Internet Draft
draft-hoffman-rescap-protocol-01.txt
June 1, 1999
Expires in six months

Paul Hoffman
Internet Mail Consortium

The rescap Resolution Protocol

Status of this memo

Abstract

The rescap protocol is a general client-server resolution protocol that translates resource identifiers to a list of attributes. For instance, a rescap client can ask a rescap server for the attributes of a particular mail user. rescap is very light-weight and acts only as a resolution protocol, not a directory service.

1. Introduction

When an Internet client is accessing a resource, it is often valuable for the client to know the attributes of the resource before contacting the resource. For example, a mail user might want to know whether another mail user is able to natively display TIFF files before creating a message with a TIFF file in it. The rescap protocol is a simple, extensible client-server protocol that allows a client to easily find the correct rescap server for a particular resource and find the attributes for that resource.

This document specifies:
- How to find the rescap server for a particular resource
- The recap protocol
- The format for rescap requests and responses

- Required rescap items that must be supported
- Registration of the "rescap" scheme name for URLs

The protocol in this document attempts to meet all the requirements described in the rescap requirements document [RESCAP-REQUIRE]. It should be noted that rescap is explicitly not to be used as a service-discovery protocol; users that want such a capability should use the Service Location Protocol [SLP].

This document and others relating to the rescap protocol are being discussed on the rescap@cs.utk.edu mailing list. To subscribe, send a message to rescap-request@cs.utk.edu. An archive of the mailing list is available at <ftp://cs.utk.edu/pub/rescap>.

1.1 Terminology

Throughout this draft, the terms MUST, MUST NOT, SHOULD, and SHOULD NOT are used in capital letters. This conforms to the definitions in [MUSTSHOULD]. [MUSTSHOULD] defines the use of these key words to help make the intent of standards track documents as clear as possible. The same key words are used in this document to help implementors achieve interoperability.

Hexadecimal values are indicated as "xNN" or "xNNNN". For example, xA0B1 corresponds to the octet xA0 followed by the octet xB1.

2. Finding a rescap Server

A rescap client that wants to find the attributes for a particular Internet resource needs to find the rescap server for that resource. The client MUST look up the A record associated with the rescap server for the host name in the resource. When SRV records (described in [SRV]) become widely deployed, the client MAY instead use the SRV protocol to locate the rescap server.

To find the A record of the rescap server, the client prepends two special domain names to the host name in the resource. The first name prepended (just to the left of the host name) is "_rescap."; the second name prepended (just to the left of "_rescap." is an underscore followed by the URL scheme name.

For example, to find the rescap server that is authoritative for the URI "mailto:someone@example.com", the rescap client would resolve the A record of the name "_mailto._rescap.example.com" and use that value as the location of the rescap server.

2.1 Optional use of SRV Records

SRV records MAY be used for locating rescap servers. The request sent to the DNS server is somewhat different than is specified in [SRV] because the rescap client must indicate the type of resource that will

be resolved by the rescap server. The resource name is given as the URI scheme name, and it appears in the left-most part of the DNS name to be resolved. The URI scheme name has an underscore character (_) prepended to it, just as the service and prototype names do.

As described in section 3 of this specification, the rescap protocol runs over both the UDP and TCP protocols, and all compliant servers must run the rescap service on both protocols on the same host. It is inefficient to force the rescap client to look up SRV records for both protocols, and doubling the number of SRV records for the rescap protocol can have an adverse effect on the domain name system. Thus, rescap clients MUST only resolve rescap SRV records for the UDP protocol, and MUST assume that the same host that was resolved for the UDP protocol will support rescap over the TCP protocol as well. rescap clients MUST NOT attempt to resolve rescap SRV records for the TCP protocol.

For example, to use SRV records to find the rescap server that is authoritative for the URI "mailto:someone@example.com", the rescap client would resolve the SRV record for the name "_mailto._rescap._udp.example.com".

3. Protocol

The rescap protocol uses a single request-response model. That is, a rescap client connects to the rescap server, sends a single request, and waits for the response. The server sends a single response and closes the connection.

The requirement that the rescap protocol be light-weight and fast leads to the conclusion that it should run on UDP instead of TCP because UDP takes less system and network resources for short exchanges. However, UDP has many problems, including that long datagrams may get split up or lost. Thus, to make rescap reliable, rescap servers run on both UDP and TCP.

