OPEN PEER – PROTOCOL SPECIFICATION

Contents

Abstract

The holy grail of communication on the Internet has been to allow peer-to-peer communication without the requirement of any centralized servers or services. A peer-to-peer approach offers some key advantages over a centralized server approach:

1) Greater network resilience – peers can continue to function independent of servers and can operate even if servers are down
2) Increased privacy and security – peers communicate directly thus the data is not centralized in one location where it can be spied upon by corporations, governments, 3rd parties or hackers.
3) Decreased cost – without the need of servers, the cost to host, administer, store and relay data is reduced substantially
4) Scalability – a peer to peer network doesn't require servers to scale as the peers can operate amongst themselves

Unfortunately, the goal of peer-to-peer and the reality of peer-to-peer do not match. Centralization of data into the Internet cloud is prolific and firewalls frequently impede direct peer-to-peer communication making peer-to-peer connection extremely difficult to setup and challenging to architect.

What further reduces the proliferation of peer-to-peer is a lack of standardization, openness and ubiquity of the technology. The standards bodies have been working for years on firewall traversal techniques and standardization of the approaches and a new joint effort called WebRTC between the W3C and IETF on how browsers can directly communication between browsers to move media. This joint effort does not specify how signalling happens between peers so it's not a complete solutions on its own.

Performing peer-to-peer approach to signalling has been notoriously difficult for a variety of reasons:

1) Without a publicly addressable intermediate 'server' machine to initiate communication, two peers behind firewalls are never able to communicate with each other. Thus, a peer network almost always requires some sort of rendezvous and relay servers to initiate contact between peers behind firewalls (and firewalls tend to be used more frequently than not for end users).

2) Automatically promoting the few publically addressable home machines into rendezvous and relay servers is not the best option. Average users tend to not want to have their home/work machines to be automatically promoted to rendezvous and relay servers since it consumes their bandwidth and costs them to relay traffic for others who "leech" off their bandwidth. This cost factor causes end users to intentionally shutdown protocols that promote end user machines into servers. Over time, the number of average users willing to have their machines operate as servers for the benefit of those leeching decreases relative to the number of those whom leech off those servers until the entire system collapses with a too great server/leech ratio. As an example, Skype's network collapsed for this very reason and they were forced to setup their own super nodes to handle the load.

3) Some peer-to-peer networks require ports to be opened on a firewall to operate. Where possible, peers will register themselves with UPnP to open the ports when the firewall automatically. Unfortunately, many firewalls lack the ability to automatically open ports or actively disallow this feature for fear that this opens the network to security holes. If opening ports automatically is not possible then users are required to open ports manually. Thus only the technically savvy can perform this task and such peer networks tend to be limited to those who are technically savvy. This is not a universal solution since it assumes too much technical ability and responsibility of the end user.

4) Many peer networks rely on mutual peers not behaving in an evil manner. Peers that do not act in an altruistic fashion can easily disrupt these networks. When all peers behave properly there is no problem with such a network; however, the moment an 'evil' node or cluster of 'evil' nodes is injected into the peer network, parts or all of the network can suffer fatal issues and security can be compromised.

Open Peer is peer-to-peer signalling protocol taking advantages of the IETF advances of firewall penetration techniques for moving media and adds a layer to performs the media signalling in a peer-to-peer fashion but does expect that a minimal requirement of rendezvous servers existing. Beyond the initial rendezvous to get past firewalls, the servers should drop out of the protocol flow and are no longer required.

Open Peer was designed with these main goals in mind:

1) Openness – a protocol is freely available for anyone to implement.

2) Greater network resilience – peers can continue to function and interoperate even if servers are down.

3) Increased privacy and security – peers communicate directly in a secure fashion designed to protect against network ease dropping, forged communication or spying by 3rd parties or being a convenient data mining target for hackers as the information does not flow through centralized servers.

4) Federation – the protocol makes it easy for users on one service to communicate with users on another independent service offering.

5) Identity protection – the ability of users to easily provide proof of their identity using existing social platforms while protecting these identities from spoofed by others.

6) Decreased cost – without the need to continuously relay signalling or media through centralized servers, the costs to host, administer, relay, replicate, process and store data on servers while providing 5 9s uptime is decreased.

