages (Client to Router)

Event Messages (Router to Client)

#	Name	Data
0	Register Client (RegisterClient)	-
1	Unregister Client (UnregisterClient)	-
3*	Request Client ID of named Client (RequestClientId)	Client Name
4	Request Client Name with ID (RequestClientName)	Client ID
5	Receive application data from Client (ReceiveData)	Application data sent by Source Client

**Not implemented in current versions of EGSE Router*

Table 4 - Event Messages (Router to Client)

3.2 Error Event Message

When a Command Message fails processing or execution by the EGSE Router, an Error Event Message is generated. If a Data Event Message was expected, it is replaced by the Error Event Message. The significant fields are:

Field	**Value**
Message Type	Message Type of Command Message
Result Code	Result code indicating the reason of the failure
Destination ID	Client ID of Source of Command Message
Source ID	ESGE Router Client ID (=0xF000)
Token	Value from Command Message
Data	None

Table 5 - Error Event Message format

See *Appendix B – Result Codes* for a description of the possible results codes.

3.3 Register Client (RegisterClient)

Function

Before a Client can communicate with other the Clients (i.e. send *SendData* messages), it must register its Client ID and Client Name with the EGSE Router. The Client ID and Client Name must be unique amongst all registered Clients (note that Client Name is case-sensitive).

Data Event

A Data Event Message with *Result Code* = Success (0) is generated to acknowledge a successful registration of the Client.

3.3.1 Command Message

Field	Value
Message Type	RegisterClient (=0)
Destination ID	ESGE Router Client ID (=0xF000)
Source ID	Client ID of the registering Client
Token	Optional value
Data	Client ID (2 octets) Client Name (null-terminated ASCII string)

Table 6 - Register Client (RegisterClient) Command Message format

3.3.2 Data Event Message

Field	Value
Message Type	RegisterClient (=0)
Destination ID	Client ID of Source of Command Message
Source ID	ESGE Router Client ID (=0xF000)
Token	Value from Command Message
Data	None

Table 7 - Register Client (RegisterClient) Data Event Message format

3.4 Unregister Client (UnregisterClient)

Function

Remove Client with Client ID from the EGSE Router registry.

The Source Client **must** be registered to the EGSE Router (using a *RegisterClient* message) before it can use this message.

Data Event

A Data Event Message with *Result Code* = Success (0) is generated to acknowledge a successful unregistration of the Client.

3.4.1 Command Message

Field	Value
Message Type	UnregisterClient (=1)
Destination ID	ESGE Router Client ID (=0xF000)
Source ID	Client ID of Source of the Message
Token	Optional value
Data	Client ID (2 octets)

Table 8 - Unregister Client (UnregisterClient) Command Message format

3.4.2 Data Event Message

Field	Value
Message Type	UnregisterClient (=1)
Destination ID	Client ID of Source of Command Message
Source ID	ESGE Router Client ID (=0xF000)
Token	Value from Command Message
Data	None

Table 9 - Unregister Client (UnregisterClient) Data Event Message format

3.5 Send application data (SendData)

Function

The *SendData* message is used to request the transfer of data to another Client. It is the only the only message that can be sent to others Clients. The message will be send to the Client specified in the *Destination ID* field. This message will cause a *ReceiveData* event to be generated in the context of the Destination Client. If *Destination ID* is set with the Broadcast Client ID (0xFFFF), all currently connected Clients on the EGSE Router will receive it (including the Source Client).

The Source Client **must** be registered to the EGSE Router (using a *RegisterClient* message) before it can use this message.

Data Event

No Data Event Message is generated if the delivery is successful.

3.5.1 Command Message

Field	Value
Message Type	SendData (=2)
Destination ID	Client ID of Destination of the Message
Source ID	Client ID of Source of the Message
Token	Optional value
Data Type	See below for usage
Data	See below for usage

Table 10 - Send application data (SendData) Command Message format

In contrast to other messages, the *Data* part of the *SendData* (and therefore *ReceiveData*) messages is *Data Type* specific. The corresponding messages data format is given in chapter 4 *Data Types*.