A rescap server MUST run the same service on both the UDP and TCP protocols, and MUST use the same port number for both services. The port number 283 has been reserved by IANA for rescap, and a rescap server SHOULD run on that port number. However, because the client is allowed to find the port number using SRV records, the rescap server can run on any UDP and TCP port.

A rescap client SHOULD make its initial request to the rescap server using UDP. However, if the rescap client expects that the response from the server to be longer than 512 octets (such as if it is asking for an attribute whose value is always longer than 512 octets), the client MAY choose to make the initial connection using TCP.

If a client sends a UDP request to the server named in an SRV record

but does not receive a response, the client SHOULD send the same request using TCP. If a client sends a UDP request to a server and receives an incomplete response, the client MUST send the same request using TCP. This is due to the fact that a later part of a response might modify or negate the meaning of an earlier part of a response.

4. Request and Response Format

Rescap requests and responses have a simple format. The format is optimized for simple processing in small clients. It is also optimized for size so that it is more likely that requests and responses can fit in single UDP datagrams.

The basic unit of a rescap request or response is the item. An item consists of exactly three parts:
 - a two-octet tag
 - a two-octet length
 - content whose length and structure are defined by the tag
Each of the three parts of an are in network byte order, as is the entire item.

The tag values are given in this document and in other documents that profile rescap for various resource types. The length is the length of the content of the item, measured in octets. Some tags define fixed-length content, while other tags define variable-length content.

A program parsing a request or response can easily skip over items whose tags it doesn't recognize by reading the tag, reading the length, and skipping over the number of octets given in the length to get to the beginning of the next item.

There are three types of items: request-type items, response-type items, and data-type items. The type of the item is given in each item's specification.

4.1 Long content

The most significant bit of the two-octet length part of an item is a continuation marker. If the continuation marker is "1", the item has been split into two or more fragments. The item following an item whose continuation marker is "1" MUST have the same tag value as the preceding item. The first fragment that has a continuation marker of "0" is the last fragment in the item. When reconstructing a fragmented item, the content of all fragments are appended in the same sequence that they appeared in the data steam.

A consequence of this design is that any item whose length is greater than x7FFF MUST be split into at least two packets. Any item MAY be split into fragments. A process interpreting a stream of rescap items MUST be able to correctly handle long content. That is, if an item has

a continuation marker of "1" in the length part, the process MUST read the next item, check that the tag matches that of the preceding item, and append the content of the second item to that of the first item, until the process comes to an item with the continuation marker set to "0". It is an error if the process comes to an item with a different tag value than the preceding item if that preceding item had a continuation marker of "1".

Design note: the method of using a two-octet length and checking for the special value for the continuation marker was chosen instead of using a four-octet length in order to keep the size of the rescap response shorter, and therefore make it more likely that a response would fit in a single UDP datagram. The vast majority of items will probably have lengths less than x7FFF. The continuation marker allows a program to marshal fewer than x7FFF octets if it has memory constraints. The cost of having to check for the special case of a continuation marker seems to be worth the tradeoff of making every item two octets longer.

5. Basic requests and responses

The items in this section are included in a rescap client's request to a rescap server and in the server's response to the client. A rescap request is always a FullRequest item; that item contains other request-type items. A rescap response is always a FullResponse item; that item contains other response-type items.

5.1 Basic request types

Clients MUST be able to emit the FullRequest and BaseURI types, and SHOULD be able to emit ItemsToReturn. Clients MAY implement the PrivUseRequest types. Servers MUST be able to interpret the FullRequest, BaseURI, and ItemsToReturn types, although servers do not have to comply with the request in ItemsToReturn. For instance, a server may not want to return certain items due to lack of authorization or due to the response becoming too long.

5.1.1 FullRequest

The FullRequest item (tag x0001) is used to encapsulate the other request-type items. A rescap client's request to the server MUST be a single FullRequest item (unless the content of the FullRequest is longer than xFFFF octets, as described in section 4.1). The structure of the item is a sequence of one or more request-type items. A FullRequest item MUST NOT be included in a FullRequest item, and a FullRequest item MUST contain only request-type items. Note that some items contain other items; the prohibition against response-type and data-type items is only for items directly encapsulated in the FullRequest item, not items that are encapsulated in lower-level items.

