Network Working Group
Request for Comments: 3423
Category: Informational

K. Zhang
E. Elkin
XACCT Technologies

November 2002

XACCT's Common Reliable Accounting for Network Element (CRANE) Protocol Specification Version 1.0

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document defines the Common Reliable Accounting for Network Element (CRANE) protocol that enables efficient and reliable delivery of any data, mainly accounting data from Network Elements to any systems, such as mediation systems and Business Support Systems (BSS)/ Operations Support Systems (OSS). The protocol is developed to address the critical needs for exporting high volume of accounting data from NE's with efficient use of network, storage, and processing resources.

This document specifies the architecture of the protocol and the message format, which MUST be supported by all CRANE protocol implementations.

Table of Contents

1 Introduction

Network Elements are often required to export usage information to mediation and business support systems (BSS) to facilitate accounting. Though there are several existing mechanisms for usage information export, they are becoming inadequate to support the evolving business requirements from service providers.

For example, some of the export mechanisms are legacies of the Telco world. Typically usage information is stored in Network Elements as Log files (e.g., CDR files), and exported to external systems in batches. These are reliable methods, however, they do not meet the real-time and high-performance requirements of today's rapidly evolving data networks.

RADIUS [1] is a widely deployed protocol that may be used for exporting usage information. However, it can only handle a few outstanding requests and is not extensible due to its limited command

and attribute address space. RADIUS also does not support unsolicited messages from a server to a client. A detailed analysis of limitations of RADIUS can be found in [3].

DIAMETER [2] is a new AAA protocol that retains the basic RADIUS model, and eliminates several drawbacks in RADIUS. The current DIAMETER protocol and its extensions focus on Internet and wireless network access, and their support to accounting is closely associated with authentication/authorization events. DIAMETER is intended to solve many problems in the AAA area; by doing so, it does not adequately address some critical issues such as efficiency and performance in an accounting protocol.

There are also SNMP based mechanisms that generally require a large amount of processing and bandwidth resources.

Based on the above analysis, a critical need for a reliable, fast, efficient and flexible accounting protocol exists. The XACCT's CRANE protocol is designed to address these critical requirements.

This document defines the CRANE protocol that enables efficient and reliable delivery of any data, mainly accounting data from Network Elements to any systems, such as mediation systems and BSS/OSS. The protocol is developed to address the critical needs for exporting high volume of accounting data from NE's with efficient use of network, storage, and processing resources.

This document specifies the architecture of the protocol and the message format, which MUST be supported by all CRANE protocol implementations.

1.1 Specification of Requirements

In this document, the keywords "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", and "MAY" are to be interpreted as described in BCP 14 [5]. These keywords are not case sensitive in this document.

1.2 Terminology

CRANE Protocol

CRANE stands for Common Reliable Accounting for Network Element. The CRANE Protocol maybe referred as CRANE, or the Protocol in this document. The CRANE Protocol is used at the interface(s) between a CRANE client and one or multiple CRANE servers for the purpose of delivering accounting data.

Client or CRANE Client

A CRANE Client is an implementation on the data producing side of the CRANE protocol. It is typically integrated with the network element's software, enabling it to collect and send out accounting data to a mediation/billing system using the protocol defined herein.

Server or CRANE Server

A CRANE Server is an implementation on the data receiving side of the CRANE protocol. It is typically part of a Business Support System (BSS) (e.g., Billing, Market Analysis, Fraud detection, etc.), or a mediation system. There could be more than one CRANE server connected to one CRANE client to improve robustness of the usage information export system.

CRANE Session

A CRANE Session is a logical connection between a CRANE client and one or multiple CRANE servers for the purpose of delivering accounting data. Multiple sessions MAY be maintained concurrently in a CRANE client or a CRANE server; they are distinguished by Session IDs.

Server Priority

A CRANE server is assigned with a Priority value. Accounting data is always delivered to the perceived operating CRANE server (from the CRANE client point of view) with the highest Priority value (the primary server) within a CRANE Session.

Message

A Message is encoded according to rules specified by the CRANE protocol and transmitted across the interface between a CRANE client and a CRANE server. It contains a common CRANE header and optionally control or user data payload.

Data Record

A Data Record is a collection of information gathered by the Network Element for various purposes, e.g., accounting. The structure of a Data Record is defined by a Template.

Template

A Template defines the structure of any types of Data Record, and specifies the data type, meaning, and location of the fields in the record.

Data Sequence Number (DSN)

An accounting Data Record level sequence number, which is attached to all data messages to facilitate reliable and in-sequence delivery.

2 Protocol Overview

The CRANE protocol is designed to deliver accounting data reliably, efficiently, and quickly. Due to the nature of accounting data, large records often need to be transmitted; thus supporting fragmentation of large records is required. Furthermore, the value associated with accounting data is high; to prevent data loss, quick detection of unresponsive CRANE servers is also required for added robustness.

The CRANE protocol can be viewed as an application that uses the data transport service provided by lower layer protocols. It relies on a transport layer protocol to deliver reliable, in-sequence data packets.

UDP is a simple connectionless transport layer protocol that has advantages of being fast and agile, but it provides no reliability and lacks flow control mechanisms. Hence, The CRANE protocol must not use UDP as the transport layer protocol to avoid the risk of adversely impacting the networks it is being run over.

TCP and SCTP [4] are two transport layer protocols that fulfill the reliability requirement of CRANE. Either one of them MAY be used to transport CRANE messages. TCP meets some of the requirements, but not all (e.g., quick detection of server failure, the fact that TCP is stream oriented and not record oriented). Therefore, SCTP [4] is the preferred way to transmit CRANE messages.

2.1 CRANE Architecture

The CRANE protocol is an application running over a reliable transport layer protocol. The transport layer protocol is responsible for delivering CRANE messages between CRANE clients and CRANE servers. It MUST support the following capabilities:

1. Reliable, in-sequence message delivery.
2. Connection oriented.
3. Delivery of messages with a length of up to 2^32 octets (i.e., the transport layer has to support fragmentation of messages when running over IP).

The transport layer MAY support:

1. Authentication.
2. Bundling of multiple messages into a single datagram.

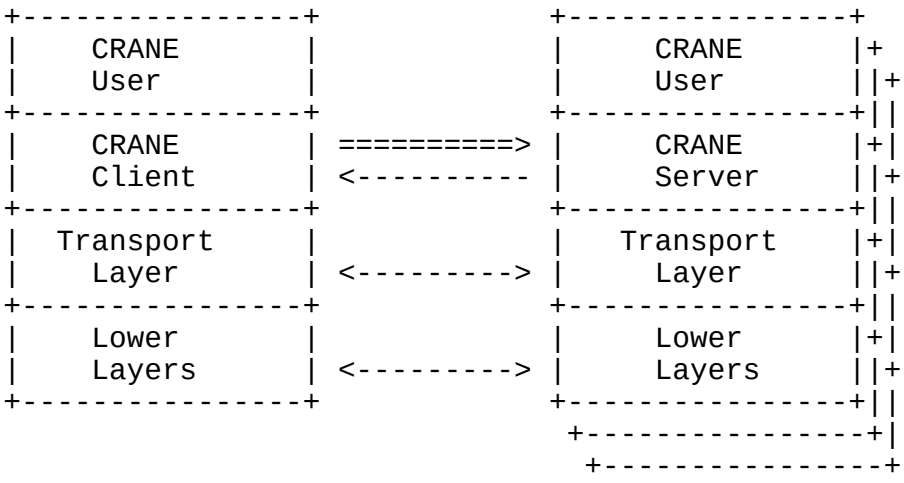
Possible transport layer protocols MAY be TCP or SCTP [4]. TCP supports the minimal requirements for CRANE, but lacks some desirable capabilities that are available in SCTP, these include:

1. Session level authentication.
2. Message based data delivery (as opposed to stream based).
3. Fast connection failure detection.

Reliable delivery of accounting data is achieved through both the transport layer level and the CRANE protocol level. The transport layer acknowledgments are used to ensure quick detection of lost data packets and unresponsive servers, while the CRANE protocol acknowledges CRANE messages after they have been processed and the accounting information has been placed in persistent storage.

Being a reliable protocol for delivering accounting data, traffic flowing from a CRANE client to a CRANE server is mostly accounting data. There are also bi-directional control message exchanges, though they only comprise of small portion of the traffic.

The following diagram illustrates the CRANE protocol architecture:



2.2 CRANE over TCP

TCP can be used as a transport layer for the CRANE protocol. CRANE running over TCP MUST conform to the following rules:

1. The CRANE client MUST accept TCP connections over a specific TCP port.
2. The CRANE server MUST connect to the CRANE client, and SHOULD be responsible for reestablishing a connection in case of a failure.
3. CRANE messages are written as a stream of bytes into a TCP connection, the size of a CRANE message is specified by the Message Length field in the CRANE message header.

2.3 Alternate servers

For purposes of improved reliability and robustness, redundant CRANE server configuration MAY be employed. The CRANE protocol supports delivering accounting data to alternate CRANE servers, which may be part of a mediation system or a BSS.

A CRANE session may comprise of one or more CRANE servers. The CRANE client is responsible for configuring network addresses of all CRANE servers belonging to the session. A Server Priority is assigned to each CRANE server. The Server Priority reflects the CRANE client's preference regarding which CRANE server should receive accounting data. The assignment of the Server Priority should consider factors such as geographical distance, communication cost, and CRANE server loading, etc. It is also possible for several CRANE servers to have the same priority. In this case, the CRANE client could randomly choose one of them as the primary server to deliver accounting data.

Additional features such as load balancing may be implemented in a multi-server environment. The process of configuring CRANE client is carried out using the NE's configuration system and is outside the scope of this document.

A CRANE client MUST deliver accounting data to its perceived operating CRANE server with the highest priority; if this CRANE server is deemed unreachable, the CRANE client MUST deliver the accounting data to the next highest priority CRANE server that is perceived to be operating. If no perceived operating CRANE servers are available, accounting data MUST be queued in the CRANE client until any CRANE server is available or the client's queue space runs out. An alarm should be generated to inform the CRANE user of the queue overflow condition.

Accounting data delivery SHOULD revert to the higher priority server when it is perceived to be operating again.

The CRANE protocol does not specify how a CRANE client should redirect accounting data to other CRANE servers, which is considered an implementation issue. But all the supporting mechanisms are provided by the protocol to work in a multiple-server environment (e.g., the template negotiation process, and configuration procedures, etc.). The transport layer (together with some other means) is responsible for monitoring server's responsiveness and notifying CRANE protocol for any failures. The client may choose to transition to an alternate server.

Implementation Note:

The transition to an alternate CRANE server is an implementation issue and should occur under the following conditions:

A) Transport layer notifies the CRANE client that the corresponding port of the CRANE server is unresponsive.

B) Total size of unacknowledged accounting records has exceeded a threshold (configurable) for certain duration (configurable).

C) A STOP message is received from the active server.

D) A lower priority server is the active one and a higher priority server has recovered.

2.4 Templates

The CRANE protocol enables efficient delivery of accounting data. This is achieved by negotiating a set of Data Templates for a CRANE session before actual accounting data is delivered. A data template defines the structure of a DATA message payload by describing the data type, meaning, and location of the fields in the payload. By agreeing on session templates, CRANE servers understand how to process DATA messages received from a CRANE client. As a result, a CRANE client only needs to deliver actual accounting data without attaching any descriptors of the data; this reduces the amount of bytes sent over communication links.