3.5.2 Data Event Message

Not applicable, no Data Event Message is generated on successful delivery.

3.6 Request Client ID of named Client (RequestClientId)

Not implemented in current versions of the EGSE Router

Function

This message type allows a Client to obtain the Client ID of another Client when the Name is known to the other Client. It is only possible to request the Client ID of Clients currently registered. Client Names are case-sensitive.

Data Event

Data contains the Client ID of the requested Client.

3.6.1 Command Message

Field	Value
Message Type	RequestClientId (=3)
Destination ID	ESGE Router Client ID (=0xF000)
Source ID	Client ID of Source of the Message
Token	Optional value
Data	Client Name (null-terminated ASCII string)

Table 11 - Request Client ID of named Client (RequestClientId) Command Message format

3.6.2 Data Event Message

Field	Value
Message Type	RequestClientId (=3)
Destination ID	Client ID of Source of Command Message
Source ID	ESGE Router Client ID (=0xF000)
Token	Value from Command Message
Data	Client ID (2 octets)

Table 12 - Request Client ID of named Client (RequestClientId) Data Event Message format

3.7 Request Client Name with ID (RequestClientName)

Function

This message type allows a Client to obtain the Client Name of another Client when the ID is known to the other Client. It is only possible to request the Client Name of Clients currently registered.

Data Event

Data contains the Client Name of the requested Client.

3.7.1 Command Message

Field	Value
Message Type	RequestClientName (=4)
Destination ID	ESGE Router Client ID (=0xF000)
Source ID	Client ID of Source of the Command Message
Token	Optional value
Data	Client ID (2 octets)

Table 13 - Request Client Name with ID (RequestClientName) Command Message format

3.7.2 Data Event Message

Field	Value
Message Type	RequestClientName (=4)
Destination ID	Client ID of Source of Command Message
Source ID	ESGE Router Client ID (=0xF000)
Token	Value from Command Message
Data	Client Name (null-terminated ASCII string)

Table 14 - Request Client Name with ID (RequestClientName) Data Event Message format

3.8 Data Message (ReceiveData)

Function

This Data Message is sent by the EGSE Router whenever a Client sends a *SendData* Command Message with Destination ID being one of the Clients registered with EGSE Router.

Data Event

Data contains the application data sent by the Source Client. Format of the *Data* is defined by the *Data Type* field. See chapter 4 *Data Types*.

3.8.1 Command Message

Not applicable as *ReceiveData* is a Data Event Message only. To send data, a Client must use the *SendData* Command Message.

3.8.2 Data Event Message

Field	Value
Message Type	ReceiveData (=5)
Destination ID	Client ID of Destination of the Message
Source ID	Client ID of Source of the Message
Token	Value as set in the *SendData* message by the Source Client
Data Type	See *SendData*.
Data	See *SendData*.

Table 15 - Data Message (ReceiveData) Data Event Message format

4 DATA TYPES

Data Types (or also known as *Templates* in original versions of the protocol) are used in conjunction of the *SendData* Command Message and *ReceiveData* Data Event Message to indicate the type of data transferred in the message. The data format of the *Data* part is dependent on the value of the *Data Type* field. Data Types 1 to 8 are defined by the EGSE reference facility of ESA/ESTEC as follow.

#	Name
1	SCOE Command Request
2	SCOE Command Verification Report
3	SCOE Observation Report
4	Send Telecommand Packet Request
5	Send Telecommand Packet Verification Report
6	Telemetry Packet Report
7	Telemetry Frame Report
8	Time Correlation Report

Table 16 - Data Types

The Data Types 0 to 63 are reserved for the ESA reference facility and Data Types 64 to 127 are mission-specific and freely useable.

The format of the *Data* part of the *EGSE Router Message* when used with Data Types 1 to 8 is defined hereafter.

4.1 SCOE Command Request (1)

The *SCOE Command Request* Data Type is used to issue custom commands to the EGSE components (not only SCOEs); e.g. commanding ground stations, configuring the TM/TC Front End, etc. It is up to these components to specify the exact format of the command they support.