A FullRequest item MUST contain exactly one instance of a BaseURI item, and MAY contain zero or one instances of each of the other request-type items.

5.1.2 BaseURI

The BaseURI item (tag x0002) specifies the Internet resource for which the rescap client wants information. The structure of the item is a URI string as defined in [URI] for a single resource. Every FullRequest MUST include exactly one BaseURI item.

5.1.3 ItemsToReturn

The ItemsToReturn item (tag x0003) lists the response-type items that the client would like the server to return. It is a request, not a demand; the server is allowed to return any response-type item it chooses. The client can use this item to specify that it only is interested in a limited number of response-type items, and that the server should not waste its time or bandwidth returning any items not listed in the ItemsToReturn item. The structure of the item is a sequence of tag values, each of which is two octets long.

If the ItemsToReturn item is not included in a request, the server may return whatever information it pleases about the resource named in the BaseURI item. If the ItemsToReturn item is included and its length is zero, the client wants the server to return all the information it has about the resource named in the BaseURI item. If the ItemsToReturn item is included and its length is greater than zero, the client wants the server to return only response-type items of the type listed in the ItemsToReturn item.

Note that it is not an error for the client to list items in the ItemsToReturn item that cannot be returned to by the server.

5.1.4 PrivUseRequest00 through PrivUseRequest255

The PrivUseRequest00 through PrivUseRequest255 items (tags xFE00 through xFEFF) are reserved for private use as request-type items and will never be assigned by IANA. These tags may be used by protocol developers to test protocols that they are developing, such as during the process of preparing Internet Drafts that contain registration for future request-type items. The structure of the items is undefined.

5.2 Basic response types

Servers MUST implement the FullResponse and Status types, and SHOULD implement the Referral type. Servers SHOULD implement the TTLOfInfo, ExpirationOfInfo, and DateOfChange types. Servers MAY implement the PrivUseResponse types.

Clients MUST be able to interpret the FullResponse, Status, and Referral types; clients SHOULD be able to process Referrals by fetching

the information from the referred-to host. A rescap client SHOULD NOT attempt to make any use of the user response-type items that it does not fully understand.

Note: clients MUST parse TTLOfInfo, ExpirationOfInfo, and DateOfChange items even if the clients do not know how to check the expiration or modification of information (such as a client that does not know when the information will be delivered to the user or a client that does not know the current time). These items encapsulate other response items that the client may want to deliver to the user regardless of the expiration status. Clients that cannot check the expiration or creation information MUST treat the enclosed response-type data as if it appeared outside of the time-specific types, and in the case of TTLOfInfo and ExpirationOfInfo, SHOULD indicate to the user that the server put an expiration on the information and that the expiration was not checked.

5.2.1 FullResponse

The FullResponse item (tag x000C) is used to encapsulate the other response-type items. A rescap server's Response to a client MUST be a single FullResponse item (unless the content of the FullResponse is longer than xFFFF octets, as described in section 4.1). The structure of the item is a sequence of one or more response-type items. A FullResponse item MUST NOT be included in a FullResponse item, and a FullResponse item MUST contain only Response-type items. Note that some items contain other items; the prohibition against request-type and data-type items is only for items directly encapsulated in the FullResponse item, not items that are encapsulated in lower-level items.

A FullResponse item MUST contain one or more instance of a Status item, and MAY contain zero or more instances of each of other response-type items. Note that some response-type items further restrict the number of times those items can appear in a FullResponse item.

5.2.2 Status

The Status item (tag x000D) gives status information to the client about its request. The structure of the status tag is a sequence of a one-octet main status code, a one-octet secondary status code, and an optional string that is a sequence of characters from the ISO/IEC 10646-1 character set encoded with the UTF-8 transformation format defined in [UTF8]. A FullResponse MUST have at least one Status item and MAY have more than one Status item. A rescap server SHOULD include only as many Status items as necessary for a client to process the response.

The optional string may be used to transmit status information, but it is optional. In fact, a rescap server that is trying to keep its

responses as short as possible SHOULD NOT include a string at all.

The rescap client MAY choose to display the string to the client. However, because there is no way to know the languages understood by the user, the string may be of little or no use to them.

Note: If a client sent a request over UDP and receives a Status item of x0201, the client SHOULD sent the same request over TCP.