A template is an ordered list of keys. A key is the specification of a field in the template. It specifies an accounting item that a network element MAY collect and export. The specification MUST consist of the description and the data type of the accounting item. (e.g., 'Number of Sent Bytes' can be a key that is an unsigned integer of 32 bit long). A CRANE client typically defines keys.

The CRANE protocol supports usage of several templates concurrently (for different accounting records). Keys contained in a template could be enabled or disabled. An enabled key implies that the outgoing data record will contain the data item specified by the key. A disabled key implies that the outgoing record will omit the specified data item. The enabling/disabling mechanism further reduces bandwidth requirement; it could also reduce processing in network elements, as only needed data items are produced.

In a CRANE session, all the CRANE servers and the CRANE client MUST use the same set of templates and associated enable/disable status. The templates' configuration and connectivity to an end application MUST be the same in all servers. The CRANE client MUST publish the relevant templates to all CRANE servers in a session through user configuration, before it starts to send data according to the templates.

The complete set of templates residing in a CRANE client MUST bear a configuration ID that identifies the template set. Each data record is delivered with the Template ID and the Configuration ID, so that the correct template can be referenced. A server, when receiving a record with an older Configuration ID, MAY handle the record gracefully by keeping some template history. The transport layer should ensure that a server would not get messages with future configuration IDs.

2.5 Template Transmission and Negotiation

As stated before, all CRANE servers MUST use the same set of templates in a CRANE session. In case that servers do not share the same set of templates (the templates are considered different if different keys are enabled or disabled), a negotiation process between the client and the server would ultimately determine one set of templates that is accepted and used by all the CRANE servers in a session.

After a CRANE session is established and the server sent a START message indicating that it is ready to take part in the session, the client MUST deliver the set of templates that it intends to use by sending a TMPL DATA message to the server. The CRANE server MUST acknowledge the reception of the set of templates.

Templates are negotiable between a CRANE client and CRANE servers. A CRANE server may propose changes to the templates received from a CRANE client (e.g., enabling some keys and disabling others), or it can acknowledge the templates as is. In the case that a template or a key is not recognized by the server (e.g., they might be added to the client after the server configuration has completed), the server MAY choose to disable each unknown key or unknown templates in order to avoid unnecessary traffic. A template is disabled when all the keys are disabled. If changes were received from the CRANE servers, the client will send the changed template set to all connected servers (using FINAL_TMPL_DATA message). It is the client's responsibility to decide what would be the final set of templates used by a session. At this time, each CRANE server MUST accept and acknowledge the templates without changing anything (to avoid deadlock and loop conditions). Each CRANE server is given a single chance to propose any changes during the negotiation process.

The template negotiation process is outlined as follows:

A) CRANE client sends a TMPL DATA message with a set of templates.

B) CRANE server either responds with the TMPL DATA ACK message with changes in the template set (process continues in step C) or responds with FINAL TMPL DATA ACK message if no changes are needed (process continues in step E).

C) CRANE client receives proposed changes, incorporates them if possible and then sends a FINAL TMPL DATA message containing the new set of templates to all servers (in order to deploy the change).

D) CRANE server receives the FINAL TMPL DATA message containing the new set of templates and MUST send a FINAL TMPL DATA ACK message to

acknowledge the reception of the templates. No changes are allowed at this stage and the templates, which the client sent, are going to be used.

E) CRANE client receives a FINAL TMPL DATA ACK message from the server and can assume that the server knows which templates to use.

All these stages take place only when there are multiple CRANE servers with differences in the template set (e.g., not all key states are identical). If all CRANE servers within a session share the same configuration exactly, all servers will respond with FINAL TMPL DATA ACK and the ping-pong between the client and the servers will end immediately. This is the common case, but in case some other CRANE servers have a different configuration, the protocol offers the way to maintain consistency among CRANE servers.

Implementation Note:

> TMPL DATA messages SHOULD be sent only after all DATA messages with the previous configuration have been acknowledged. This ensures the server to transition properly to the new configuration.

2.6 Changing Templates

Though TMPL DATA messages allow for deploying and publicizing template, a need to configure the template set still exists. Each of the CRANE servers in a CRANE session may change the template set, which is typically requested by an end-user through User Interface. If the end-users need to know what templates are available and the current template set status, they may issue the GET TMPL message.

The following steps are performed in order to change the templates:

A) The server MUST retrieve from CRANE client the template set that requires change by issuing GET TMPL message. The server can issue a GET TMPL even if it has not yet issued a START message.

B) After received a GET TMPL message, the client sends back a GET TMPL RSP message with the requested data.

C) The server makes the necessary changes to the templates and sends back a START NEGOTIATION message. This message triggers the CRANE client to inquire about changes made by the CRANE server.

D) After received a START NEGOTIATE message, the client MUST respond with START NEGOTIATE ACK message followed by a TMPL DATA message. From this point on, the template negotiation process starts.

2.7 Flow Control

After templates have been deployed, DATA messages start to arrive at the primary CRANE server (the operational one with the highest priority within the CRANE session). Each DATA message contains a Data Sequence Number (DSN). The primary CRANE server MUST accept the data as long as it is in-sequence. Out-of-sequence DATA messages should be discarded.

The CRANE server detects the start of accounting data when it receives the first DATA message either after startup or after a server transition. The first DATA message MUST have the 'S' bit ('DSN Synchronize' bit) set by the CRANE client. Upon reception of the message with initial DSN, the server MUST accept all in-sequence DATA messages. The DSN MUST be incremented by 1 for each new DATA message originated from the client.

A CRANE server MUST acknowledge the reception and correct processing of DATA messages by sending DATA ACK messages. The DATA ACK MUST contain the DSN of the last processed in-sequence DATA message. If the CRANE server receives an Out Of Sequence DATA message, it MUST also send a DATA ACK message. It will trigger an immediate retransmission of unacknowledged records.

The CRANE client is responsible for delivering all the records. In the case of a redundant server configuration, there could be a scenario when one server does not receive all the records but another redundant CRANE server for the same mediation system receives the rest of the records. For example, server #1 could receive records 3042-3095 and then 3123-..., with server #2 receiving records 3096-3122. It is the sender's responsibility to deliver all the records, in-sequence, but not necessarily to the same server.

The billing/mediation system eventually receives all the records, possibly through more than one CRANE server. The CRANE client MUST convey all the records it received to the billing/mediation system. This MAY result in duplicate records in the billing/mediation system. In this case, the DSN MUST be used to remove duplicates. To aid the process of duplicate removal, whenever a record is re-sent to another server, its 'Duplicate' bit MUST be set to suggest that this record might be a duplicate.

Implementation Note:

> When the amount of unacknowledged records reaches a threshold, a timer should be started. When the timer expires, all the unacknowledged records should be transmitted to an alternate

server with 'D' bit set in the DATA message; if alternate servers are not available, the records should be retransmitted.

The CRANE flow control also supports redundant server configuration. A server MUST send a START message in order to move to the 'ready' state. In the 'ready' state, the server can receive and process CRANE messages. To leave the 'ready' state and stop the message flows from the client, the server should send a STOP message to the client.

2.8 The CRANE Client Query Messages

A CRANE server may query a CRANE client's status by sending query messages after it has established a session with the client. A CRANE client that is connected to the server MUST respond with response messages. All the Query Messages MUST be initiated by a CRANE server. The CRANE protocol defines three such Query Message pairs, they are:

Get Session (GET SESS)
Get Session Response (GET SESS RSP)

Get Template (GET TMPL)
Get Template Response (GET TMPL RSP)

Status Request (STATUS REQ)
Status Response (STATUS RSP)

All the query messages incorporate a Request ID field for tagging purposes and matching requests and responses. This field contains a 16 bit counter incremented with every request and is set by the initiator of the request. Along with the CRANE server's IP address and port number, this constitutes a unique identifier for a request. This value MUST be copied to Request ID field in the response message in order to associate a specific response with a request.

The CRANE client SHOULD collect and send out meta-data about the data collected (counters, statistics, etc.). This is done by creating status templates, which are treated like any other template, with the exception that these templates are marked with a /'Status' bit. Status templates are used with the set of STATUS REQ and STATUS RSP messages. A server MAY issue a STATUS REQ to a CRANE client and receive a STATUS RSP message with the requested data.

2.9 CRANE Sessions

A CRANE client MAY deliver accounting data to different mediation/billing systems by establishing different CRANE sessions.

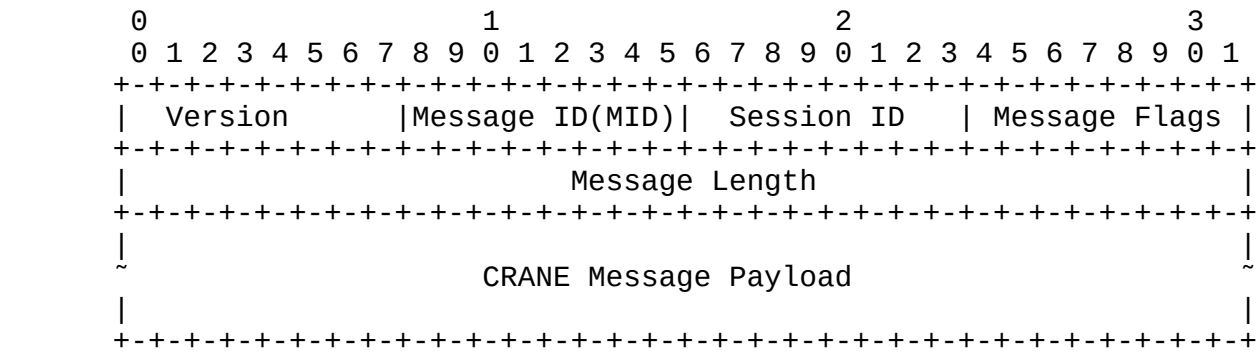
Each session MAY consist of several CRANE servers in a redundant configuration. The session ID imbedded in all the CRANE messages enables the correct association of CRANE sessions with CRANE users. All the CRANE processes (e.g., template negotiation, configuration, flow control, etc.) should be carried out in the same way in a multi session scenario.

Each session has its set of templates (these may be the same templates, but the keys could be enabled or disabled differently). The sessions are configured in the NE, each with a different session name with associated Session IDs. The session ID is carried in each message to associate the message with a specific session.

A CRANE server MAY take part in different sessions. When configuring a server, it needs to know the sessions in which it participates. The server can issue a GET SESS message to receive a list of relevant sessions.

3 CRANE Message Format

A summary of the CRANE protocol message format is shown below. A CRANE message consists of an 8 octet message header; it is followed by a variable length message payload that is aligned to 32 bit boundary. Some of the messages do not have the CRANE Message Payload part. The fields are in network byte order and transmitted from left to right.



Version: 8 bit unsigned integer

The Version field indicates the supported CRANE protocol implementation. This field MUST be set to 1 to indicate the CRANE protocol Version 1.0. CRANE protocol Version 1.0 only supports Ipv4 addressing; however, it can be used to transfer information related to Ipv6 flows.

