Internet-Draft
Category: Informational
March 26, 2001
Expires in six months

Tom Arnold (CyberSource)
Jason Eaton (Jungawunga.com)

Simple Commerce Messaging Protocol (SCMP)
Version 1 Message Specification
(draft-arnold-scmp-08.txt)

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.

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

1. Introduction

The Simple Commerce Messaging Protocol (SCMP) is a general-purpose electronic commerce message protocol for secure, real-time communication of a set of data from a sending agent's application to a receiving agent's server. Additionally the response by the receiving agent's server to the sending agent is the reply from the request represented by the set of data in the message's payload. The intent of this protocol is to define a method where trading partners can perform on-line business requests in an environment where the sending partner is fully authenticated, and the message cannot be repudiated.

The taxonomy of the SCMP message payload is not in the scope of this document. The SCMP protocol does not specify payload definitions or how trading partners are expected to process the payload, beyond basic server-level functions related to processing SCMP headers. This intent

is to permit trading partners the flexibility to implement either a standard commerce message format as in ANSI-X12 Electronic Data Interchange (EDI) or some other non-standard payload format. This document does give an example implementation of a payload format based on [XML].

The only requirement on the message payload is that it be prepared as specified in [SMIME] section 3.1.

In this manner, SCMP fundamentally differs from many emerging commerce message protocols. Beyond specifying the method for encryption, authentication and handling, these other protocols specify the contents of the message and details how a server is to process and respond to a given message payload.

In this version of the protocol, a requirement has been added that describes the process trading partners will implement to agree on payload content formats. To this end, section 2.2.2 has been modified to accommodate the identification of payload content format, and an appropriate example has been appended.

NOTE from the Authors:

Our intent has always been to propose a secure, request and reply messaging protocol that is unencumbered by the negotiation and debates around commerce message content (payload). One need only consider the number of proposed credit card payment "standards" to become fully embroiled in this issue.

By including a requirement related to the negotiation and agreement on a payload format along with implementation specifications, we intend to complete this specification so as to facilitate implementations and interoperability.

We fully recognize that businesses will evaluate the proposed content standards from industry consotiums, IETF, RosettaNet, CompTIA and others then decide on a payload format that meets their requirements. >From this point, we feel SCMP is well suited as a method to implement the transaction messaging between their busines and other businesses.

1.1. Terminology

Throughout this draft, the terms MUST, MUST NOT, SHOULD, and SHOULD NOT are used in conformance to the definitions in RFC2119 [MUSTSHOULD].

1.2. Definitions

Several terms will be used when specifying SCMP.

Trading Partners	Two entities wishing to perform some on-line request processing where authentication, privacy, integrity and non-repudiation of the requests are important. Trading partners have established a trusted relationship between each

	other.
Client	An application program that executes on a remote system, used by a trading partner to request services from a server via an untrusted or publicly switched packet network, like the Internet.
Server	An application program used to process SCMP messages received from a client, and generate appropriate replies which are sent back to the client.
Sending Agent	An entity that operates or uses a Client for requesting on-line services from a server.
Receiving Agent	An entity that operates a Server, receives and processes requests from a plurality of Clients.
Request	An SCMP formatted message containing a set of directives set in a textual form requesting a set of directives be executed on behalf of the sending agent.
Reply	An SCMP formatted message containing a set of result data generated as a result of processing an SCMP request.
Payload	The meaningful content provided by a client to a server, encapsulated in an SCMP message. Similarly the meaningful content provided by a server to a client, encapsulated in an SCMP message.
Services	Groups of operations and/or algorithms implemented by the server application which are executed as designated by the payload. Each available group of operations and/or algorithms is a service.

1.3. Document Overview

This document describes SCMP from the standpoint of how trading partners would implement a client/server request processing system via an untrusted network connection.

In a on-line, electronic commerce environment, trading partners require a scalable, message handling system that will meet these minimum requirements:

1.3.1 Real-time Request and Response

A single message containing all credentials and payload data is prepared by a sending agent and sent to a receiving agent. The

receiving agent, upon verification of sender's credentials, SHOULD process the payload, format a reply, and respond to the sender as the response to the request.