The values for the main status code (the first octet of the Status item) are based loosely on those in SMTP. They are:
x00 - positive completion
x01 - transient negative completion
x02 - permanent negative completion

A rescap client MAY use just the main status code to decide how to display any results of the request to the user who made the request.

The complete list of status codes is:

x0000 The request was fully processable
x0001 Successful authorization through AuthInTheClear
x0002 Successful authorization through AuthDigest
x0003 Successful authorization through AuthPublicKey
x0004 Successful authorization through AuthIP
x0005 No valid KeyID items in the the SigningPrefs
x0006 No valid KeyID items in the SignedRequest
x0007 The content of the SignedRequest didn't validate against the signature

x0100 Too busy; try again whenever you feel like it
x0101 Too busy; try again later than 10 seconds from now
x0102 Too busy; try again later than 60 seconds from now
x0103 Too busy; the Referral item in the response leads to another rescap server authoritative for this resource

x0200 The body of the request was longer than what was indicated in the length
x0201 The body of the request was shorter than what was indicated in the length
x0202 The request included items not of request-type
x0203 The request included more than one BaseURI item
x0204 This server is not authoritative for the resource named in the BaseURI item and no referral is available
x0205 This server is not authoritative for the resource named in the BaseURI item; the URI in the Referral item leads you to a server that this server believes is authoritative
x0206 No valid KeyID items in EncryptingPrefs

5.2.3 Referral

The Referral item (tag x000E) tells the client that another rescap

server has authoritative information for the requested resource. If some information may be available from several sources, an "authoritative" source is one whose response should be regarded as superseding all others in the event of any discrepancy. The structure of the Referral item is a URI string as defined in [URI] for a single resource. The scheme in the URI MUST be "rescap". If a FullResponse item contains a Referral item or a Referral item enclosed in a SignedResponse or an EncryptedResponse item, there MUST NOT be any other items in the FullResponse item other than Status items.

5.2.4 TTLOfInfo

The TTLOfInfo item (tag x0017) specifies how long the server assures that the information enclosed in the item will be valid. The TTLOfInfo item has the structure of a four-octet integer followed by one or more response-type items. The four-octet integer represents the number of seconds before the server no longer assures that the listed items are valid.

If a rescap client does not support expiration of items, it MUST treat the items in a TTLOfInfo item as if they appeared outside of a TTLOfInfo item. This allows a server to respond with TTLOfInfo without knowing whether or not the client handles the expiration time listed in the item.

5.2.5 ExpirationOfInfo

The ExpirationOfInfo item (tag x0018) specifies the final time in the future that the server assures that the information enclosed in the item will be valid. The ExpirationOfInfo item has the structure of a 14-octet integer followed by one or more response-type items. The 14-octet integer represents the date at Greenwich Mean Time as a string of ASCII characters in YYYYMMDDHHMMSS format after which the server no longer assures that the listed items are valid.

If a rescap client does not support expiration of items, it MUST treat the items in a ExpirationOfInfo item as if they appeared outside of a ExpirationOfInfo item. This allows a server to respond with ExpirationOfInfo without knowing whether or not the client handles the expiration time listed in the item.

5.2.6 DateOfChange

The DateOfChange item (tag x001C) specifies the time that the server last changed the value of the attribute (or, if it has never been changed, first created the value). The DateOfChange item has the structure of a 14-octet integer followed by one or more response-type items. The 14-octet integer represents the date at Greenwich Mean Time as a string of ASCII characters in YYYYMMDDHHMMSS format.

If a rescap client does not support showing modification times of

items, it MUST treat the items in a DateOfChange item as if they appeared outside of a DateOfChange item. This allows a server to respond with DateOfChange without knowing whether or not the client handles the display of modification time listed in the item.

5.2.7 PrivUseResponse00 through PrivUseResponse255

The PrivUseResponse00 through PrivUseResponse255 items (tags xFF00 through xFFFF) are reserved for private use as response-type items and will never be assigned by IANA. These tags may be used by protocol developers to test protocols that they are developing, such as during the process of preparing Internet Drafts that contain registration for future response-type items. The structure of the items is undefined.

6. Security-related requests and responses

The security-related requests and responses are optional for both clients and servers.

Note: clients SHOULD be able to parse SignedResponse items even if the client does not know how to verify signatures. These items contain other response items that the client MAY want to deliver to the user regardless of the signature status.