Message ID (MID): 8 bit unsigned integer

The Message ID field identifies the type of the message. The message IDs defined by CRANE Version 1 are:

Message Name	Short Name	Message ID
Reserved		0x00
Flow Start	START	0x01
Flow Start Acknowledge	START ACK	0x02
Flow Stop	STOP	0x03
Flow Stop Acknowledge	STOP ACK	0x04
Connect	CONNECT	0x05
Template Data	TMPL DATA	0x10
Template Data Acknowledge	TMPL DATA ACK	0x11
Final Template Data	FINAL TMPL DATA	0x12
Final Template Data Ack.	FINAL TMPL DATA ACK	0x13
Get Sessions	GET SESS	0x14
Get Sessions Response	GET SESS RSP	0x15
Get Template	GET TMPL	0x16
Get Template Response	GET TMPL RSP	0x17
Start Negotiation	START NEGOTIATE	0x18
Start Negotiation Ack.	START NEGOTIATE ACK	0x19
Data	DATA	0x20
Data Acknowledge	DATA ACK	0x21
Error	ERROR	0x23
Status Request	STATUS REQ	0x30
Status Response	STATUS RSP	0x31

Session ID: 8 bit unsigned char

The Session ID field identifies the session with which the message is associated. The session ID is ignored in the case of GET SESS and GET SESS RSP messages. More details about session can be found in Section 2.9.

Message Flags: 8 bit unsigned char

The Message Flags field can be used to identify options associated with the message. For CRANE Version 1.0, all the flags are reserved; unless otherwise specified, the flags are set to zero on transmit and are ignored on receipt.

Message Length: 32 bit unsigned integer

The Message Length field is the total length of the CRANE message in octet including the header.

4 CRANE Messages

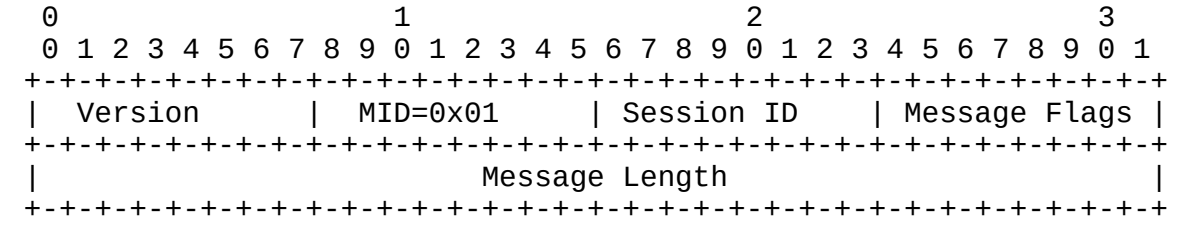
This section defines CRANE mandatory messages. They MUST be supported by any CRANE protocol implementation.

4.1 Flow Start (START)

Description

The Flow Start message is sent from a CRANE server to a CRANE client to indicate that the CRANE server is ready to receive CRANE messages.

Message Format

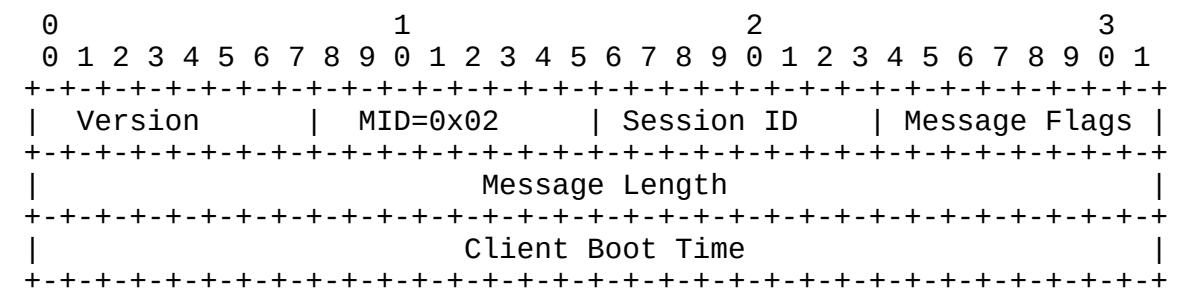


4.2 Flow Start Acknowledge (START ACK)

Description

The Flow Start Acknowledge message is sent by a CRANE client to acknowledge the reception of a START message from a specific CRANE server. It is sent only to that server to indicate that the client considers it ready to receive CRANE messages.

Message Format



Client Boot Time: 32 bit unsigned integer

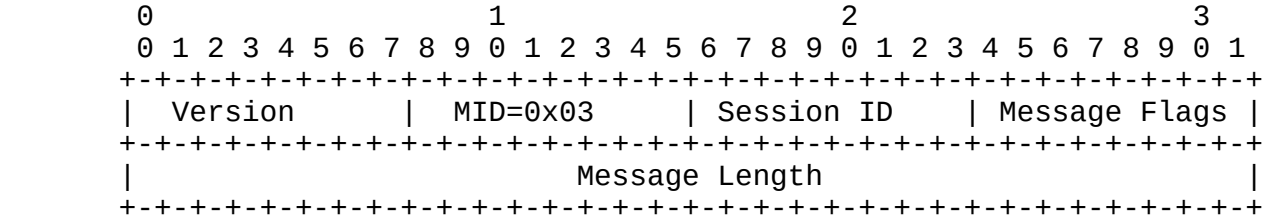
The Client Boot Time field is the timestamp of the last client startup in seconds from 1970. This field can be combined with the DSN and the client's IP address to serve as a system wide unique record identifier.

4.3 Flow Stop (STOP)

Description

The Flow Stop message is sent from a CRANE server to a CRANE client to instruct it to stop sending data (to that server). The STOP message does not disconnect the server; it only stops the CRANE client from sending "DATA" messages.

Message Format

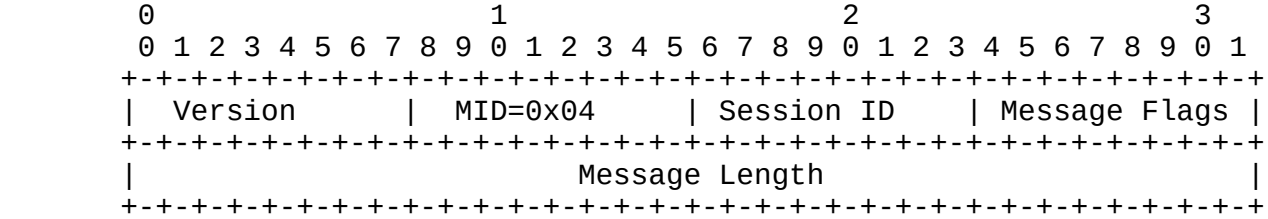


4.4 Flow Stop Acknowledge (STOP ACK)

Description

The Flow Stop Acknowledgement message acknowledges the STOP message issued by a CRANE server.

Message Format

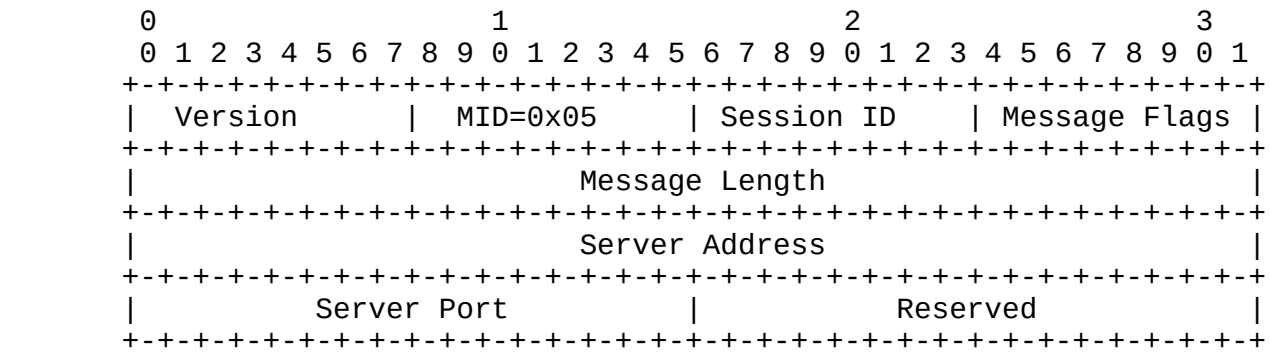


4.5 Connect (CONNECT)

Description

The CONNECT message is sent from a CRANE server to a CRANE client to identify itself. The message MUST be the first message sent over a transport layer connection between the server and the client.

Message Format



Server Address: 32 bit unsigned integer

The Server Address field is the server's IP address (IPV4).

Server Port: 16 bit unsigned integer

The Server Port field is the server's port number for the transport layer (the port number specified here doesn't necessarily have to match the port number used by the transport layer)

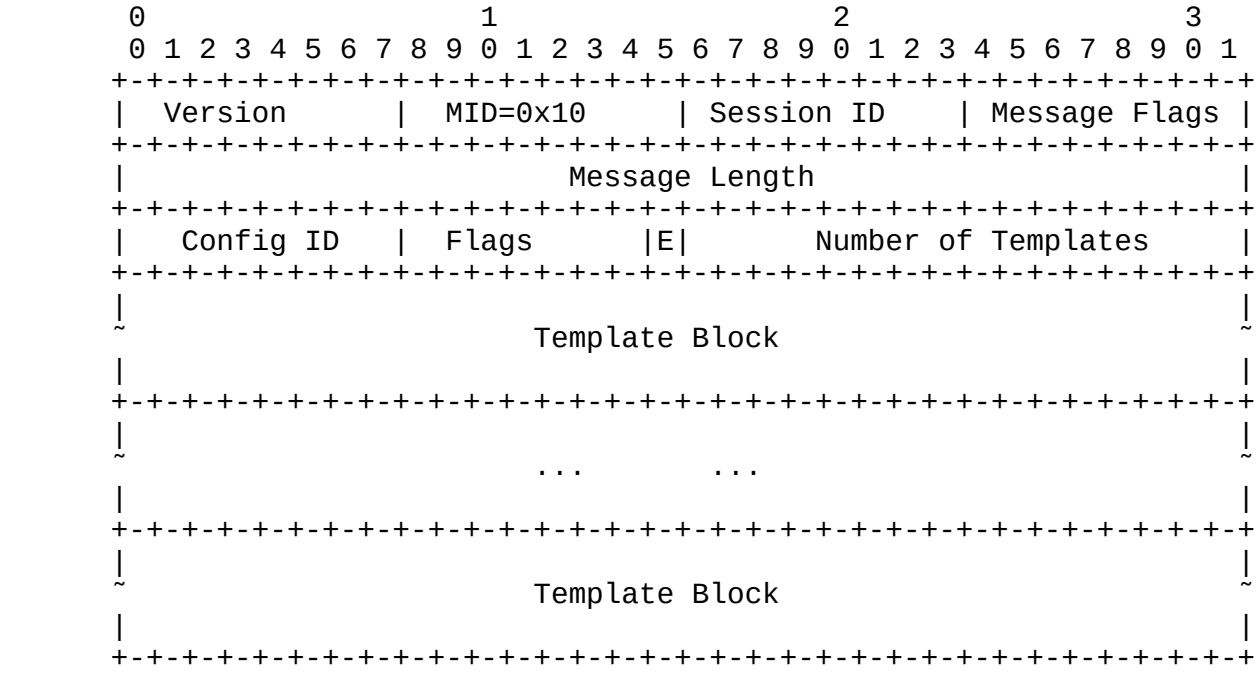
4.6 Template Data (TMPL DATA)

Description

A CRANE client sends the Template Data message to a CRANE server after a START or a START NEGOTIATE message was received from the server. The message MUST contain all the templates that are going to be used for the session. It SHOULD also include the template for the status records (See section 2.8)

The receiving CRANE server MUST acknowledge the message by sending either a TMPL DATA ACK (if template changes are needed) or a FINAL TMPL DATA ACK message. For more information, see section 2.5.

Message Format



Configuration ID (Config. ID): 8 bit unsigned char

The Configuration ID field identifies the version number associated with a template set. Changes to any of the templates would result in a new template version, and the version number would be incremented by one. An implementation SHOULD handle rollovers of the version number.

