



## **Scream Protocol**

Network protocol specification

### **Specification**

Part No. SWA-RFC-SCRM

Designed and manufactured by  
Güralp Systems Limited  
3 Midas House, Calleva Park  
Aldermaston RG7 8EA  
England

**Proprietary Notice:** The information in this manual is propriety to Güralp Systems Limited and may not be copied or distributed outside the approved recipient's organisation without the approval of Güralp Systems Limited. Güralp System Limited shall not be liable for technical or editorial errors or omissions made herein, nor for incidental or consequential damages resulting from the furnishing, performance or usage of this material.

---

Issue B    March 3, 2009

# Contents

---

<b>Contents</b>	<b>2</b>
<b>1 Introduction</b>	<b>5</b>
<b>2 UDP commands</b>	<b>7</b>
2.1 Command: GCFPING . . . . .	7
2.2 Command: GCFSEND . . . . .	7
2.3 Command: GCFSTOP . . . . .	8
2.4 Response: GCFACKN . . . . .	8
<b>3 TCP commands</b>	<b>9</b>
3.1 Command: 0xFF (request block by 16-bit sequence number) . . . . .	9
3.2 Command: 0xFE (request oldest 16-bit sequence number) . . . . .	9
3.3 Command: 0xFC (request version string) . . . . .	10
3.4 Command: 0xF9 (start TCP block transmission) . . . . .	10
3.5 Command: 0xF8 0xFF (request block by 64-bit sequence number) . . . . .	10
3.6 Command: 0xF8 0xFE (request oldest 64-bit sequence number) . . . . .	11
3.7 Command: 0xF8 0xFD (request terminal passthrough) . . . . .	11
3.8 Command: 0xF8 0xFC (request version string) . . . . .	11
3.9 Command: 0xF8 0xF9 (start TCP block transmission) . . . . .	12
3.10 Command: 0x00–0xEF (request terminal mode) . . . . .	12
<b>4 Block/packet layout</b>	<b>13</b>
4.1 v3.1 . . . . .	13
4.2 v4.0 . . . . .	14
4.3 v4.5 . . . . .	14

---

<b>5</b>	<b>Differences between versions</b>	<b>15</b>
5.1	New in v4.0 . . . . .	15
5.2	New in v4.5 . . . . .	15
<b>6</b>	<b>Revision history</b>	<b>17</b>



# 1 Introduction

---

This document details the Scream protocol, which is designed for the transmission of GCF data over IP links. This protocol is primarily used to transmit data from a CMG-DCM to a PC running the Scream software, although the protocol is suitable for more complex purposes.

In the most basic mode of operation, a Scream client sends a UDP request packet to a Scream server at a regular interval. The Scream server transmits GCF blocks with some additional information to any clients that have sent a recent request. The usual port number (both TCP and UDP) for Scream is 1567, although this is not enforced by any Güralp software.

The Scream protocol has gone through three major revisions, v3.1, v4.0 and v4.5. These are to a large extent backwards compatible.



## 2 UDP commands

---

UDP commands are sent from a Scream client to a Scream server. Commands are sent in a very simple packet structure: the ASCII bytes of the command are sent, followed by any options, and possibly an identifier:

COMMAND: OPTIONS ; IDENTIFIER

Each option is preceded by a colon character. Multiple options may be specified. The identifier is not used by the server, but is instead echoed back in any reply; this can be used to associate a response from the server with a request from the client. Note that early Scream servers may ignore the identifier.

### 2.1 Command: GCFPING

---

A diagnostic command, this causes the server to issue a GCFACKN response, but take no further action.

### 2.2 Command: GCFSEND

---

This command causes the server to start sending data to the client. On receipt of a GCFSEND command, the server issues a GCFACKN response, and adds the client to its list of recipients. Whenever the server acquires a new GCF block, this block is then transmitted to each client in the recipient list.

Note that a Scream client must send exactly one byte order option (:B or :L).

Recommended policy is that a server keeps a client in its list of recipients for five minutes after reception of a GCFSEND command, and that a client sends a GCFSEND command every two minutes.

### **Option: L (little-endian)**

---

This option requests that the Scream server sends the GCF data in little-endian byte order. The server transforms GCF packets and Scream headers so that multi-byte integer fields are in little-endian order. This option is deprecated, and may be ignored by the server. A byte in the Scream header notifies the client of the packet byte order.