6.1 Security-related request types

6.1.1 AuthInfo

The AuthInfo item (tag x0004) holds authorization items that can be used by the server to choose to give the client more or less information in the response. The AuthInfo item MUST contain a list of one or more request-type items that relate to authorization. The acceptable items that may be contained in the AuthInfo item are AuthInTheClear, AuthDigest, AuthPublicKey, and AuthIP. If more than one item is in the AuthInfo item, the server MAY use any of the items to determine authorization to the information about the resource named in the BaseURI item.

If one or more of the authorizations succeeds, the server SHOULD include a Status item indicating which type of authorization succeeded. If none of the given authorizations succeeds, the server MAY still include information items in the response.

6.1.1.1 AuthInTheClear

The AuthInTheClear item (tag x0005) holds a sequence of binary octets. The format of the sequence is determined by private agreement between the client and the server. This type of authorization is inherently insecure because an eavesdropper can see the sequence and use it get authorization on a later request.

6.1.1.2 AuthDigest

The AuthDigest item (tag x0006) holds a hashed password. TBD: This will probably be done with something similar to the Digest mechanism in draft-ietf-http-authentication-03.txt. However, for this to work, the server must give the client a nonce before the client creates the AuthDigest item; otherwise, the mechanism suffers from a simple replay attack.

6.1.1.3 AuthPublicKey

The AuthPublicKey item (tag x0007) holds a password that has been encrypted with the client's private key. TBD: This will probably be done using a DSS signature with no PKI specified. However, for this to work, the server must give the client a nonce before the client creates the AuthDigest item; otherwise, the mechanism suffers from a simple replay attack.

6.1.1.4 AuthIP

The AuthIP item (tag x0008) is a zero-length item that indicates that the client requests that it gain some authorization simply based on the IP address of the client, which the server will determine by examining the IP address of the connection.

6.1.2 SigningPrefs

The SigningPrefs item (tag x0009) tells the server how the client prefers the server to sign the items in the response. The SigningPrefs item contains a sequence of one or more items; it SHOULD contain a Nonce item, and MAY contain one or more KeyID items. The Nonce item is a nonce that the server SHOULD use if the server returns signed items, and the KeyID items are the key identifiers with which the client would prefer the server to sign. The KeyID items are in decreasing order of preference.

If the FullRequest item includes a SigningPrefs item, the server SHOULD encapsulate as many items as possible in one or more SignedResponse items. If the FullRequest item does not include a SigningPrefs item, the server MAY still include SignedResponse items in the response. If the FullRequest item includes a SigningPrefs item that does not contain any KeyID items that correspond to keys currently usable by the server, the server SHOULD still return information in the response.

If the SigningPrefs item does not include a Nonce item, the server's SignedResponse item will not contain a NonceReply item. Note that the server may chose not to include the NonceReply item if it has pre-signed the objects it is returning. If the SignedResponse item does not contain a NonceReply item, the server's response is still signed, but the reply can be replayed by an attacker at a later time.

6.1.3 EncryptingPrefs

The EncryptingPrefs item (tag x000A) tells the server the encrypting key with which the client would like the response-type items other than Status items encrypted. The structure of the item is a sequence of one or more KeyID items, in decreasing order of preference. If an EncryptingPrefs item is included in the request, the server MUST NOT include anything other than one or more Status items and one or more EncryptedResponse items in the FullResponse item. That is, because the client requested an encrypted response, the server MUST NOT include anything other than a Status item in the FullResponse that is not encrypted.

6.1.4 SignedRequest

The SignedRequest item (tag x000B) gives a digital signature for the other items in the FullRequest. The structure is a single KeyID item followed by a single SignatureValue item. The signature is computed over the contents of the FullRequest item, excluding the entire SignedRequest item.

If the server is able to process a SignedRequest item, it MUST first verify that the KeyID in the SignedRequest is a valid signing key known to be associated with a client. It MUST then use the signature algorithm associated with the KeyID to check the integrity of the content of the SignedRequest. If the integrity check fails, the server SHOULD indicate the failure in the Status item returned in the response, and MAY choose to not include some or all other items in the response.

If the server is not able to process a SignedRequest item, it SHOULD indicate this fact in the Status item in the response. However, other than the status indication, such a server SHOULD treat the FullRequest item as if it did not contain a SignedRequest item.