Flags: 8 bit unsigned char

The Flags field identifies any options associated to the message.

The flag defined by the CRANE Version 1 is:

The 'E' bit indicates the transmission order of the "DATA" messages. If the field is set to 1, data is in big endian format; otherwise, little endian format is used.

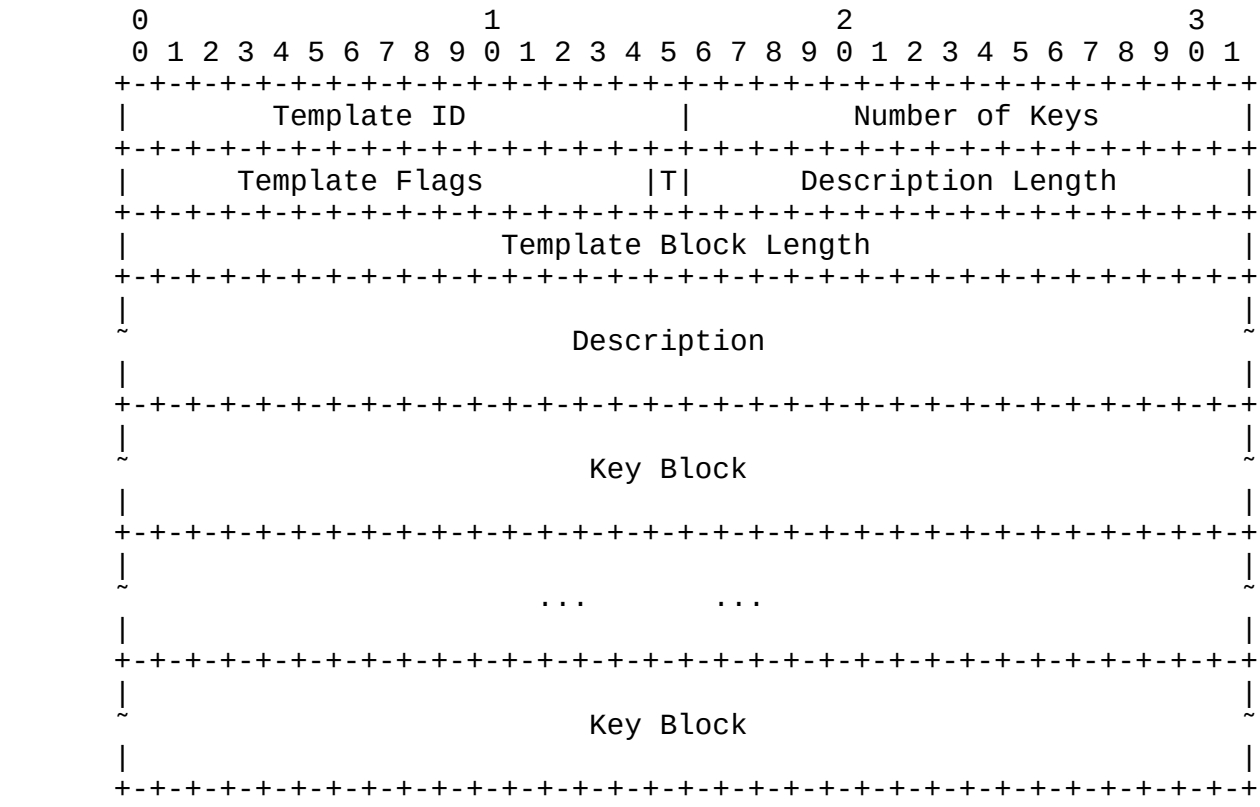
Number of Templates: 16 bit unsigned integer

The Number of Templates field is the number of Templates (a template is described by a Template Block) specified by the message.

Template Block

The Template Block field is of variable length and aligned to 32 bit boundary. It is the specification of a template.

Template Block Format:



Template ID: 16 bit unsigned integer

The Template ID field identifies a specific template.

Number of Keys: 16 bit unsigned integer

The Number of Keys field is the number of keys included in the template.

Template Flags: 16 bit unsigned integer

The Template Flags field is composed of flags that indicate different attributes of the template. In CRANE Version 1.0, only the 'T' bit is defined, other bits in the field SHOULD be set to zero by the sender and ignored by the receiver.

The 'T' bit ('Status' bit) indicates that the template is a status template that is used by the STATUS RSP message only. See section 2.8 for more details.

Description Length: 16 bit unsigned integer

The Description Length field is the length of the Description field. If no description is supplied, the length MUST be 0.

Template Block Length: 32 bit unsigned integer

The Template Block Length is the length of the template block in octets.

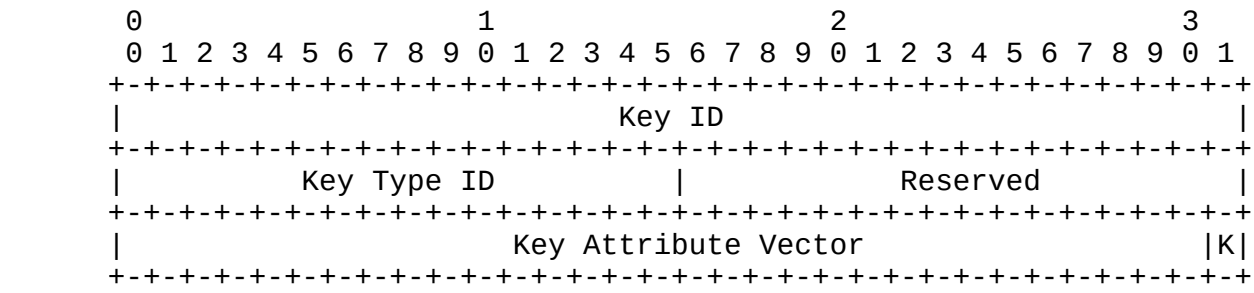
Description: Variable length unsigned char

The Description field contains the text description of the template (e.g., "Aggregated by interface and ToS bits"). It is a variable length field of up to 64Kb long, and padded with 0 to the next 32 bit boundary.

Key Block

A key Block contains the specification of a key within a template.

Key Block Format



Key ID: 32 bit unsigned integer

The Key ID field identifies the key within a template. See section 2.4 for more details.

Key Type ID: 16 bit unsigned integer

The Key Type ID field specifies the data type of the key.

The fixed length data types are defined as following:

Data Type	Data Type ID
Boolean (1)	0x0001
Unsigned Integer8	0x0002
Signed Integer8	0x0003
Unsigned Integer16	0x0004
Signed Integer16	0x0005
Unsigned Integer32	0x0006
Signed Integer32	0x0007
Unsigned Integer64	0x0008
Signed Integer64	0x0009
Float (2)	0x000a
Double (2)	0x000b
IP address (Ipv4)	0x0010
IP address (Ipv6)	0x0011
Time_SEC (3)	0x0012
Time_MSEC_64(4)	0x0013
Time_USEC_64 (5)	0x0014
Time_MSEC_32 (6)	0x0015
Time_USEC_32 (7)	0x0016

The variable length data types are defined as following:

String (8)	0x400c
Null Terminated String	0x400d
UTF-8 String	0x400e
UTF-16 String	0x400f
Arbitrary Data (BLOB) (9)	0x4015

(1) Boolean is represented as a single octet holding 0 for a value of FALSE and 1 for a value of TRUE.

(2) Float and Double are single and double precision floating point numbers that comply with the IEEE-754 standard.

(3) Time_SEC is a 32 bit value, most significant octet first - seconds since 00:00:00 GMT, January 1, 1970.

(4) Time_MSEC_64 is a 64 bit value, most significant octet first - milliseconds since 00:00:00 GMT, January 1, 1970.

(5) Time_USEC_64 is a 64 bit value, most significant octet first - microseconds since 00:00:00 GMT, January 1, 1970.

(6) Time_MSEC_32 is a 32 bit value, most significant octet first - milliseconds since 00:00:00 GMT, January 1, 1970.

(7) Time_USEC_32 is a 32 bit value, most significant octet first - microseconds since 00:00:00 GMT, January 1, 1970.

(8) String is prefixed by a 32 bit length field that indicates the length of the string, and followed by ASCII codes of the string characters. This representation MUST only be used for encoding data records in a "DATA" message.

(9) The arbitrary data is prefixed by a 32 bit length field and followed by the data in binary format.

Key Attribute Vector: 32 bit unsigned integer

The Key Attribute Vector field indicates different attributes of the key. In CRANE Version 1, only the 'K' bit is defined, other bits in the field SHOULD be set to zero by the sender and ignored by the receiver.

The 'K' bit ('Disabled bit') is set to 1 when the key is disabled in this template.

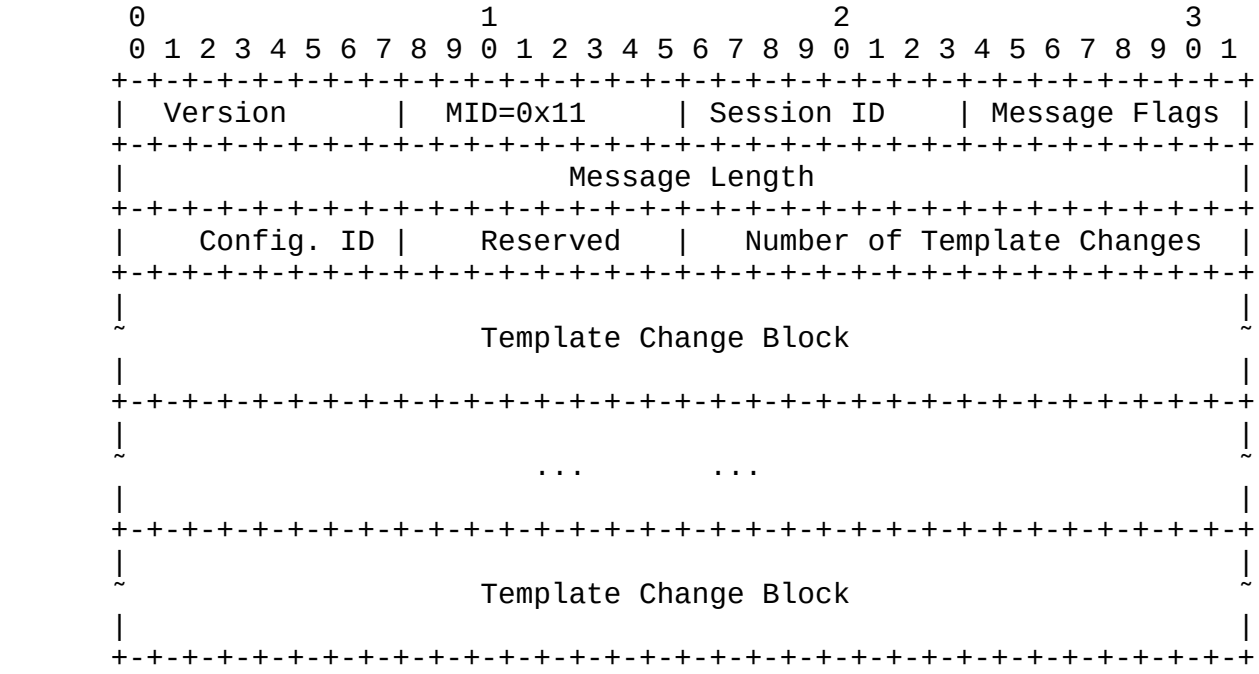
4.7 Template Data Acknowledge (TMPL DATA ACK)

Description

The Template Data Acknowledge message is sent from a CRANE server to a CRANE client after a TMPL DATA message has been received. It proposes changes of the templates and/or key status changes (enable/disable) for the templates.