The format of the *Data* part is similar to the PUS [N1] request format where:

- The CCSDS and PUS Secondary header have been removed and replaced by specific EGSE headers (as defined in the *EGSE Message Format*).
- The PUS *Application Data* structure remains and is referred as *Parameters* in the following table.
- The *Parameters* structure format depends of the *Service Type* and *Subtype* fields. For those types and subtypes defined within the PUS, the parameter structure shall comply with the one defined within the PUS.

	Field	**Size**	**Value**
Message Data	Service Type	1 octet	Any - See PUS
	Service Subtype	1 octet	Any - See PUS
	Ack	1 octet	Bit 0,1,2,3 (not used) Bit 4 = Completion Ack requirement Bit 5 = Progress Ack requirement Bit 6 = Start Ack requirement Bit 7 = Acceptance Ack requirement
	Parameters	any	Any

Table 17 - Data Type 1: SCOE Command Request format

4.2 SCOE Command Verification Report (2)

To monitor the execution of the SCOE commands, the source application process may have required (using the appropriate command ack directive) the destination application process to report the status of the execution of the command. The command verification reports that the SCOE shall issue, shall comply with the following structure:

The format of the *Data* part is similar to the PUS Service Type 1 [N1] reports format.

	Field	Size	Value
Message Data	Service Type	1 octet	=1
	Service Subtype	1 octet	1 to 8 depending of type of acknowledgement
	SCOE command Token	4 octets	Token value of the related SCOE Command
	Step Number	*1 octet*	This field is only present when the service subtype is 5 or 6.
	Code	*1 octet*	This field may only be present for failure reports i.e. when the service subtype is 2, 4, 6 or 8.
	Parameters	*any*	This field may only be present for failure reports. If used, the code field shall also be used.

Table 18 - Data Type 2: SCOE Command Verification Report format

The *Step Number* field is only present for service subtype 5 and 6. This field indicates the intermediate step number of the command execution profile whose execution has been completed.

The meaning of the different values for the *Service Subtype* field is as following:

Subtype	Meaning
1	Successful Acceptance
2	Failed Acceptance
3	Successful Start
4	Failed Start
5	Successful Progress
6	Failed Progress
7	Successful Completion
8	Failed Completion

Table 19 - SCOE Command Verification Report Service Subtypes

4.3 SCOE Observation Report (3)

Components can send observation reports to any other ground application process.

The format of the *Data* part is similar to the PUS [N1] report format where:

- The CCSDS and PUS Secondary header have been removed and replaced by specific EGSE headers (as defined in the *EGSE Message Format*).
- The PUS *Application Data* structure remains and is referred as *Parameters* in the following table.
- The *Parameters* structure format depends of the *Service Type* and *Subtype* fields. For those types and subtypes defined within the PUS, the parameter structure shall comply with the one defined within the PUS.

	Field	Size	Value
Message Data	Service Type	1 octet	Any - See PUS
	Service Subtype	1 octet	Any - See PUS
	Parameters	any	Any - See PUS

Table 20 - Data Type 3: SCOE Observation Report format

4.4 Send Telecommand Packet Request (4)

Telecommand packets are usually generated by the MCS and sent to the spacecraft via the TM/TC Front End and a Ground Station. The telecommand packet is encapsulated into this Data Type and the command sent to the TM/TC Front End. This command contains the directives to be used by the TM/TC Front End for sending the telecommand packet and the telecommand packet itself.