6.2 Security-related response types

6.2.1 SignedResponse

The SignedResponse item (tag x000F) encloses other response-type items and gives a digital signature for all the items in the SignedResponse item other than the signing control items. The structure of the SignedResponse item is one SignatureControlInfo item followed by one or more response-type items. The digital signature in the SignatureControlInfo is computed over the items other than the SignatureControlInfo item. If the FullRequest item contained a SigningPrefs item, the SignedResponse SHOULD contain a NonceReply item.

If the client can process the signature in the SignedResponse item, it MUST first verify that the KeyID in the SignedResponse is a valid signing key known to be associated with the server or with the resource provider. It MUST then use the signature algorithm associated with the

KeyID to check the integrity of the content of the SignedResponse. If the integrity check fails, the client software SHOULD indicate the failure to the user, and MAY choose to reject the values given in the SignedResponse.

Note that all rescap clients MUST be able to process SignedResponse items even if they cannot process the signatures. If a client cannot process signatures, it MUST treat the items enclosed in the SignedResponse as if they were outside a SignedResponse but inside the FullResponse item. Requiring clients to handle SignedResponse items in this fashion allows a server to create pre-signed items and groups of items and serve them without first checking if the client can handle a particular signature algorithm. However, a server SHOULD still check the SigningPrefs item of the FullRequest item to assure that it is returning signatures that the client can use.

6.2.1.1 SignatureControlInfo

The SignatureControlInfo item (tag x0010) gives the key and digital signature used to sign the items in the SignedResponse item. The structure of the SignatureControlInfo item is an optional single KeyID item followed by the value of the digital signature. The algorithm used to calculate the digital signature depends on the KeyID. See the DefaultKeyID item for information about handling SignatureControlInfo items that have no KeyID item.

6.2.1.2 DefaultKeyID

The DefaultKeyID item (tag x0011) contains the KeyID that is used by default in every SignatureControlInfo item that does not contain a KeyID item. The structure of the DefaultKeyID item is a single KeyID item. A DefaultKeyID item MUST only appear at the top level of a FullResponse item or the top level of an EncryptedResponse item; in either case, it MUST only appear zero or one time. A DefaultKeyID item MUST NOT appear at any level of a SignedResponse item, a TTLOfInfo item, or an ExpirationOfInfo item.

6.2.1.3 NonceReply

The NonceReply item (tag x0012) contains a single item, the Nonce item that was specified in the SigningPrefs item in the request.

6.2.2 EncryptedResponse

The EncryptedResponse item (tag x0013) carries information that has been encrypted with the client's encryption key. The structure of the EncryptedResponse item is a sequence of one EncryptionControlInfo item and one EncryptedBlob item. The EncryptedBlob item contains the encryption of one or more response-type items.

If a FullResponse item contains an EncryptedResponse item, the FullResponse MUST only contain EncryptedResponse items and Status

items. A FullResponse item MAY contain more than one EncryptedResponse item, although doing so is of limited value and causes greater processing overhead for the client.

The server MUST ONLY prepare an EncryptedResponse item using a KeyID that is associated with a known client. Otherwise, an attacker can cause the server to think that it is sending protected information on the Internet when in fact it can be decrypted by the attacker.

6.2.2.1 EncryptionControlInfo

The EncryptionControlInfo item (tag x0014) contains the encrypted session key used for encrypting the EncryptedBlob item. The structure of the EncryptionControlInfo is a sequence of the KeyID item of the key that was used to create the key encryption key, and the KeyEncryptionKey item that holds the key encryption key.

6.2.2.2 EncryptedBlob

The EncryptedBlob item (tag x0015) contains binary data that has been encrypted. When decrypted, the contents of the EncryptedBlob item MUST be a sequence of response-type items.

6.2.2.3 KeyEncryptionKey

The KeyEncryptionKey item (tag x0016) contains binary data that is a key encryption key. The structure of the key encryption key is determined by the algorithm used, which is specified in the KeyID item.

7. Data-type items

7.1 Nonce

The Nonce item (tag x0019) contains a random or pseudo-random list of octets. The length of the Nonce item is determined by the process using the nonce, but MUST NOT be less than eight octets.

7.2 KeyID

The KeyID item (tag x001A) holds an identifier for an encrypting or signing key. The contents of the KeyID item is always an 8-octet identifier. The identifier is derived by first concatenating an algorithm identifier to the end of the key, then taking the SHA-1 [SHA-1] hash of the result, then taking the first eight octets of the hash.