If a CRANE server wishes to acknowledge reception of TMPL DATA without changes, it MUST respond with the FINAL TMPL DATA ACK message.

Message Format



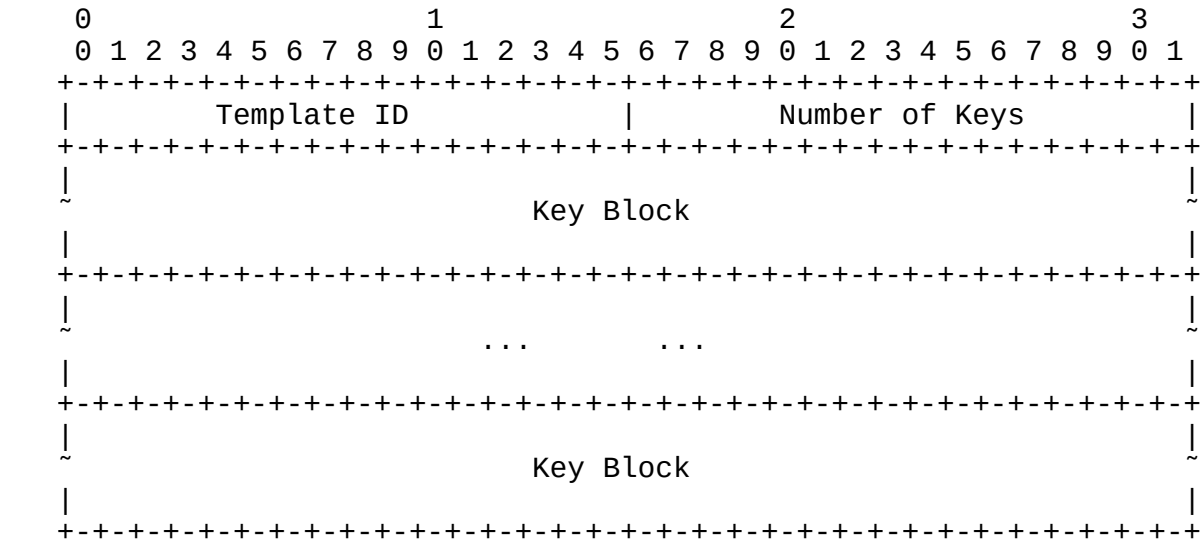
Configuration ID (Config. ID): 8 bit unsigned char

See Section 4.6. The value MUST be identical to the Config. ID field of the acknowledged TMPL DATA message.

Number of Template Changes: 16 bit unsigned integer

The Number of Template Changes field is the number of changed Templates (a changed template is described by a Template Change Block) specified by the message.

Template Change Block



Template ID: 16 bit unsigned integer

See Section 4.6.

Number of Keys: 16 bit unsigned integer

See Section 4.6.

Key Block

See Section 4.6, only relevant keys are described.

4.8 Final Template Data (FINAL TMPL DATA)

Description

The Final Template Data message is sent by a CRANE client to all the CRANE servers in a session, to convey the finalize templates. It is similar to the TMPL DATA message, with the only difference that a server must accept the templates in this message.

Message Format

Identical to the TMPL DATA (see section 4.6)

Message ID (MID)

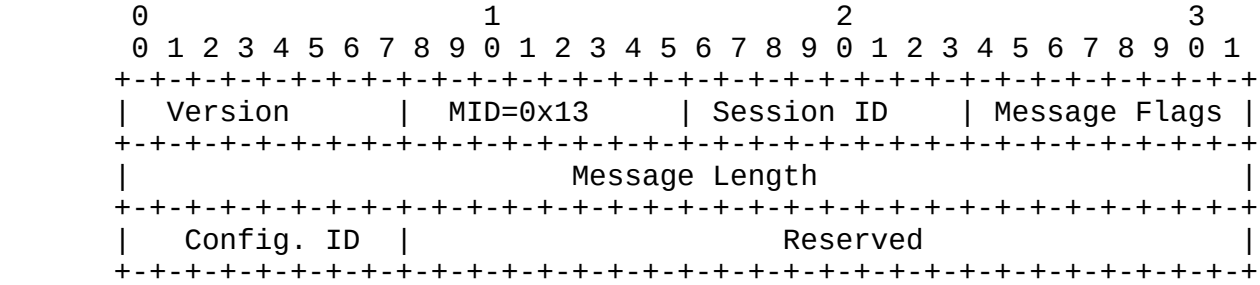
0x12 Final Template Data

4.9 Final Template Data Acknowledge (FINAL TMPL DATA ACK)

Description

The CRANE server acknowledges reception of the TMPL DATA or FINAL TMPL DATA by sending a Final Template Data Acknowledge message. It does not carry any changes to the templates. Unlike TMPL DATA ACK messages, a FINAL TMPL DATA ACK message indicates the acceptance of the templates for the session. A server MAY respond with this message to a TMPL DATA (if it does not want any changes in the templates). A server MUST respond with this message to a FINAL TMPL DATA.

Message Format



Configuration ID: 8 bit unsigned char

See Section 4.6. This field MUST copy the configuration ID from the acknowledged message.

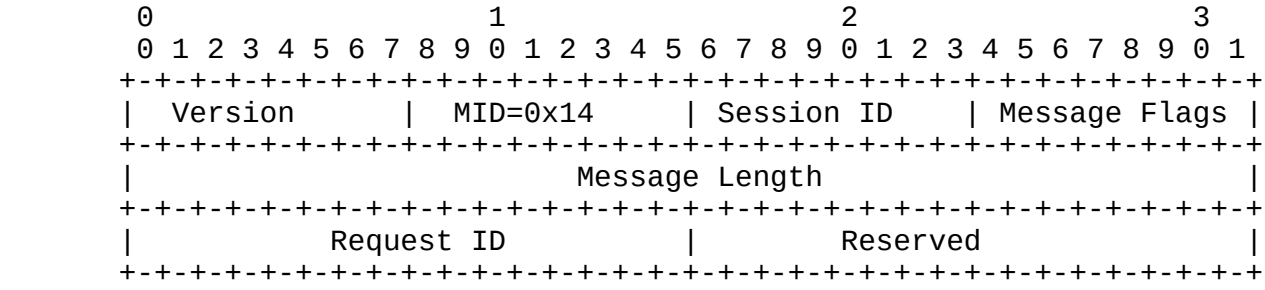
4.10 Get Sessions (GET SESS)

Description

The Get Sessions message is sent by a CRANE server to a CRANE client to query what are the sessions it should participate. This is typically done just before a UI configuration of the CRANE client's templates. As each session has its own set of templates, there is a need to know the server's participation of all the sessions.

The Session ID field in the CRANE message header MUST be ignored by the receiver.

Message Format



Request ID: 16 bit unsigned integer

The Request ID field identifies the specific request issued by the server. The same Request ID MUST be placed in the responding message in order to associate it with the request.

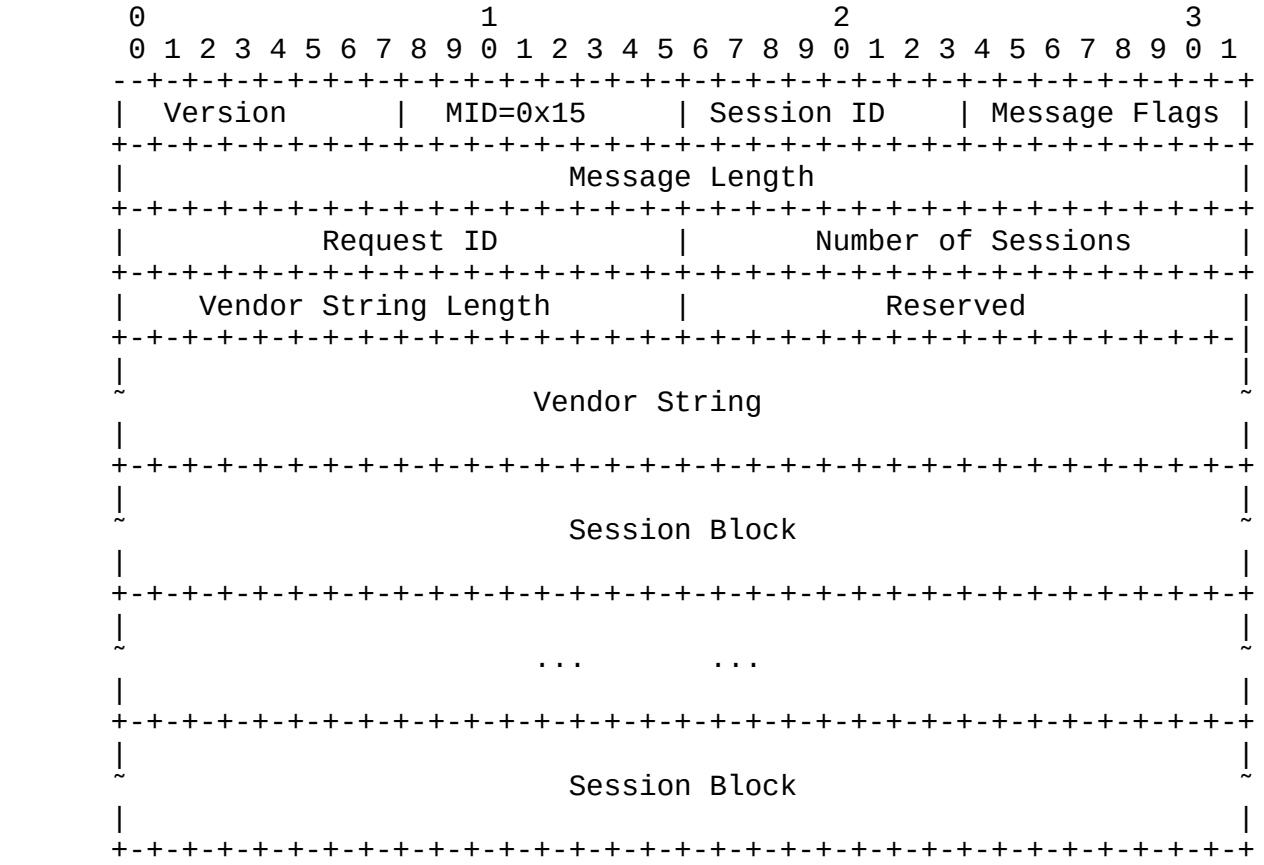
4.11 Get Sessions Response (GET SESS RSP)

Description

The Get Sessions Response message is sent by a CRANE client to a CRANE server as a reply to a GET SESS request. The message MUST contain all the information related to any session with which the requesting server is associated.

The Session ID field in the common message header MUST be ignored by the receiver.

Message Format



Request ID: 16 bit unsigned integer

See Section 4.10.

Number of Sessions: 16 bit unsigned integer

The Number of Sessions field is the number of session blocks in the message.