	Field	Size	Value
Message Data	Reserved	1 octet	=0x0E. Obsolete, kept for compatibility with earlier versions
	Virtual Channel	1 octet	Virtual Channel number (set to 0 if not used)
	MAP ID	1 octet	Multiplexed Access Point Identifier (set to 0 if not used)
	Service Mode	1 octet	AD = 0; BD = 2 (see below)
	Telecommand Packet	any	Full telecommand packet

Table 21 - Data Type 4: Send Telemetry Packet format

4.4.1 Virtual Channels and MAP IDs

The Virtual Channel facility allows one Physical Channel (a stream of bits transferred over a space link in a single direction) to be shared among multiple higher-layer data streams, each of which may have different service requirements. A single Physical Channel may therefore be divided into several separate logical data channels, each known as a Virtual Channel (VC). Each Transfer Frame transferred over a Physical Channel belongs to one of the Virtual Channels of the Physical Channel. The Multiplexed Access Point Identifier (MAP ID) is used to create multiple streams of data within a Virtual Channel. See [N3] for details on this subject.

The Virtual Channel and/or MAP ID functionalities may not be supported by the telecommand packet source or destination and these fields shall be ignored; set them to 0 if this is the case.

4.4.2 Service Modes

When sending telecommand packets in AD mode (acknowledged by spacecraft at frame level), it implies that the FE, GS and ONB command verification reports are to be sent back to the telecommand source (see Data Type *Send Telecommand Packet Verification Report (5)*).

When sending telecommand packets in BD mode (unacknowledged by spacecraft at frame level), it is implied that only the FE and GS command verification reports are to be sent back to the telecommand source (see Data Type *Send Telecommand Packet Verification Report (5)*).

4.5 Send Telecommand Packet Verification Report (5)

Command verification reports generated by the TM/TC Front End in answer to a *Send Telecommand Packet Request* have the following structure (which is similar to the PUS Service Type 1 [N1] report format):

	Field	Size	Value
Message Data	Service Type	1 octet	=1
	Service Subtype	1 octet	9 to 14 (see hereafter)
	EGSE Command Token	4 octets	As set with the corresponding *Send Telecommand Packet Request*
	Code	1 octet	Any
	Parameters	any	Any

Table 22 - Data Type 5: Send Telemetry Packet Verification Report format

4.5.1 Service Subtypes (Verification Types)

Subtype	Meaning
9	Successful TM/TC Front End acceptance (FE ack)
10	Failed TM/TC Front End acceptance (FE nack)
11	Successful Ground Station acceptance (GS ack)
12	Failed Ground Station acceptance (GS nack)
13	Successful Onboard acceptance (ONB ack) (in AD only)
14	Failed Onboard acceptance (ONB nack) (in AD only)

Table 23 - Sending TC Packet Verification Report Service Subtypes (Verification Types)

Verification Types description:

- FE Ack: verification report to be generated by the TM/TC Front End when the command message containing the TC packet is received by the Front End.
- GS Ack: verification report to be generated by the Ground Station when the TC packet has been up-linked to the spacecraft. This message is mandatory when sending TC packet in BD mode.
- ONB Ack: verification report to be generated by the TM/TC Front End when the reception of the TC packet has been received correctly onboard the spacecraft (command issued in AD mode).

4.6 Telemetry Packet Report (6)

This Data Type is used to transfer telemetry packets, usually reconstructed from frames by the TM/TC Front End and sent to the MCS.

Telemetry packets are encapsulated by the telemetry packet source directly as the *Data* without any additional field, as following:

Field	Size	Value
Telemetry Packet	any	Full telemetry packet

Table 24 - Data Type 6: Telemetry Packet Report format

4.7 Telemetry Frame Report (7)

This Data Type is used to transfer telemetry frames, usually received by the Ground Stations and sent to the TM/TC Front End.

Telemetry frames are encapsulated by the telemetry frame source directly as the *Data* without any additional field, as following:

Field	Size	Value
Telemetry Frame	any	Full telemetry frame

Table 25 - Data Type 7: Telemetry Frame Report format

4.7.1 Time Correlation Report (8)

The time correlation reports (TCR) are produced by the Ground Stations and sent to the TM/TC Front End which forwards them unmodified to the MCS. This allows the time tagging of both telemetry frames and packets using the onboard time contained within these TCR. Each time correlation report contains a pair of time values: the absolute ground time and the onboard time for the same event (such as reception/transmission of first or last bit of a frame, see below).