7.3 SignatureValue

The SignatureValue item (tag x001B) holds the value of the signature. The contents of the SignatureValue item is a binary value whose length is determined by the signature algorithm used.

8. Security Algorithms

TBD. The spec will require the use of DSA for signatures, Diffie-Hellman for encryption key exchange, and TripleDES for encryption. It will suggest also implementing RSA for both signatures and encryption key exchange. It may or may not suggest support for shared-secret keys for encryption key exchange.

9. Security Considerations

The current draft doesn't specify how the security algorithms will be used, and thus doesn't give enough detail to analyze the security considerations of the specification.

A server may choose to allow authorization through a variety of mechanisms, some of which have better security properties than others. Specifically, the AuthInTheClear mechanism passes authorization in the clear; this allows an attacker to copy the authorization information and impersonate the client in the future.

All public keys in this specification are trusted based on private agreement. The client and server can use out-of-band mechanisms to agree to public key management, but none of the mechanisms are described here. A successful attack on the out-of-band mechanism could allow the attacker to impersonate the client or the server, or could allow the attacker to read encrypted responses.

10. References

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

[RESCAP-REQUIRE] "ResCap Requirements", draft-beck-rescap-req.

[SHA-1] "Secure Hash Standard", NIST FIPS publication 180-1, April 1995.

[SLP] "Service Location Protocol", RFC 2165. NOTE: RFC 2165 is being updated by draft-ietf-svrloc-protocol-v2-xx.txt.

[SRV] "A DNS RR for specifying the location of services (DNS SRV)", RFC 2052. **NOTE: RFC 2052 is being updated by** draft-ietf-dnsind-rfc2052bis-xx.txt. The examples in this document are based on the Internet Draft, not on RFC 2052.

[URI] "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396.

[UTF8] "UTF-8, a transformation format of ISO 10646", RFC 2279.

A. IANA Considerations

A.1 Item registry

IANA will maintain a registry of all rescap items. The registry will be populated by the items in this specification and any other RFC. The RFCs must give sufficient detail so that interoperability between independent implementations is possible. The registry will contain the name of the item, the tag value, the type, and the RFC which defines the item. Each name in the registry must be unique, and each tag value must also be unique. The types in the registry are "request-type", "response-type", and "data-type". IANA should also list the RFC in which the items are defined so that protocol designers know where to find the document defining the items.

Each RFC that defines rescap items must include an application of the form:

To: iana@iana.org
From: <author's address>
Subject: Registration of rescap items

The following are the rescap items defined in this document.

Item Type Tag

A.2 Status value registry

IANA will maintain a registry of all rescap status values. The registry will be populated by the status values in this specification and any other RFC. The RFCs must give sufficient detail so that interoperability between independent implementations is possible. The registry will contain the value of the status value, a brief description of the meaning of the value, and the RFC which defines the value. Each value in the registry must be unique. IANA should also list the RFC in which the status values are defined so that protocol designers know where to find the document defining the status values.

Each RFC that defines rescap status values must include an application of the form:

To: iana@iana.org
From: <author's address>
Subject: Registration of rescap status values

The following are the rescap status values defined in this document.

Value Meaning

B. Registration of rescap Types, Status Values, and URL Scheme

B.1 Registration of rescap Types

To: iana@iana.org
From: Paul Hoffman <phoffman@imc.org>
Subject: Registration of rescap items
The following are the rescap items defined in this document.

Item	Type	Tag
FullRequest	request	x0001
BaseURI	request	x0002
ItemsToReturn	request	x0003
AuthInfo	request	x0004
AuthInTheClear	request	x0005
AuthDigest	request	x0006
AuthPublicKey	request	x0007
AuthIP	request	x0008
SigningPrefs	request	x0009
EncryptingPrefs	request	x000A
SignedRequest	request	x000B
FullResponse	response	x000C
Status	response	x000D
Referral	response	x000E
SignedResponse	response	x000F
SignatureControlInfo	response	x0010
DefaultKeyID	response	x0011
NonceReply	response	x0012
EncryptedResponse	response	x0013
EncryptionControlInfo	response	x0014
EncryptedBlob	response	x0015
KeyEncryptionKey	response	x0016
TTLOfInfo	response	x0017
ExpirationOfInfo	response	x0018
Nonce	data	x0019
KeyID	data	x001A
SignatureValue	data	x001B
DateOfChange	response	x001C
PrivUseRequest00 through PrivUseRequest255	request	xFE00-xFEFF
PrivUseResponse00 through PrivUseResponse255	response	xFF00-xFFFF