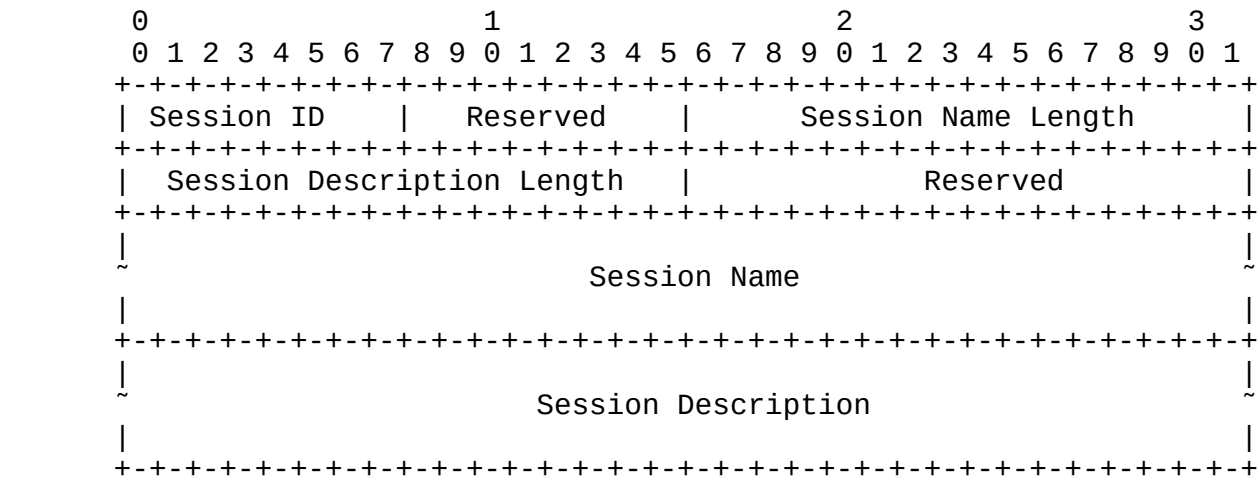
Vendor String Length: 16 bit unsigned integer

The Vendor String Length field is the length of Vendor String field in octet. The field limits vendor strings to 64Kb long. If no such string is supplied, the length MUST be set to 0.

Vendor String: Variable length unsigned char

The Vendor String field is a variable length field. It identifies the vendor that created the session. It MUST be padded with 0 to the next 32 bit boundary. The information differentiates similar templates from different vendors. The actual format of the information is application specific and outside the scope of this document.

Session Block



Session ID: 8 bit unsigned char

See Section 3.

Session Name Length: 16 bit unsigned integer

The Session Name Length field is the length of the Session Name field. The field limits the session name strings to 64 Kb long. As a name is mandatory to differentiate between sessions, this field MUST NOT be 0.

Session Description Length: 16 bit unsigned integer

The Session Description Length field is the length of a session description. The field limits the session description to 64Kb long. If no such Description is supplied, the length MUST be set to 0.

Session Name: Variable length unsigned char

The Session Name field is the name for a session, which MAY be displayed to end-users. It MUST be padded with 0 to the next 32 bit boundary. Session Name MUST be unique within a CRANE client. This field is mandatory and MUST be a part of any Session Block.

Session Description: Variable length unsigned char

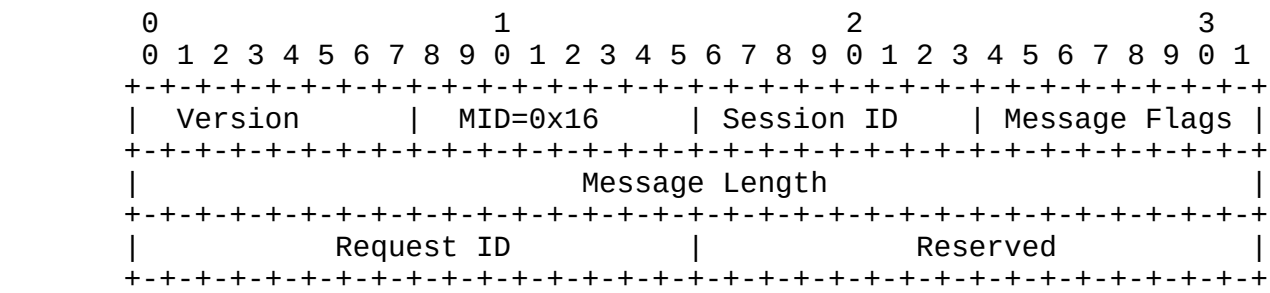
The Session Description field is the text description of a session; it could be displayed to end-users. It MUST be padded with 0 to the next 32 bit boundary.

4.12 Get Templates (GET TMPL)

Description

The Get Templates message is sent by a CRANE server to a CRANE client to query templates in a session.

Message Format



Request ID: 16 bit unsigned integer

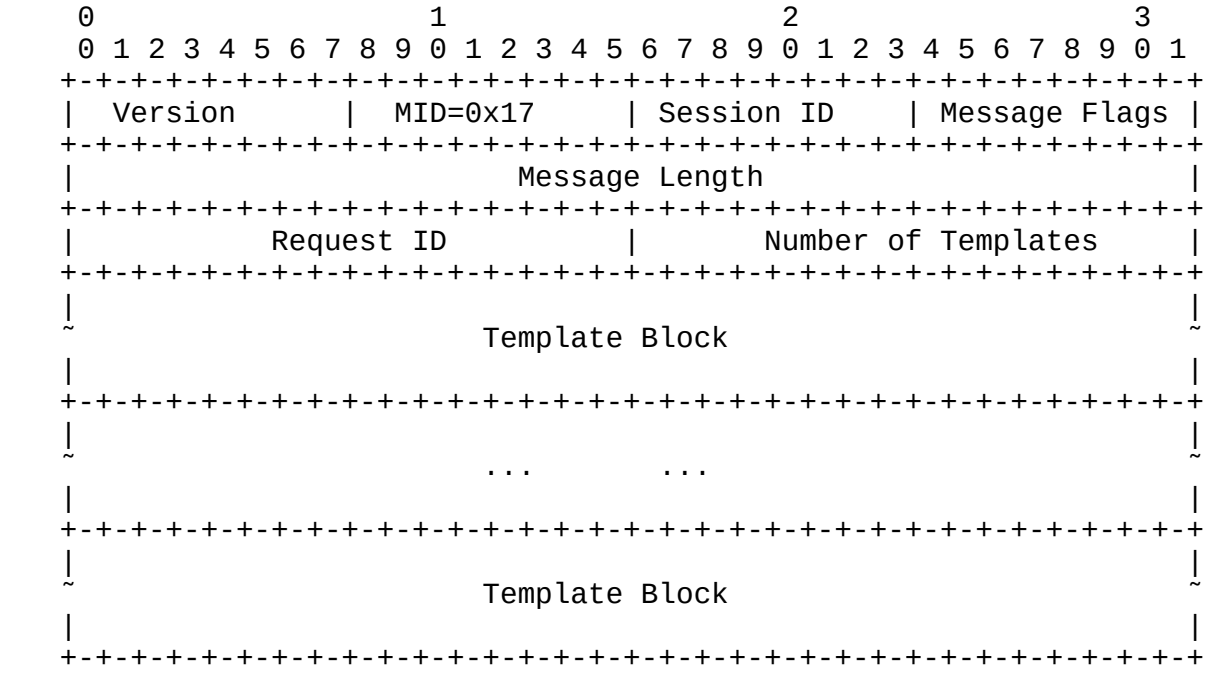
See Section 4.10.

4.13 Get Templates Response (GET TMPL RSP)

Description

The Get Templates Response message is sent by a CRANE client to a CRANE server as a response to a GET TMPL message. The message SHOULD contain all templates available for the specific session.

Message Format



Request ID: 16 bit unsigned integer

See Section 4.10.

Number of Templates: 16 bit unsigned integer

See Section 4.6.

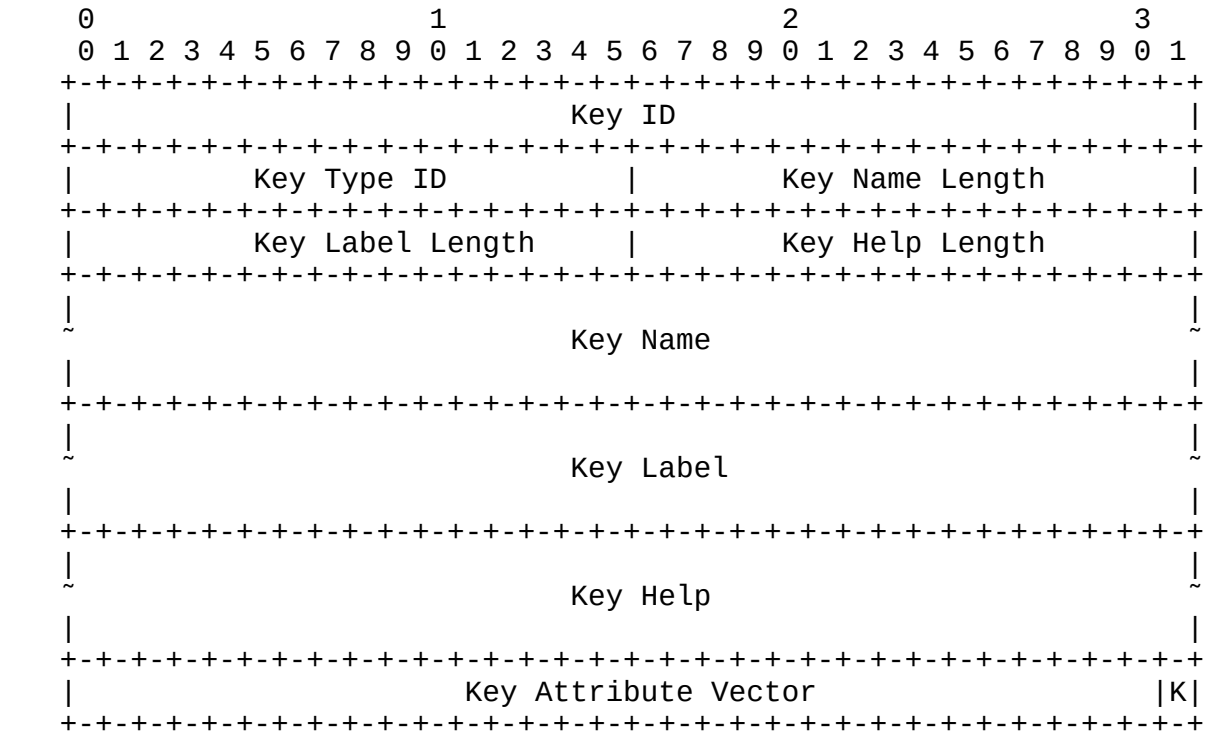
Template Block

Same as the template block defined in the TMPL DATA message (see Section 4.6). However, Extended Key Blocks MUST be used instead of Key Blocks. Extended key Block field provides extensive informational data that MAY be displayed to end-users.

Extended Key Block

The Extended Key Block field provides comprehensive information about a key.

Extended Key Block Format:



Key ID: 32 bit unsigned integer

Same as section 4.6.

Key Type ID: 16 bit unsigned integer

Same as section 4.6.

Key Name Length: 16 bit unsigned integer

The Key Name Length field is the length of the Key Name field. The field limits Key Name strings to 64 Kb long. As a name is mandatory to a key, this field MUST NOT be 0.

Key Label Length: 16 bit unsigned integer

The Key Label Length field is the length of the Key Label field. The field limits Key Label strings to 64 Kb long. Length of 0 means that the Label field is to be skipped.

Key Help Length: 16 bit unsigned integer

The Key Help Length field is the length of the Key Help field. The field limits Key Help strings to 64 Kb long. Length of 0 means that the Help field is to be skipped.

Key Name: Variable length unsigned char

The Key Name field is the name for the key, which could be displayed to end users. It MUST be padded with 0 to the next 32 bit boundary. Key Name MUST be unique (within the template) and case sensitive. This field is mandatory and MUST be a part of any Extended Key Block.