7) webRTC enabling protocol – designed to be the engine that allows webRTC to function, supporting federation of independent websites and services, provide security and online identity protection and validation, and peer-to-peer signalling bypassing the need for heavy cloud based infrastructure.

8) Scalability – whether starting at 50 users or moving beyond 5,00,000 users, the protocol is designed to allow for easy scalability by removing the complexity of communications out of the servers.

Design Considerations

The Open Peer Protocol has several design considerations to address the realities of the Internet infrastructure while delivering the functionality required:

- Must allow any peer to connect to any other peer (if authorized).
- Must understand firewall principles and to offer an architecture which factors that firewalls are prevalent and within the natural scope of the architecture's basic design.
- Must accept that it's not always desirable to have peer machines automatically promoted to rendezvous servers.
- Must allow additional services to be layered onto of the architecture
- Must enable peers to find each other using directory services.
- Must enable secure peer-to-peer communication without penetration or monitoring by third parties.
- Must allow peers to perform identity validations.
- Must allow anonymous peers, i.e. similar to unlisted and non-guessable phone numbers.
- Must allow for differing server rendezvous architectures, i.e. anywhere from peer-to-peer self-organized models to centralized network layouts are to be abstracted from the protocol.
- Must not require end user signed certificates from a known authority chain for each peers on the network to establish secure communications.
- Must not require end users or administrators to configure firewalls or open ports under normal circumstances.

Key Object Concepts

Identity

An Identity is the persona of a peer contact, be they the representation of a real person or representative entity (much like a corporation is a legal entity but not a real person). An Identity maps to a single Peer Contact although a Peer Contact can have multiple Identities.

Asserted Identity

An Asserted Identity is an identity that can be verified through an identity service as being the legal owner of the persona rather than a fraudulent representation. In other words, a validated asserted identity can be trusted that they are whom they claim to be. Different levels of identity assertion can be claimed for any given identity starting with no provable assertion at all and moving anywhere from weak to strong verification depending on the identity validation service types available.

Identity Lookup Server

A server that looks up and returns the Peer Contact associated with an Identity or a set of Identities and can return the public profile information for Peer Contacts.

Identity Signing Service

A service that provides the Asserted Identities for the various personas that are owned within a particular service offering.

Identity Provider

Any service offering that grants Identity personas, such as Facebook, LinkedIn, Twitter or other 3rd parties that offer their own Identities.

Peer Contact

A Peer Contact is the representation of a single point of contact on the Internet regardless of the personas represented by the peer contact. A peer contact can exist at zero or more Peer Locations at any given time.

Peer

A Peer is the single instance of a peer client application on the Internet, which registers a single Peer Contact in the Peer Domain at a particular Peer Location.

Peer Location

A Peer Location is the representation of where a peer is located. A Peer can only exist at a single location but the Peer Contact for the Peer can register at multiple Peer Locations.

Peer URI

A Universal Resource Identifier (URI) starting with "peer:" offering the ability to locate a specific peer resource, protocol and request type within a peer domain.

Peer Domain
A Peer is always connected to a Peer Domain and the domain is the organization responsible for managing the connected peers.

Peer Finder
A Peer Finder is a rendezvous server that keeps track of connected peers at their peer locations since they are connected in a dispersed fashion through a peer domain. A peer finder will utilize a database (typically distributed) to facilitation the introduction of peer communication on the same domain or across domains.

Bootstrapper
A Bootstrapper is the introductory server where peers first go to be introduced to one (or more) Peer Finders. Peers should attempt to connect to introduced Peer Finders in order to gain entry to the Peer Domain. Once a Peer is connected to a Bootstrapped Network, the Peer should no longer require communication back to the Bootstrapper unless access to previously introduced Peer Finders are no longer accessible.

Bootstrapped Network
A Bootstrapped Network is the representation of the entire peering network that was introduced from a Bootstrapper.

Public Peer File
A file that contains a cryptography public key for secure conversations, information required to locate the Peer Contact within a Peer Domain, information to authorize a connection to that Peer by another Peer and public Identities associated to a peer. Any Peer without the correct Public Peer File for another Peer Contact will be unable to connect to that peer. A directory service can host and offer these Public Peer Files between peers but without this file no communication is possible between peers (thus allowing for "unlisted" peers).