Field	Size	Value
UTC Time	8 octets	Absolute time of event
GS Internal Delays	8 octets	Delays duration in microseconds
Onboard Time Packet	Mission dependent	Complete time packet received from the spacecraft (mission-specific time)

Table 26 - Data Type 8: Time Correlation Report format

UTC Time

This field contains the absolute time of the event encoded in the same format as the *Time* field of the EGSE Message Header (see *3.1.1 Fields explanation*).

GS Internal delays

This field allows the specification of the Ground Station internal delays duration in microseconds. This duration is the treatment delay of the Ground Station to be subtracted to the UTC Time.

Onboard Time

This field contains a simple copy of the onboard time as received from the spacecraft for the event (usually in telemetry transfer frame such as [N2] or [R1]).

The Packet Telemetry Standard [N2] states that:

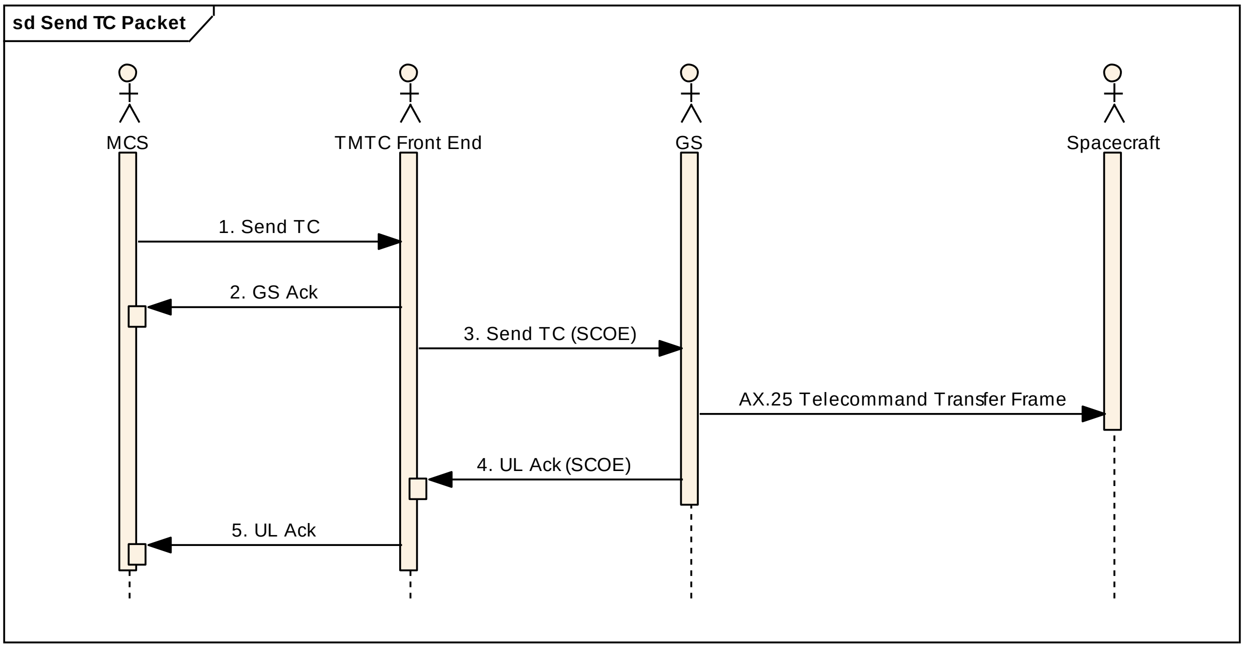
> *Onboard the spacecraft, the contents of the S/C elapsed time clock are sampled at the instant of occurrence of the leading edge of the "first bit of the Attached Synchronisation Marker of that telemetry transfer frame of virtual channel 0" with a VC frame count of "0" (called VC0_F0_Bit0 in the remainder of the note).*
>
> *This time sample is then placed into the standard spacecraft time source packet and telemetered to ground before the Frame Counter of Virtual Channel 0 has counted 255 (configurable) more frames.*

In cases where the first bit of a telemetry transfer frame cannot be used as the time correlation event, such as when using standard HDLC TNCs where only the reception of the full frame is signalled, the chosen event can be the reception/transmission of the last bit of the previous frame.