Key Label: Variable length unsigned char

The Key Label field is a descriptive label, which could be displayed to end users concerning this key. It MUST be padded with 0 to the next 32 bit boundary. This field SHOULD be a part of any Extended Key Block.

Key Help: Variable length unsigned char

The Key Help field is any Help string that could be displayed to end users concerning this key. It MUST be padded with 0 to the next 32 bit boundary. This field MAY be a part of any Extended Key Block.

Key Attribute Vector: 32 bit unsigned integer

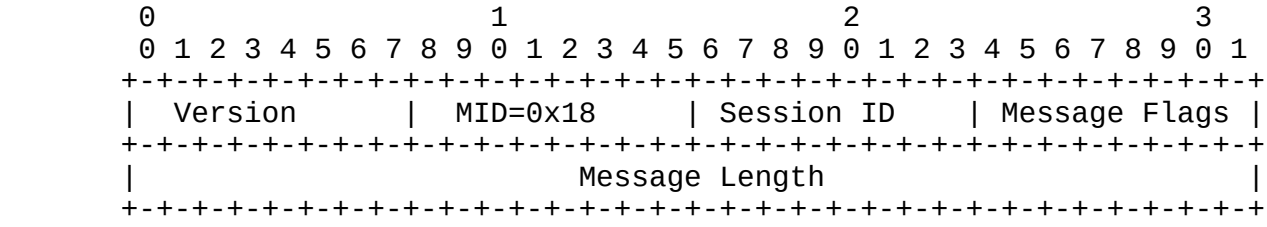
Same as section 4.6.

4.14 Start Negotiation (START NEGOTIATE)

Description

The Start Negotiation message is sent by a CRANE server after the configuration process has completed. The message should initiate template negotiation by the client with all CRANE servers in a session. The CRANE server MAY re-send this message up to 3 times with repeat interval of 5 seconds unless it is acknowledged by the CRANE client. Otherwise, the CRANE user will be informed. The client should send TMPL DATA message to the servers after acknowledged the message.

Message Format

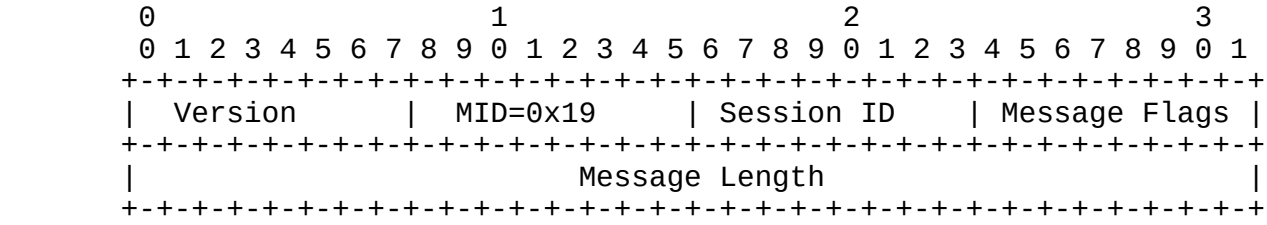


4.15 Start Negotiation Acknowledge (START NEGOTIATE ACK)

Description

The Start Negotiation Acknowledge message MUST be sent by a CRANE client to the server to acknowledge the reception of the START NEGOTIATE message.

Message Format

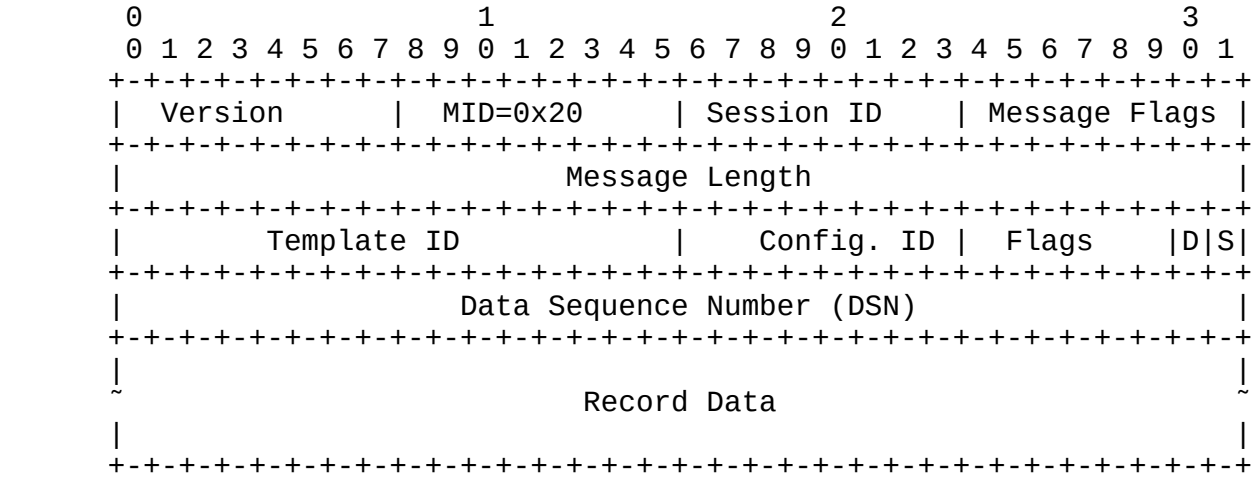


4.16 Data (DATA)

Description

The DATA message carries actual data records from a CRANE client to a CRANE server. A data record is a structured collection of fields that matches a specific template.

Message Format



Template ID: 16 bit unsigned integer

See Section 4.6.

Configuration ID: 8 bit unsigned char

See Section 4.6. The Config. ID field can prevent out-of-the-blue messages with outdated templates arriving and erroneously processed. A server MAY keep a short history of templates in order to cope with this scenario.

Flags: 8 bit unsigned char

The Flags field is composed of flag bits that indicate processing requirements of the data records. The CRANE Version 1 defined two flags for these purposes. Unless otherwise specified, the other flags are set to zero on transmit and are ignored on receipt.

The following flags are defined in CRANE Version 1:

The 'D' bit ('Duplicate' bit): It is set for records that are re-sent to an alternate server after a server transition occurs. When the same records are sent to different servers, there is a possibility that duplicated data exists. The Status of the 'D' bit will help the billing/mediation system to perform de-duplication if desired.

The 'S' bit ('DSN Synchronize' bit): When set, it indicates that the record is the first one received by the server after starting (or restarting) of data transmission to this server. The server MUST set the initial DSN to the DSN specified in the record. The flag is set to zero by default.

Data Sequence Number: 32 bit unsigned integer

The Data Sequence Number field is the record sequence number used for preserving data orders and detecting data losses. The DSN MUST be incremented by one for each new record transmitted. The selection of the initial DSN number is implementation specific.

Record Data: Variable Length unsigned octets

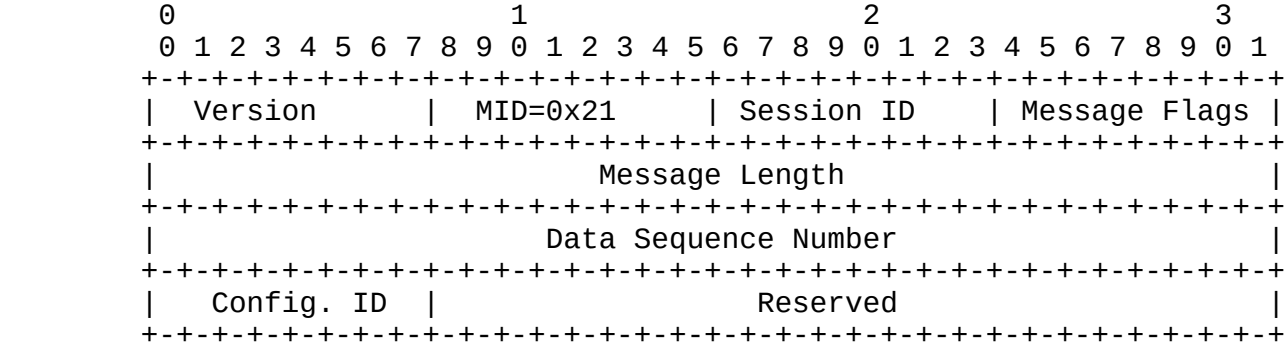
The Record Data field carries the actual accounting/billing data that is structured according to the template identified by the Template ID field.

4.17 Data Acknowledge (DATA ACK)

Description

The Data Acknowledgement message is sent from a CRANE server to acknowledge receipt of records. It acknowledges the maximal in-sequence DSN received.

Message Format



Data Sequence Number: 32 bit unsigned integer

See Section 4.16. It MUST be DSN of the last in-sequence record that was received by the server.

Configuration ID: 8 bit unsigned char

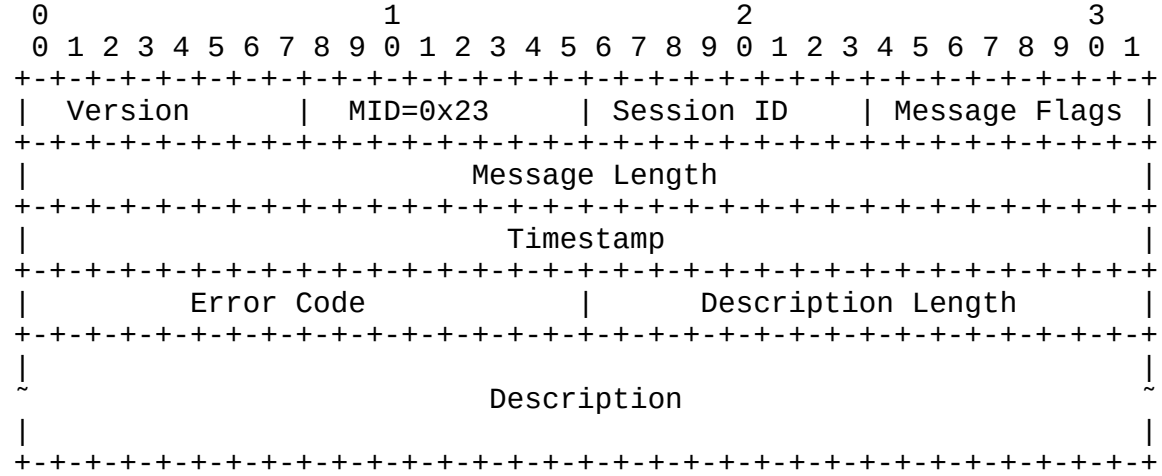
See Section 4.16.

4.18 Error (ERROR)

Description

The Error message MAY be issued by either a CRANE server or client. It indicates an error condition that was detected by the sender.

Message Format



Timestamp: 32 bit unsigned integer

The Timestamp field is a timestamp in seconds since 00:00:00 GMT, January 1, 1970.

Error Code: 16 bit unsigned integer

The Error Code field is a code assigned to an error condition.

The following error codes are defined in CRANE Version 1:

Error Condition	Error Code
Unknown	0

Description Length: 16 bit unsigned integer

The Description Length field is the length of the Description field. The field limits Description strings to 64 Kb long. Length of 0 means that the Description field is to be skipped.

Description: Variable Length unsigned char

The Description field is a text description that allows the sender to provide more detailed information about the error condition. It MUST be padded with 0 to the next 32 bit boundary.