1.3.2 Message Privacy

Through use of cryptographic methods, the privacy of the sender's message payload MUST be assured in the event a message payload is intercepted.

1.3.3 Message Integrity

If a message arrives in an incomplete or tampered condition from a sending agent, the receiving agent's server MUST detect the condition, deny message payload processing, and respond with an appropriate error message.

1.3.4 Authentication of Sending and Receiving Agents

Messages between trading partners, as represented by sending and receiving agents, MUST contain attributes that assure a given request could only be from a specific trading partner. Additionally SCMP requests and SCMP replies MUST be authenticated.

Prior to performing any application functions on a SCMP request payload, the receiving agent's server MUST verify that the request has been made by an authorized sender.

1.3.5 Non-repudiation

When a receiving agent's server receives a request to process a payload, the receiving agent's application SHOULD guarantee that the sender cannot, at some later time, refute having sent the request.

Non-repudiation is defined as, the inability of either trading partner (sender or receiver) to refute the sending of an SCMP request or SCMP reply.

An Electronic Commerce provider will typically provide financial based functionality such as authorization, settlement, and crediting, of credit card accounts and/or merchant accounts. This functionality requires that the Electronic Commerce Provider execute financial transactions on behalf of the merchant, or trading partner. Therefore it is desirable that the transaction directives which are given in an SCMP message are non-refutable.

1.3.6 Payload Independence

The messaging system SHOULD perform consistently for all payload formats.

1.3.7 Standards Based

The messaging protocol SHOULD be based on proven, existing cryptography and Internet standards.

1.3.8 Use of Standard Credentials

Standard credential formats SHOULD be used to maximize interoperability of common Public Key Infrastructures.

1.3.9 Transport Independent

The message SHOULD be transportable over the most common Internet transport protocols.

1.3.10 Service Level Guarantee

The receiving agent SHOULD guarantee a response within the time designated by the sender, or reject the message with an appropriate error message.

1.3.11 State Independence

State dependency by either a sender's or receiver's application SHOULD be minimalized as to support multiple transport mechanisms.

1.3.12 Payload Content Agreement

The sending agent MUST agree to create message payloads in a format that is acceptable to the receiving agent's application server. This agreement can take a few different forms:

- Receiving agent publishes a private payload specification. This specification MUST define the application(s) that will be offered, all input and output data formats, processing rules and other information about the application behavior. The value for "SCMP-message-type" (see Section 2.2.2 in this document) will have the word "private-" prepended to any distinct value. Example:

    "SCMP-message-type: private-CYBSCommerceService/1.0"

- Receiving agent agrees to accept transactions that comply with a published industry standard or consortium format. The value for "SCMP-message-type" (see Section 2.2.2 in this document) will have the word "std-" prepended to any distinct value. Example:

    "SCMP-message-type: std-CXML/1.0.2"

- Receiving agent agrees to implement data format from sending agent's published specification. A local value for "SCMP-message-type" MUST be agreed to and implemented. Further, the word "private-" MUST be prepended to the value. Example:

    "SCMP-message-type: private-userCompanyShippingManifest/2.1"

- Receiving agent agrees to implement a data format that is a combination of any the aforementioned forms. In this situation, the value for "SCMP-message-type" header will be set by the sending agent's client application to appropriately match the data format for the message payload. There MUST NOT be a requirement that an individual trading partner relationship use one and only one payload format. Different formats MAY be used to accommodate different application functions.

2. SCMP Message Construction

The payload of an SCMP message is divided into two parts. The outer SMIME entity that contains the cryptographically enhanced payload and the inner MIME encapsulation of the payload. In this way the inner MIME message can be enveloped protecting the header information which may contain sensative data.

All of the header fields defined in this document are subject to the general syntactic rules for header fields specified in [RFC822].

Both the sending agent and receiving agent MUST specify all SCMP headers specified in this document.