B.2 Registration of rescap Status Values

To: iana@iana.org
From: Paul Hoffman <phoffman@imc.org>
Subject: Registration of rescap status values

The following are the rescap status values defined in this document.

```
Value Meaning
x0000 The request was fully processable
x0001 Successful authorization through AuthInTheClear
x0002 Successful authorization through AuthDigest
x0003 Successful authorization through AuthPublicKey
x0004 Successful authorization through AuthIP
x0005 No valid KeyID items in the the SigningPrefs
x0006 No valid KeyID items in the SignedRequest
x0007 The content of the SignedRequest didn't validate against the
      signature

x0100 Too busy; try again whenever you feel like it
x0101 Too busy; try again later than 10 seconds from now
x0102 Too busy; try again later than 60 seconds from now
x0103 Too busy; the Referral item in the response leads to another
      rescap server authoritative for this resource

x0200 The body of the request was longer than what was indicated in the
      length
x0201 The body of the request was shorter than what was indicated in
          the length
x0202 The request included items not of request-type
x0203 The request included more than one BaseURI item
x0204 This server is not authoritative for the resource named in the
      BaseURI item and no referral is available
x0205 This server is not authoritative for the resource named in the
      BaseURI item; the URI in the Referral item leads you to a server
      that this server believes is authoritative
x0206 No valid KeyID items in EncryptingPrefs
```

B.3 Registration for rescap URL Scheme

URL scheme name: rescap

URL scheme syntax: <scheme-name>://<hostport>?<base64-of-request>
<scheme-name> is the string "rescap". <hostport> is exactly as defined in RFC 2396. <base64-of-request> is the Base64 encoding of a rescap request

For example, to indicate a query to the rescap server on the host "an.example.com" at the default port of 283, the URL would be rescap:an.example.com/SK398cske002CcksleEEx

For example, to indicate a query to the rescap server on the host at 10.20.30.40 at port 1234, the URL would be rescap:10.20.30.40:1234/SK398cske002CcksleEEx

Character encoding considerations: None. All three parts are expressed in US-ASCII.

Intended usage: To identify queries rescap servers. The resolution of a
rescap URL will normally cause a query to be sent to the named
server. The resolver would decode the Base64 of the third part of
the URL and send it to the specified port (or the default port of
283 if no port is specified in the URL) using the TCP protocol.

Applications and/or protocols which use this URL scheme name:
This document describes the rescap protocol.

Interoperability considerations: None known.

Security considerations: Same as in this document.

Relevant publications: This document.

Person & email address to contact for further information:
Paul Hoffman <phoffman@imc.org>

Author/Change controller:
Paul Hoffman <phoffman@imc.org>

C. Acknowledgments

Graham Klyne provided all the suggestions for changes to the first draft.

D. Changes Between Versions of This Document

D.1 Changes from -00 to -01

Changed the title of the document. Also slightly reworded the abstract.

3: Second paragraph, added "for short exchanges". Last paragraph,
changed the "MAY choose to not go to TCP" to "MUST get a TCP response"
because some extensions to rescap may have later parts of a response
modifying earlier parts.

4: Added the bit about network byte order for each part of an item.

4.1: Changed the entire "long content" design.

5.: Added the last sentence about the rationale for not returning
requested items.

5.2: Changed the last sentence in the second paragraph to get rid of
"display".

5.2.3: Added the defintion of authoritative.

6.1.1.4: Changed "empty" to "zero-length".

6.2.1: Added " or with the resource provider" in the second paragraph.

7.2: Added the reference to SHA-1.

10: Added [SHA-1].

B: Renumbered the section.

B.3: Changed "<host>" to "<hostport>". More importantly, changed the scheme from "rescap:host.example.com/<base64>" to "rescap://host.example.com?<base64>".

E. Author Contact Information

Paul Hoffman
Internet Mail Consortium
127 Segre Place
Santa Cruz, CA 95060 USA
phoffman@imc.org