### **Option: B (big-endian)**

---

This option requests that the Scream server sends the GCF data in big-endian byte order, which is the recommended default policy. This option is deprecated, and may be ignored by the server. A byte in the Scream header notifies the client of the packet byte order.

## **2.3 Command: GCFSTOP**

---

This command causes the server to remove the client from its recipient list. It causes the server to issue a GCFACKN response.

## **2.4 Response: GCFACKN**

---

This signals that the server has acted upon a client command. It will be followed by any identifier transmitted in the original client command, although some servers may strip this.

## 3 TCP commands

---

The Scream server offers a TCP mode which is used to request missing data (from dropped UDP packets), open a terminal connection, etc. There is also a TCP mode to transfer block data, not using the UDP mechanism. The Scream client initiates a TCP connection to the Scream server, and then issues one or more of the following commands.

### 3.1 Command: 0xFF (request block by 16-bit sequence number)

---

This command requests a missing data block (e.g. from a dropped UDP packet) by its 16-bit sequence ID number. The server response is either to transmit the missing block, in v3.1 or v4.0 format, or to transmit the octet sequence 0xFF 0xFF 0xFF 0xFF, which signifies that the block is no longer available in the backfill buffer.

```
[ uint8] command (0xFF)
[uint16] sequence number, big-endian/network byte order
```

A client may issue other commands after a backfill request.

### 3.2 Command: 0xFE (request oldest 16-bit sequence number)

---

This command requests the oldest block ID held by the server. The server responds with a 16-bit sequence number in network byte order. A client may issue other commands after this request.

### 3.3 Command: 0xFC (request version string)

---

This command requests the version string of the server. The server responds with a string in the following format:

```
[ uint8] length, in octets, of following string
[string] ASCII string, not null terminated
```

The client of the Scream software package expects the server response to start with the identifier GCFSERV followed by the protocol number, e.g. “GCFSERV 4.5”. The server may optionally provide some further identifying version information after this initial section. A client may issue other commands after this request.

### 3.4 Command: 0xF9 (start TCP block transmission)

---

This command requests that the server start transmitting GCF blocks on the TCP link, in v4.0 format. This command is not supported by v3.1 servers. Once issued, the server will not accept further commands on the TCP link. The server will continue to transmit GCF blocks until the client closes the connection.

### 3.5 Command: 0xF8 0xFF (request block by 64-bit sequence number)

---

This is a v4.5 command. This command requests that the server sends the block identified by a 64-bit sequence number. The server response is either to transmit the specified block in v4.5 format or, if it is not available, to transmit the octet sequence 0xFF 0xFF 0xFF 0xFF.

```
[ uint8] command: 0xF8 (v4.5 command prefix)
[ uint8] command: 0xFF (request block)
[uint64] block sequence number, in big-endian/network byte order
```

A client may issue further commands after this request.

---

### 3.6 Command: 0xF8 0xFE (request oldest 64-bit sequence number)

---

This is a v4.5 command. This command requests that the server transmits the 64-bit sequence number of the oldest block its backfill buffer holds. The 64-bit sequence number is transmitted by the server in big-endian/network byte order. A client may issue further commands after this request.

### 3.7 Command: 0xF8 0xFD (request terminal passthrough)

---

This is a v4.5 command. This command requests a connection to the terminal of the data source whose 32-bit routing code is transmitted by the client. Once issued, the TCP link becomes a terminal passthrough (data transmitted by the client is sent to the terminal of the instrument, and data produced by the terminal is transmitted to the client by the server).

```
[ uint8] command: 0xF8 (v4.5 command prefix)
[ uint8] command: 0xFD (request terminal)
[uint32] routing code
```

A client may issue no further commands after this request, as any further data will be transmitted to the instrument terminal. If terminal mode cannot be established, or if the terminal mode times out, then the server will close the TCP link. If the client closes the TCP link, the server is responsible for exiting terminal mode (although the server may choose to rely on the terminal mode timeout to implement this).

### 3.8 Command: 0xF8 0xFC (request version string)

---

Same as command 0xFC, this is only included for symmetry of protocol.

### **3.9 Command: 0xF8 0xF9 (start TCP block transmission)**

---

As command 0xF9, except blocks are transmitted in the v4.5 layout.

### **3.10 Command: 0x00–0xEF (request terminal mode)**

---

In protocol v3.1 and v4.0, data sources were identified by a “COMxx” number in the string description of the source. By interpreting the decimal xx number, and then transmitting this as a single octet to a TCP server, a client would request terminal mode on the data source. This mode has been deprecated in favour of the more flexible v4.5 command 0xF8 0xFD, but it is recommended that servers still support it.