2.1 Outer SMIME Message Construction

The outer SMIME message MUST be constructed in accordance with [SMIME] section 3.1. An SCMP compliant server SHOULD implement the three message types as described in [SMIME], signed, enveloped, and signed/enveloped. An SCMP compliant server MUST implement signed/envelope message type as described in [SMIME].

For non-repudiation concerns, the trading partners MUST exchange signed or signed/enveloped SCMP message types.

SCMP error messages MUST be of signed type and NOT encrypted.

In addition to the headers listed below use of any additional standard SMIME and MIME headers are assumed. These headers will most likely be ones that need to be processed prior to payload decryption.

2.1.1 SCMP Protocol Version

The SCMP-protocol-version header is used to designate the SCMP protocol version. Server implementations MAY reject the request based upon protocol version, before any message processing occurs.

An example SCMP-protocol-version header will be in this format:

```
SCMP-protocol-version: 2.0
```

The value of the protocol-version header MUST be in the following format, any number of digits, followed by a the special character ".",

followed by any number of digits. Where special character, and digits is defined in [RFC822].

If a particular protocol version is not supported by the implimentation, the receiving agent MUST reject the request with an appropriate SCMP error message.

2.2 Inner MIME Message Construction

The payload of an SCMP message MUST be prepared as a standard MIME entity as defined in the [MIME] specification.

The remainder of this section describes the payload-based extensions that MUST be implemented by both the client and server to ensure correct and proper request processing.

Setting the SCMP service headers is the responsibility of the sending agent's client application. Processing the SCMP payload headers is the responsibility of the receiving agent's server application processing the request.

The following headers are described for the inner MIME entity which contains the payload. Thus if the SMIME message type is signed/enveloped (which is recommended), then the SCMP headers will be encrypted with the sender's message payload.

Both the sending agent and receiving agent MUST specify all SCMP headers specified in this document.

2.2.1. Request Time to Live

This describes the amount of actual processing time in seconds the client expects the server to complete payload processing prior to responding with an appropriate reply.

An SCMP server receiving a SCMP message MUST evaluate the request time to live value and determine if it can execute the required service(s) in the amount of time designated. Assuming the server believes it can complete the work within the allowed time, it will accept the request. If not, the server MUST return an error to the client stating it could not accept the request.

Once a server has accepted a request, it MUST process it until the time to live value has been reached or until completion. If the time to live value is reached during execution, the server MUST return an error to the client stating that a time-out has occurred.

Application functions to ensure data consistency, integrity, or rollback after the time to live value has been exceeded will be the responsibility of the server application. A policy on what application actions a server will take upon exceeding a time to live value SHOULD be published by the receiving agent operating the server.

An example of a policy in this are would be one where a receiving agent's server will continue processing the request after a request time to live value has been exceeded. Given this policy, a client, having received a time-out error message, would send a "request status message" to the server, referencing the original scmp-request-id (from the message that timed out) in the message payload. The server's reply to this status message would be the reply that would have been sent had the processing time not exceeded the time to live metric.

The time to live header will be in this format:

    SCMP-request-time-to-live: 90

Where the value of the time-to-live header is a digit or digit(s) as specified in [RFC822]. The value of the time-to-live is represented as any number of digit(s) which will designate a number of seconds.

2.2.2. Message Type

This value specifies the type of payload that is contained in the SCMP message. The intent of this header is to provide a meta-level description of the message payload and allow a receiving server to decide which services or associated algorithms to use in processing the payload.

Message type is specified as follows:

    SCMP-message-type: [service-name]/[version-number]

Where service-name is text as specified in [RFC822] and version-number is a digit or digit(s), followed by the special character ".", followed by a digit or digit(s). Where digit and special character are defined in [RFC822].

For instance, if a service was published called "CommerceService", the SCMP-message-type would be represented as:

    SCMP-message-type: private-CYBSCommerceService/1

Trading partners MUST agree on payload data formats and the distinct value for the SCMP-message-type field before requests are processed.