5 DATA FLOW SCENARIOS

5.1 Send telecommand packet scenario

Figure 5 shows an example of a complete data flow of sending of a telecommand packet with acknowledgments coming from the TM/TC Front End and Ground Station.

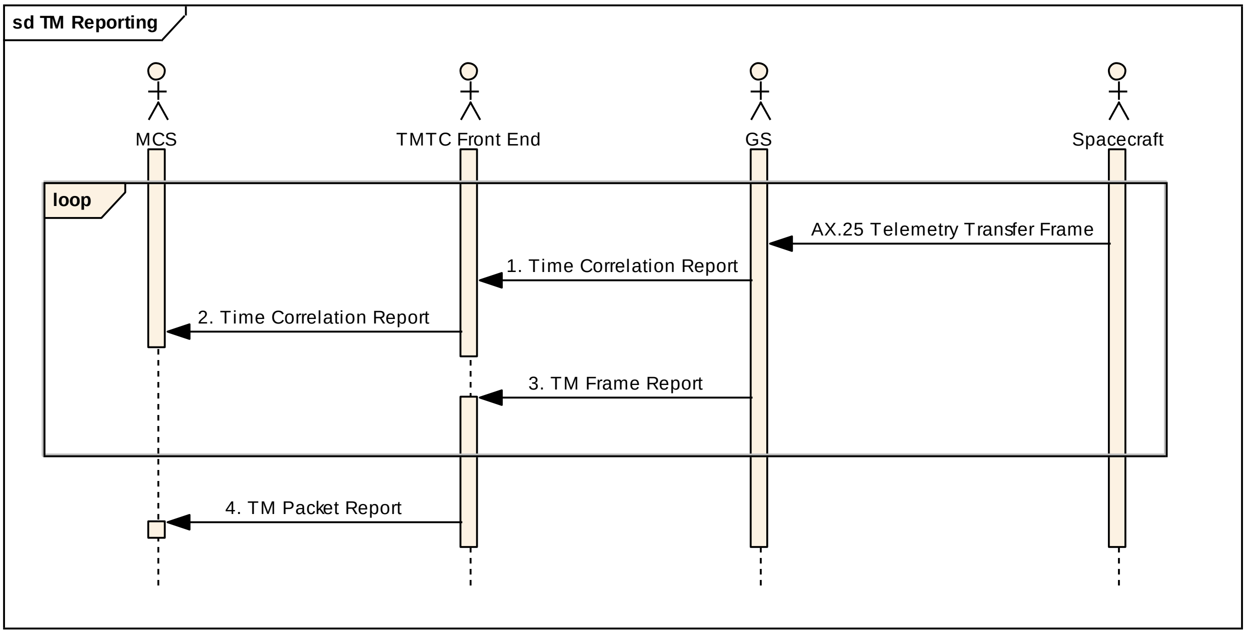


	Message	Source	Destination	Data Type
1	Send TC	MCS	TM/TC Front End	4
2	GS Ack	TM/TC Front End	MCS	5
3	Send TC (SCOE)	TM/TC Front End	Ground Station	1
4	UL Ack (SCOE)	Ground Station	TM/TC Front End	2
5	UL Ack	TM/TC Front End	MCS	5

Figure 5 - Example of data flow when sending a TC packet

5.2 Telemetry reporting scenario

Figure 6 shows an example of a complete data flow when receiving telemetry from the spacecraft. The telemetry frames are received by the Ground Station that transfers them, along with the generated Time Correlation Reports, to the TMTC Front End. The later processes the frames, archiving them and extracting the telemetry packet(s) then sent to the MCS. TCR are archived and forwarded to the MCS.