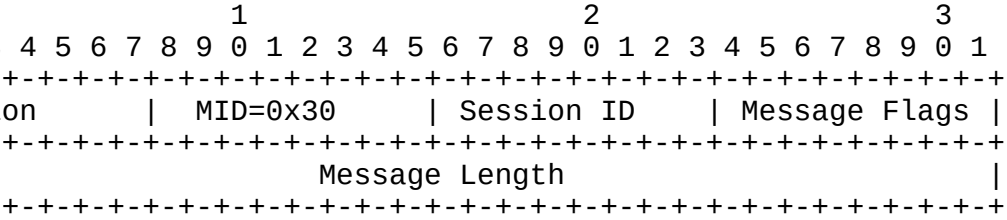
4.19 Status Request (STATUS REQ)

Description

CRANE servers MAY inquire general operation status of a client by sending the Status Request message. The status information SHOULD include a collection of states, counters, accumulators of the data collection functions that reside with the client. The status MAY include more information about the CRANE client itself.

The status reporting mechanism relies on the status template of a session. It is determined similarly as other templates. Without a determined status template, no status information can be delivered.

Message Format

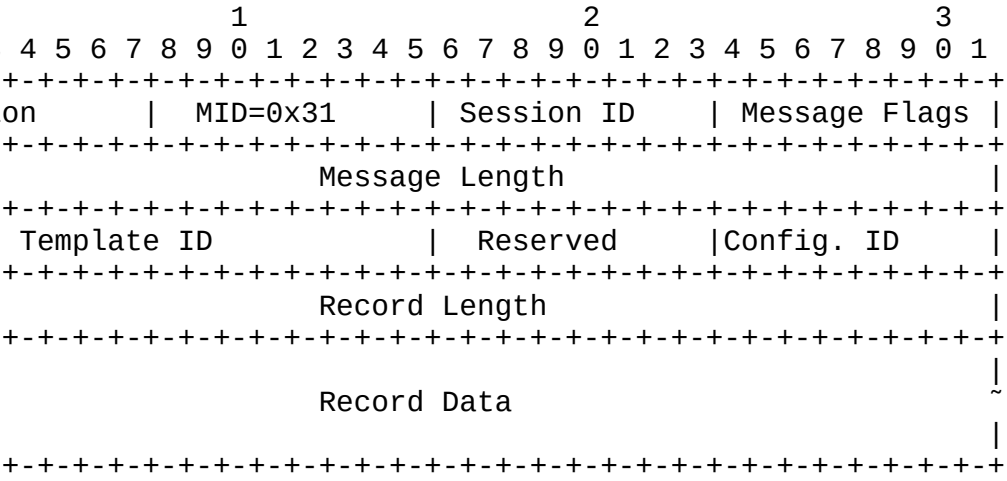


4.20 Status Response (STATUS RSP)

Description

The Status Response message contains a status report that MUST be compatible with the status template of the session. It is client's response to a STATUS REQ message from a server.

Message Format



Template ID: 16 bit unsigned integer

See Section 4.6.

Configuration ID: 8 bit unsigned integer

See Section 4.6. The version is needed here to prevent out-of-the-blue messages with outdated templates arriving and erroneously processed. A server MAY keep a short history of templates in order to cope with this scenario.

Record Length: 32 bit unsigned integer

The Record Length field is the length of the Record Data field in octets.

Record Data: Variable Length unsigned octets

The Record Data field contains the status data that complies with the status template. For more details see section 2.4

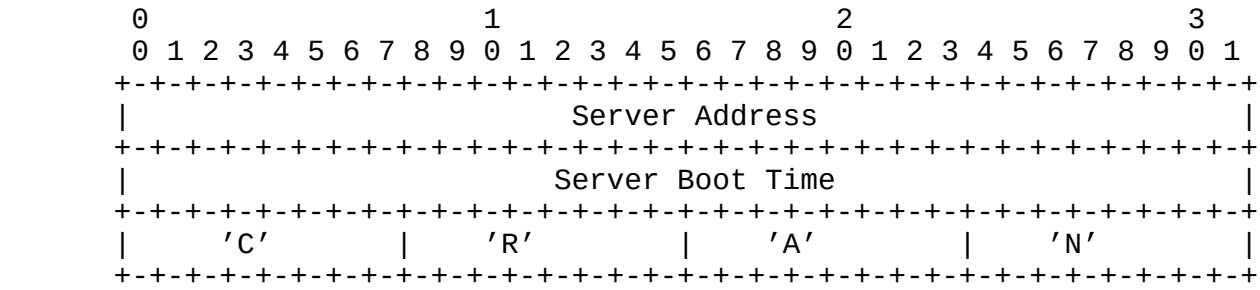
5 Protocol Version Negotiation

Since the CRANE protocol may evolve in the future and it may run over different transport layers, a transport neutral version negotiation mechanism running over UDP is defined. A CRANE server MAY inquire a CRANE client about the CRANE protocol version and transport layer support by sending a UDP packet on an agreed UDP port. The client MUST respond to this request with a UDP packet carrying the protocol

version, the transport type and the port number used for the specific transport. The Protocol Version Negotiation is optional for CRANE Version 1.

The CRANE server sends the following message to query the client's protocol support.

Message Format



Server Address:

The Server Address field is the IP address (Ipv4) of the CRANE server.

Server Boot Time

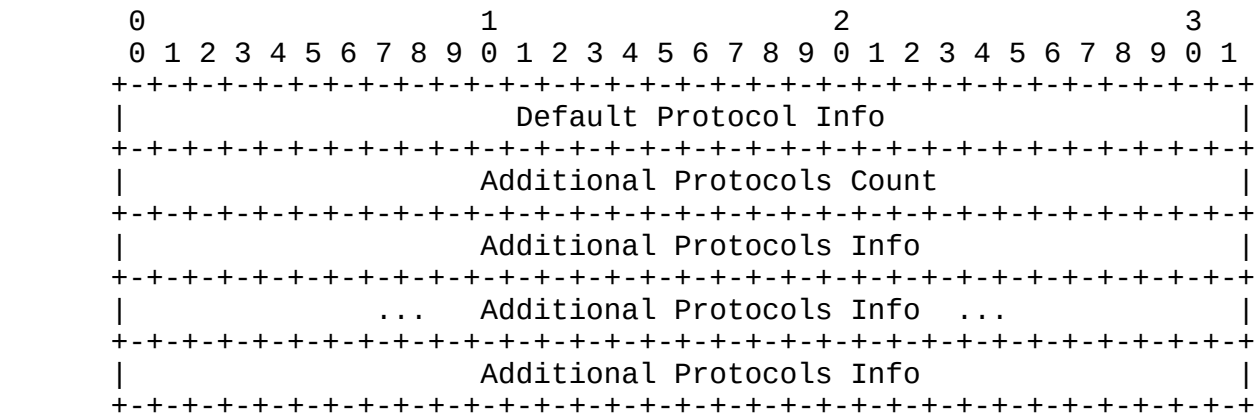
The Server Boot Time field is the timestamp of the last server startup in seconds from 1970.

'C', 'R', 'A', 'N':

The 'C', 'R', 'A', 'N' fields are ASCII encoded characters to identify the CRANE server.

The client's reply to a version negotiation request MUST comply with the following structure:

Message Format



Default Protocol Info:

The Default Protocol Info field contains information of the default protocol supported by the client. The field is structured as a Protocol Info Block described below.

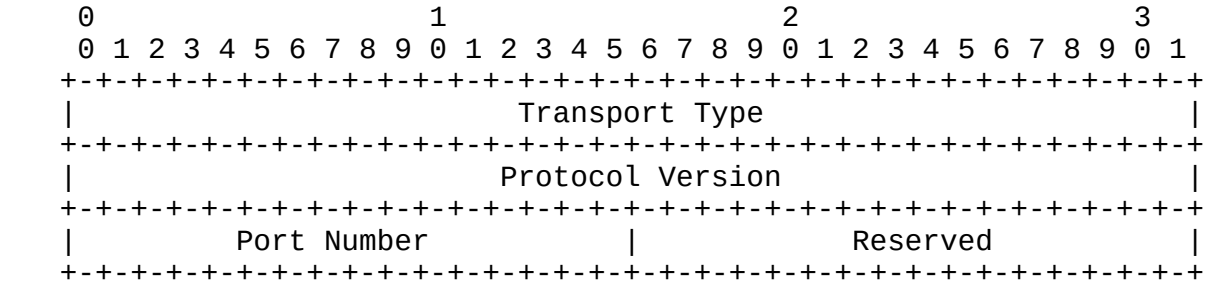
Additional Protocols Count: 32 bit unsigned integer

The Additional Protocols Count field specifies the number of additional protocols supported by the client. In the case that only the default protocol is supported, the field MUST be set to 0.

Additional Protocols Info:

The Additional Protocol Info field is an array of Protocol Info Blocks (described below) contain information about additional protocols supported by the client.

Protocol Info Block



Transport Type: 32 bit unsigned integer

1 - TCP, 2 - SCTP

Protocol Version: 32 bit unsigned integer

Version number of the CRANE protocol supported over the specific transport layer, the current version is 1.

Port Number: 16 bit unsigned integer

Port number (either SCTP or TCP port) used for the protocol

6 Security Considerations

The CRANE protocol can be viewed as an application running over a reliable transport layer, such as TCP and SCTP. The CRANE protocol is end-to-end in the sense that the CRANE messages are communicated between clients and servers identified by the host address and the transport protocol port number. Before any CRANE sessions can be initiated, a set of CRANE servers' addresses should be provisioned on a CRANE client. Similarly, a CRANE server maintains a list of CRANE clients' address with which it communicates. The provisioning is typically carried out securely using a network management system; in this way, the CRANE end-points can be authenticated and authorized. As this scheme is static, without additional security protections the CRANE protocol is vulnerable to attacks such as address spoofing.

The CRANE protocol itself does not offer strong security facilities; therefore, it cannot ensure confidentiality and integrity of CRANE messages. It is strongly recommended that users of the CRANE protocol evaluate their deployment configurations and implement appropriate security policies. For example, if the CRANE protocol is deployed over a local area network or a dedicated connection that

ensure security, no additional security services or procedures may be required; however, if CRANE clients and servers are connected through the Internet, lower layer security services should be invoked.

To achieve a strong security protection of communications between CRANE clients and servers, lower layer security services are strongly recommended. The lower layer security services are transparent to the CRANE protocols. Security mechanisms may be provided at the IP layer using IPSEC [6], or it may be implemented for transport layer using TLS [7]. The provisioning of the lower layer security services is out of the scope of this document.

7 References

[1] Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[2] Calhoun, P., "DIAMETER Base Protocol", Work in Progress.

[3] Calhoun, P., et. al., "DIAMETER Framework Document", Work in Progress.

[4] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Simple Control Transmission Protocol", RFC 2960, October 2000.

[5] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[6] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[7] Dierks, T. and C. Allen, "The TLS Protocol, Version 1.0", RFC 2246, January 1999.

8 Acknowledgments

Special thanks are due to Tal Givoly, Limor Schweitzer for conceiving the work, and Nir Pedhatzur, Batya Ferder, and Peter Ludemann from XACCT Technologies for accomplishing the first CRANE protocol implementation.

Thanks are also due to Nevil Brownlee for his valuable comments on the work, as well as the IETF IPFIX WG.

9 Authors' Addresses

Kevin Zhang
10124 Treble Court
Rockville, MD 20850
U.S.A.

Phone +1 301 315 0033
EMail: kevinzhang@ieee.org

Eitan Elkin
XACCT Technologies, Ltd.
www.xacct.com
12 Hachilazon St.
Ramat-Gan, Israel 52522

Phone +1 972 3 576 4111
EMail: eitan@xacct.com

10 Full Copyright Statement

Copyright (C) The Internet Society (2002). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.