If a particular message type is not supported by the implimentation, the receiving agent MUST reject the request with an appropriate SCMP error message.

2.2.3. Request ID

Request ID's MUST be generated by the client application, thus assuring that the scmp-request-id is available in the event that the

request cannot be sent to the server due to errors.

The format of value of the request id header is 22 digits, where digits is defined by [RFC822].

An example of a request scmp-request-id is:

```
scmp-request-id: 091729304909616790 4518
```

The scmp-request-id MUST be unique in the domain of a client application and SHOULD NOT be easy to predict so as to prevent a potential replay attack.

A client application, when preparing the scmp-request-id, SHOULD perform a random number generation with sufficient degrees of randomness, to ensure unpredictability, and generate a client side time value, to ensure uniqueness of the result. These two data items together SHOULD form the resulting scmp-request-id.

Servers MAY use a scmp-request-id as a reference and handle to the original request during server message processing.

Servers MUST return the submitted request id back to the client via the SCMP reply message in the SCMP-request-id header.

3. Transport Implementations

SCMP can be implemented using any variety of transport methods as defined by the service provider. Here are a few examples.

HTTP: This delivers a SCMP message to a server URL and SHOULD use a POST function. Used in this manner the SCMP reply would be the entity-body of the HTTP response. SCMP error messages would be the entity-body of the HTTP response.

SMTP: This will support a queued batch processing service. Used in this manner the SCMP messages would be the body of the SMTP message. SCMP error messages would be sent in the body of the SMTP message.

4. Receiving Server Functions

This section describes minimal server functions required to implement SCMP.

4.1. General

A SCMP server receives a message from a client, processes the message and generates a reply. If the message type is signed or signed/enveloped the server initially validates the outer signature. If the outer signature is not valid the server MUST NOT process the request further.

4.1.1. Message Timestamp

The time a request was received SHOULD be derived from the environment which recieves the message. Clients and servers SHOULD be synchronized using [NTP] or Secure NTP.

The message timestamp SHOULD be used, in combination with the scmp-request-id, by the server to aid in detection of a potential replay attack.

It is recommended that servers SHOULD run a client-visible NTP server to allow SCMP client applications to synchronize clocks as required.

4.1.2 Support for Request Non-Repudiation

Support for non-repudiation MUST be included in any complete SCMP implementation, as described in the following subsections.

Implemenations MAY support non-repudation of error message replies. This document addresses the non-repudation concerns of the server or receiving agent. The non-repudiation concerns of of the client or sending agent MAY be fulfilled by the same means as the server or receiving agent supports non-repudiation.

4.1.2.1 Client Message Signing

The client application MUST send signed or signed/enveloped message type as specified in [SMIME].

4.1.2.2 Server Message Signing

The server application MUST send signed or signed/enveloped message type as specified in [SMIME].

4.1.2.3 Server Processing

The receiving agent's server application evaluates the digital signature, thereby guaranteeing that the message payload has not been altered in transit, and that the message was, in fact, signed by a specific trading partner (client) who possess the proper credentials.

4.1.2.4 Server Accounting

The receiving agent's server application MUST store the original signed/ encrypted message in an unprocessed state along with the timestamp for identifying when the message was received.

4.1.2.5 Client Accounting

The sending agent's client application MAY store the original signed/ encrypted message in an unprocessed state along with the timestamp for identifying when the message was received.

4.1.2.6 Revocation

All messages signed by a sending agent's client application in accordance with [SMIME] and sent to a receiving agent's server SHALL be considered non-repudiable.

To satisfy the non-repudiation requirements, the receiver of the message MUST support revocation mechanisms for the certificates of the potential senders of the SCMP messages that are accepted by the server application.

4.2. Application issues

The server MUST evaluate the signature of the message, if the message is of signed or signed/enveloped type, prior to processing the message payload. In performing this authentication process, the server MUST validate the senders certificate and verify that the senders certificate is not listed in any available revocation systems.

Assuming the SCMP message's signature is valid, the server will process requests with the appropriate service designated by the SCMP-message-type value.