	Message	Source	Destination	Data Type
1	Time Correlation Report	Ground Station	TM/TC Front End	8
2	Time Correlation Report	TM/TC Front End	MCS	8
3	TM Frame Report	Ground Station	TM/TC Front End	7
4	TM Packet Report	TM/TC Front End	MCS	6

Figure 6 - Example of data flow when receiving telemetry

6 FIGURES AND TABLES

7 INDEXES

7.1 Messages Types

7.2 Data Types

8 REFERENCES

8.1 Normative references

[N1] ECSS-E-70-41A Ground systems and operations - Telemetry and telecommand packet utilization. 30 January 2003.

[N2] ESA PSS-04-106 Packet Telemetry Standard, Issue 1, January 1988.

[N3] ESA PSS-04-107 Packet Telecommand Standard, Issue 2, April 1992.

8.2 Informative references

[R1] S3-BC-SE-1-1-AX.25 Transfer Frames Format

9 APPENDIX A – CONVENTIONS

9.1 Unsigned Integer values format convention

Hexadecimal values are always prefixed by the two characters "0x". Example 0x8000 is equal to the decimal value 32768.

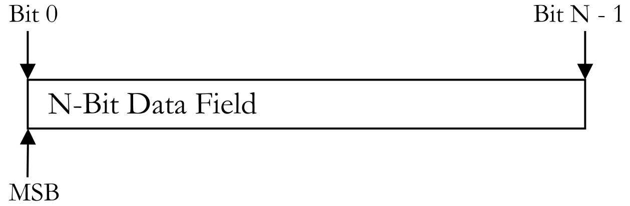
9.2 Bit/Octet Numbering Convention

The following convention is used to identify each bit in a forward-justified N-bit field.

The first bit in the field to be transmitted (i.e. the most left-justified bit when drawing a figure) is defined to be "BIT 0"; the following bit is called "Bit 1" and so on up to "Bit N-1".

When the field is used to express a binary value (such as an integer), the Most Significant Bit (MSB) shall be the first transmitted bit of the field (i.e. "Bit 0"), i.e. in **big-endian** format.

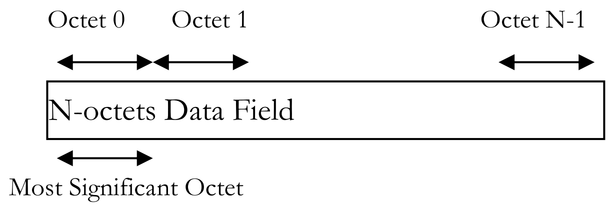
An octet (i.e. a byte) is 8-bits length.



A short word is 16-bits length (i.e. 2 octets).

A word is 32-bits length (i.e. 4 octets).

A long word is 64-bits length (i.e. 8 octets)



The above convention for identifying a bit is also used for identifying each octet in a forward-ordered N-octet field.

10 APPENDIX B – RESULT CODES

#	Name	Description
0	Success	No error. Execution successful.
1*	ChannelInUse*	A Client is already signed on this Channel.
2	UnknownClientName	Client Name unknown or not registered.
3*	NoData*	No data available on the Channels input queue.
4	NotImplemented	Function not implemented.
5	UnknownClientId	Client ID not registered. Usually occurs when sending a message to Client that is not connected.
6	Failure	General failure in server.
7	SignOnDuplicate	Client ID or Name already in use. Cannot register Client.
8	NotSignedOn	Attempt to invoke method prior to registering (other than Request Client ID / Name).
9*	NoSibling*	Attempt to address a Sibling which either is not registered or does not belong to this Parent.
10	ChannelOverflow	Overflow of the client channel. Can occur if the message length is over the limit of the implementation.
11*	TypeConflict*	A Parameter is not of the expected type.
12	MessageFormatError	Message format error.
13	InvalidMessageType	The *Message Type* field has an invalid value.
14	ReceiveDataInCommand	A *ReceiveData* message was sent in a Command Message.
15	InvalidDestination	The destination of the message is invalid. E.g. attempt to send a *RegisterClient* to a Client or a *ReceiveData* to the Router.
16	InvalidClientId	Attempt to register with an invalid or reserved Client ID.

* Result Code present for compatibility with earlier versions and never returned by current version of the EGSE Router