Once established, data from the client is unescaped and transmitted to the terminal. Data from the terminal is escaped and transmitted to the client. Any octet in the range 0xF0–0xFF is escaped by transmitting the two-octet sequence 0xF0 <original octet>.

In terminal mode, the command 0xFA starts an “out of band” command transmission, and 0xFB ends such a transmission. The server stores anything that occurs between 0xFA and 0xFB, and this is transmitted from the server to the instrument when the connection is closed. Note that OOB commands are not cumulative; transmitting a new OOB command will erase any previous stored data.

---

## 4 Block/packet layout

---

Blocks may be transmitted via UDP, in which case each UDP packet contains exactly one block and its associated Scream headers. Blocks may also be transmitted via TCP when requesting blocks by ID or in TCP transmission mode.

All blocks have the same initial 1025 octets: the first 1024 are the GCF block, and the following octet is the packet version number. A very simple client could just take the first 1024 octets and discard the rest.

Multi-byte integers may be in either endianness; this is determined by the endianness code. Recommended default policy is to use big-endian/network byte order.

Each packet is assigned a sequence number, which increments every time there is a new block. The sequence number can be used to request older blocks for backfilling missing data. In v3.1 and v4.0, the sequence number is 16 bits and wraps around to 0 when it reaches 65535. In v4.5, the sequence number is 64 bits and persistent across software/hardware restarts, so that a 64-bit sequence number uniquely identifies every single block ever produced from a given server. The 16-bit sequence number is simply the lowest 16 bits of the 64-bit sequence number.

### 4.1 v3.1

---

Packet size: 1061 octets.

```
[ 1024] GCF block, see SWA-RFC-GCFR
[ uint8] version (31)
[ uint8] source description length
[   32] source description, string in the form
        STRID/COMxx/HOSTNAME, left-justified
[uint16] sequence number
[ uint8] byte order, 1=big-endian/network, 2=little-endian
```

## 4.2 v4.0

---

Packet size: 1077 octets.

```
[ 1024] GCF block, see SWA-RFC-GCFR
[ uint8] version (40)
[ uint8] byte order, 1=big-endian/network, 2=little-endian
[uint16] sequence number
[ uint8] source description length
[   48] source description, string in the form
        STRID/COMxx/HOSTNAME, left-justified
```

## 4.3 v4.5

---

A v4.5 packet has the same layout for the initial 1077 octets as a v4.0 packet, but an additional 12 octets are added to its end, giving a packet size of 1089 octets.

```
[ 1024] GCF block, see SWA-RFC-GCFR
[ uint8] version (45)
[ uint8] byte order, 1=big-endian/network, 2=little-endian
[uint16] low 16 bits of sequence number
[ uint8] source description length
[   48] source description, free-form string, left-justified
[uint32] terminal routing code
[uint64] full 64-bit sequence number
```

## **5 Differences between versions**

### **5.1 New in v4.0**

The packet layout changed to introduce a longer source description, required as networks become more complex. A real-time TCP transmission mode for GCF blocks (command 0xF9) was introduced.

### **5.2 New in v4.5**

The packet layout was expanded to add a terminal routing code and a 64-bit sequence number. Older clients can simply truncate packets to the 1077 bytes of v4.0 and continue to work as before.

A set of TCP commands prefixed with 0xF8 were added. These use the new 64-bit sequence number and 32-bit terminal routing code. Servers are encouraged to implement the older TCP commands in addition to the new commands, but a v4.5-only server need only implement the 0xF8-prefixed commands.

The source description string is no longer used by v4.5 clients to get a terminal; instead, a unique (per server) routing code is used. Pre-v4.5 clients will still use the old terminal method. The new terminal mode does not have “out of band” command transmission (closing terminal is now handled by the client), or byte escaping.

The 64-bit sequence number is unique per server. It starts at 0 and every block the server sees is assigned a new sequence number, with the sequence incrementing by 1 for each block. This should be persistent over software and hardware reboots. It allows arbitrary backfill, as long as the server has enough storage. Older clients use the lowest 16 bits of the full sequence number, which means they can only address the most recent range of packets.



## 6 Revision history

---

Issue A 2008-05-31. Initial release.

Issue B 2009-03-03. Update formatting and add note that the GCFSEND UDP command requires an explicit byte-order option.