4.2.1. Request Serialization

A server SHOULD NOT guarantee serialized request processing. If request serialization is a application requirement, it is expected that all of the serialized transactions will be received in a single message payload or that other content specific serialization systems will be used.

4.2.2. Server Errors

During application processing, a server could encounter several classes of error conditions. The server MUST be capable of reporting an error as described in section 5 of this document. Error Detection may vary based on specific implementation.

Additionally, a server MUST be capable of detecting a duplicate scmp-request-id and reply to the sending client application with an appropriate SCMP error message. Duplicate request detection MUST be based on the scmp-request-id and the distinguished name of the signer to prevent potential denial of service attacks.

5. Protocol Level Error Messages

In general SCMP does not concern itself with application level errors. Such errors SHOULD be returned in an SCMP reply with appropriate application specific formatting.

5.1. Format

SCMP error messages MUST be signed SMIME messages. SCMP errors

MUST NOT be encrypted to permit clients to process encryption related errors.

The format of SCMP errors is:

```
SCMP <error number> <error message text>
```

Where the format of "error number" is a digit or digits as defined in [RFC822] and "error message text" is text as defined in [RFC822].

5.2. Client Application Error Handling

Client action in the case of error return is error specific and not defined. If the server fails to return any reply within the time to live requested (due to unspecified server or network failure) the client MAY re-send the request. Clients MUST NOT retry a request in an interval which is less than the time to live value of the original request.

6. Security Considerations

Security considerations are addressed throughout this document.

6.1 Encryption Strength

It is recommended that strong enough cryptographic methods be used to ensure authenticity, integrity, non-repudiation, and privacy of the payload.

6.2 Non-repudiation

Non-repudation implimentation is specified in section 4.1.2.

As addressed above, this document does not describe how a sending agent may support non-repudiation. The intent of this document does describe how a receiving agent can support non-repudiation.

If the receving agent accepts and processes a transaction after the private key of the sending agent has been comprimised, that request is refutable, or not non-refutable.

6.3 Public Certificate Considerations

6.3.1 Certificate Exchange

Every trading partner implementing SCMP SHOULD exchange certificates that have been issued and signed by one or more mutually trusted certificate authorities. Prior to establishing trading partner relationships, the sender and receiver SHOULD acquire mutually acceptable public root certificates from the agreed upon certificate authority or authorities.

Sending and receiving agents MAY utilize certificate only messages to exchange certificates as specified in [SMIME].

6.3.2 Certificate Authentication and Revocation

Trading partners, upon receiving or exchanging public key certificates for the first time, SHOULD validate the certificate and certificate chain before processing an SCMP request.

A server certificate revalidation policy, related to the frequency certificates are revalidated against a certificate authority's certificate revocation list, is not specified by SCMP. This matter is left as a policy decision for the operator of the SCMP server.

The timestamp of a certificate revocation event SHOULD be the time the private key was known to be comprimised, or the time that the revocation event was made.

6.4 Private Key Considerations

6.4.1 Private Key Generation

Private key generation SHOULD be of a secure manner as not to jepordize the integrety of the private key.

6.4.2 Private Key Storage

The sending agent, maintaining a SCMP client application, MUST maintain the private key in a secure location.

6.4.3 Private Key Revocation

Should a sending agent loose control of their private key, they MUST notify the agreed upon, trusted, certificate authority. This notification mechanism is not defined in this document, and SHOULD be done via an out of band mechanism.

6.5 Request Id

The request id MUST be unique as to prevent possible replay attack senarios.