Private Peer File
A file that contains a private key to be the pair of a public key inside the Public Peer file that is used by a Peer to be used to establish secure communications between Peers. The Private Peer File is encrypted and can only be decoded with the correct key.

Peer Pair
A file pairing consisting of both a Public Peer File and a Private Peer File.

Provisioning Service
A service that provides account creation and account profile maintenance.

Peer Service
Any additional services offered to peers are done through what is called a Peer Service. Examples of such services are those that perform identity assertion, TURN or future services like video conferencing mixers.

The "peer:" URI scheme

Syntax:

peer://<domain>/contact-id

[/<resource>][?<query>][;protocol=<protocol>][;request=<request>][#<fragment>]

<domain> - the domain service where the Bootstrapped Network is introduced, e.g. "foo.com" or "bar.com".

<contact-id> - the hash result of the section A of the Public Peer File

Examples:

peer://foo.com/e852191079ea08b654ccf4c2f38a162e3e84ee04
peer://example.org/3b0056498dc7cdd6a4d5373ac0860f9738b071da
peer://<domain>/contact-id

Syntax (future extensions):

peer://foo.com/id[/<resource>][?<query>][#<fragment>][;protocol=<protocol>][;request=<request>]

<resource> - and optional resource within the peer that is being requested.

<query> - an optional component which identifies non-hierarchical information about a resource.

<protocol> - the default value of "peer-dialog" is presumed, other extensions like "peer-http" are possible and might be presumed depending on the "lookup-type" requested

<request> - this allows control over the action required which this URI is requested, e.g. "call" might be used to establish an audio/video call to the peer or "get" might be used (or even assumed) in combination with a protocol type of "peer-http" to indicate performing an HTTP request over Open Peer.

<fragment> - an optional component which identifies direction towards a secondary resource.

Example future extensions:

peer://example.org/3b0056498dc7cdd6a4d5373ac0860f9738b071da/index.php;protocol=peer-http
peer://hookflash.com/3b00564d6a4d5373ac0860f9738b071da/;protocol=peer-http;request=get
peer://foo.com/e852191079ea08b654ccf4c2f38a162e3e84ee04;request=call

The "identity:" URI scheme

Syntax:

identity:[type:][//<domain>/]<identity-string>

If a "//" is not present after the "identity:" scheme, then the identity is assumed to be a specialized registered type that must be resolved in a specialized manner. If the "//" is present then the identity is resolved by the specified domain, with an optional username/password access credentials to the identity lookup service.

<username> - the username used for basic auth to the identity lookup service

<password> - the password used for basic auth to the identity lookup service

<domain> - the domain service where the identity is resolved, e.g. "foo.com" or "bar.com".

<identity-string> - the string that represents the persona of the identity but the interpretation is specific to a particular domain.

The URL characters '?' ';' '#' are reserved characters in the URI scheme and should never be used within an identity.

Examples:

identity:phone:14165551212
identity:email:foo@bar.com
identity://foo.com/alice.78
identity://facebook.com/id3993232
identity://bar.com/linkedin.com/zs39923yf

The Makeup of the Public Peer File

A Public Peer File contains information required to locate, connect and establish a secure channel between peers and contains Identities within the same file. The Public Peer File is made up of three sections, appropriately named Section "A", Section "B" and Section "C". Section "A" can be safely transmitted to any third party allowing any third party to know the Peer Contact's "contact ID" as well as its pubic key to prove ownership (only the owner would have the related private key). Section "B" allows for another peer to find the peer and initiate contact. Section "B" can be withheld from any other peer if the peer does not wish itself to be found by that peer. Section "C" is used to include proven identities of the contact. This section can be given or withheld depending how anonymous the peer wishes to be. The entire file is only given to third parties (i.e. Section A+B+C) allowed to communicate directly to the peer containing where the peer wants to be contacted and wants to expose its identities. At minimal, Section "A" is required to establish a secure channel between peers and Section "B" is required to allow a peer to be found by another peer and Section "C" is required to prove public identities of the peer.