7. SCMP Message Example

```
[ OUTER MIME START ]
Content-Type: application/pkcs7-mime
Content-Transfer-Encoding: base64
Content-Length: 1024
SCMP-protocol-version: 2.0

  [ INNER MIME START - enveloped entity ]
  SCMP-request-time-to-live: 90
```

```
  SCMP-message-type: private-CYBSCommerceService/2.0
  SCMP-request-id: 012345678901234567890
  Content-Type: application/pkcs7-mime
  Content-Transfer-Encoding: base64
  Content-Length: 512

    [SIGNEDPAYLOAD - a SignedData with payload as encapsulatedContent]
  [ INNER MIME END ]
[ OUTER MIME END ]
```

8. XML Payload Example

This section is intended to be an example implementation of a payload and is NOT required for this protocol. The parties communicating could agree on two "scmp-message-type" values. The first would be the exchange of the DTD template, (ie. scmp-message-type=widget xml definition). The second could be the actual data generated from that template, (ie. scmp-message-type=widget xml data). The DTD would be transfered defining the data format. The server could store the format for later transferring of these types of messages. An example DTD follows.

```
<!ELEMENT request ( merchant_id, order+ )>
<!ATTLIST request number NUMBER #REQUIRED>
<!ELEMENT request type (#PCDATA)>
<!ELEMENT merchant_id ( #PCDATA )>
<!ELEMENT order ( product_sku )>
<!ATTLIST order_number NUMBER #REQUIRED>
<!ELEMENT product_sku (#PCDATA)>
```

This DTD would produce data as follows.

```
<request number="1">
<request type="widget xml data">
  <merchant_id>Widget Maker</merchant_id>
  <order_number="1">
    <product_sku>0349003</product_sku>
  </order>
</request>
```

This XML data would be the payload of the SCMP. When the agent recieves this type of SCMP message they could validate the message format with the previously received template.

9. Author's Address

Tom Arnold
CyberSource Corporation
1295 Charleston Road
Mountain View, CA 94043
E-mail: toma@cybersource.com
Phone: 650-965-6000

Jason Eaton
CyberSource Corporation
1295 Charleston Road Mountain View, CA 94043
E-mail: jeaton@cybersource.com
Phone: 650-965-6000

10. Acknowledgements

The authors wish to recognize and thank several individuals (listed in alphabetic order) who have and continue to support the development of requirements and improvement of this protocol.

Mike Agostino (Vulcan), Ron Bose (LitleNet), David Burdett (Mondex), Leonard Cantor (IBM), Dan Corcoran (Equifax), Steve Crocker (Crocker Assoc.), Tony Curwen (Ingram Micro), Donald Eastlake (IBM), Richard Frank (Intertrust), James Gavin (Commercenet), Paul Guthrie (Brodia), Lauren Hall (SIIA), Bengamin Hipp (FUSA/Paymentech), Andy Jeffrey (Sonnet Financial), Helle Jespersen (IBM), Sean Kiewiet (Hypercom.com), Connie Lindgreen (IBM), Michael Myers (VeriSign), Allan Ottosen (PBS), John Pettitt (Beyond.com), Jesse Rendleman (CyberSource), Don Sloan (Tech Data), Carl Stucke (Equifax), Frank Tyksen (Portland Software), Huy Vu (VISA USA), Sean Youssefi (CobWeb)

11. References

[SMIME] B. Ramsdell, "S/MIME Version 3 Message Specification", RFC 2633, IETF, June 1998.

[MIME] "MIME Part1: Format of Internet Message Bodies", RFC 2045; "MIME Part2: Media Types", RFC 2046; "MIME Part 3: Message Header Extensions for Non-ASCII Text", RFC 2047; "MIME Part 4: Registration Procedures", RFC 2048; "MIME Part 5: Conformance Criteria and Examples", RFC 2049, IETF.

[MUSTSHOULD] "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, IETF.

[NTP] D. Mills. "Network Time Protocol", RFC 1119, IETF, September 1989.

[PKCS-7] B. Kaliski, "PKCS #7: Cryptograpic Message Syntax" RFC 2315, IETF, March 1998.

[RFC822] D. Crocker, "Standard for the format of arpa internet text messages", RFC 822, IETF, August 1982.

[X.520] "ITU-T Recommendation X.520: Information Technlogy - Open Systems Interconnection - The Directory: Selected Attributes Types, 1993.

[XML] Extensible Mark Up Language. See http://www.w3.org/TR