A Public Peer File should contain the extension of ".peer"

Section "A" (packaged and signed by identity's private key)

- Cipher suite to use for all hash computations and encryptions related to contents of the peer file (note: this does not apply to signed JSON segments which have their own method to describe algorithms and key selection)
- Public peer file creation date
- Public peer file expiry date
- Salt signed by salt service's key
- Extension authorized-peer data
- Peer's public key (in signature) – The key is base 64 encoded version of "SubjectPublicKeyInfo" PKCS #1/X.509 DER encoding (BER decoding) format. This x.509 format is used for all x.509 keys in the system.

Section "B" (packaged and signed by identity's private key)

- Peer Contact's full URI (to know how to locate the peer universally) that do not reveal any identities. Peer's contact ID is calculated as follows: hash value of section "A", i.e. tolower(hexEncode(hash("contact:" + <public-peer-section-A>))). When the input <public-peer-file-section-A is used in the hash, the same canonical algorithm method as the signature in section "A". The input into the algorithm is the entire section "A" bundle including the certificate signing the section bundle.
- Find secret (must be known by peer attempting to initiate a finder connection to another peer). This is a simple random string and is not encoded into base-64.
- Extension authorized-peer data

Section "C" (packaged and signed by identity's private key)

- Peer Contact's full URI (to know how to locate the peer universally).

- Any/all asserted public identities
- Extension authorized-peer data

The public key is used as a way to send the peer privately encrypted data from another source. As long as the other source has the correct public key it is possible to establish direct secure communication by exchanging keys using public/private keys as the encryptions method.

The salt is used in Section "A" to establish randomness into the files that is not forgeable or controllable by the creator of the file this ensuring that hashes are dispersed based on randomness being present.

Asserted identities are used to prove the owner of the peer file is whom they claim to be.

Extension data is arbitrary for future extension of the peer file.

The peer's contact ID is used to prove that Section "B" and Section "C" correlates properly to section "A" and isn't two or three distinct files being glued together as a forgery.

Security Considerations

The Section "A" bundle must contain salt that has been signed by the salt service whose certificate is still within the window of validity. Further the Section "A" bundle must be signed by a self-signed certificate whose certificate is included in the signature of the bundle. This ensures the integrity of the Section "A" bundle and ensures anyone who has Section "A" knows the public key for the peer.

Section "B" includes a finder secret that other peers will use to find the peer to which the peer file belongs.

Asserted Identities contained within section "C" can be verified whenever verification is needed. Verification is dependent on the identity assertion type.

The integrity of Section "B" and Section "C" should be verified by ensuring that the public key contained in Section "A" signed these sections. The URIs in these sections are not verified as it's up to the client generating the URIs to generate resources that are accurate to the network and up to other peers to ensure they are contacting identities they should be contacting.

Only elements contained within the signed sections are ever considered as part of the file. All other elements are erroneous and should be discarded or ignored and when present. In Section "A", erroneous elements outside the protection of a signature should never be used as part of the calculation of the contact ID.

Example Public Peer File:

```
{
  "peer": {
    "$version": "1",
    "sectionBundle": [
      {
        "section": {
          "$id": "A",
          "cipher": "sha256/aes256",
          "created": 54593943,
```

```
          "expires": 65439343,
          "saltBundle": {
            "salt": {
              "$id": "cf9c4688b014e13d8bdd2655912ffd3253f53768",
              "#text": "Z3nfnDenen29291mfde…21n"
            },
            "signature": {
              "reference": "#cf9c4688b014e13d8bdd2655912ffd3253f53768",
              "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
              "digestValue": "ODMxNmI3Mjd...MzIzYzk5Nzc0ZGY5MQ==",
              "digestSigned": "DEfGM~C...0/Ez=",
              "key": {
                "$id": "db144bb314f8e018303bba7d52e",
                "domain": "example.org",
                "service": "salt"
              }
            }
          }
        },
        "signature": {
          "reference": "#A",
          "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
          "digestValue": "OzIyMmI3NG...WJlOTY4NmJiOWQwYzkwZGYwMTI=",
          "digestSigned": "G4Fwe...0E/YT=",
          "key": { "x509Data": "MIID5jCCA0+gA...lVN" }
        }
      },

      {
        "section": {
          "$id": "B",
          "contact": "peer://example.com/ab43bd44390dabc329192a392bef1",
          "findSecret": "YjAwOWE2YmU4OWNlOTdkY2QxNzY1NDA5MGYy"
        },
        "signature": {
          "reference": "#B",
          "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
          "digestValue": "MWU4ODQwM2ZlOTQ...IzNDAyZTE0OWZkYg==",
          "digestSigned": "MC0...E~LE=",
          "key": { "uri": "peer://example.com/ab43bd44390dabc329192a392bef1" }
        }
      },

      {
        "section": {
          "$id": "C",
          "contact": "peer://example.com/ab43bd44390dabc329192a392bef1",
          "identities": {
            "identityBundle": [
              {
                "identity": {
                  "$id": "b5dfaf2d00ca5ef3ed1a2aa7ec23c2db",
                  "contact": "peer://example.com/ab43bd44390dabc329192a392bef1",
                  "uri": "identity://facebook.com/id48483",
                  "created": 54593943,
                  "expires": 65439343
                },
                "signature": {
                  "reference": "#b5dfaf2d00ca5ef3ed1a2aa7ec23c2db",
                  "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
                  "digestValue": "IUe324koV5/A8Q38Gj45i4jddX=",
                  "digestSigned": "MDAwMDAwMGJ5dGVzLiBQbGVhc2UsIGQ=",
                  "key": {
                    "$id": "b7ef37...4a0d58628d3",
                    "domain": "hookflash.org",
                    "service": "identity"
                  }
                }
              },
              {
                "identity": {
```

```
                  "$id": "0a9b2290343734118469e36d88276ffa6277d196",
                  "contact": "peer://example.com/ab43bd44390dabc329192a392bef1",
                  "uri": "identity://twitter.com/booyah",
                  "created": 54593943,
                  "expires": 65439343
                },
                "signature": {
                  "reference": "#0a9b2290343734118469e36d88276ffa6277d196",
                  "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
                  "digestValue": "IUe324koV5/A8Q38Gj45i4jddX=",
                  "digestSigned": "MDAwMDAwMGJ5dGVzLiBQbGVhc2UsIGQ=",
                  "key": {
                    "$id": "cb231aa9a9...eaf43f",
                    "domain": "twitter.com",
                    "service": "identity"
                  }
                }
              }
            ]
          }
        },
        "signature": {
          "reference": "#C",
          "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
          "digestValue": "UrXLDLBIta6sk...oV5/A8Q38GEw44=",
          "digestSigned": "MC0...E~LE=",
          "key": { "uri": "peer://example.com/ab43bd44390dabc329192a392bef1" }
        }
      }
    ]
  }
}
```

The Makeup of the Private Peer File

The Private Peer File is never given out to any other peer. However, the peer file can be stored with a trusted service as the contents are encrypted. The contents of the file must be encrypted to prevent unauthorized access to the private key that is the matching pair for the public key in the Public Peer File. This file can be used to prove ownership of the Public Peer File.

The file does not carry any recommended extension and is managed by the client application that must maintain the security and integrity of the file.

The contents of the file are as follows:

Section "A"

- Cipher suite to use for all hash computations and encryptions related to contents of the private peer file and this cipher suite must match the public peer file's cipher suite (note: this does not apply to signed JSON segments which have their own method to describe algorithms and key selection)
- Peer Contact's full URI (to know how to locate the peer universally).
- Salt (base-64 encoded binary salt)
- 'Private Peer File secret' proof, hash = hmac(<private-peer-file-secret>, "proof:" + <peer-contact-id>)

Section "B" (encrypted using the method described in Section A)

- Encrypted Peer Contact's full URI (to know how to locate the peer universally), key=hmac(<private-peer-file-secret>, "contact:" + base64(<salt>)), iv=hash("contact:" + base64(<salt>))
- Encrypted private key - the key is stored in "PrivateKeyInfo" unencrypted RSA encoding using PKCS #8 but then encrypted and base 64 encoded using: key = hmac(<private-peer-file-secret>, "privatekey:" + base64(<salt>)), iv=hash("privatekey:" + base64(<salt>))
- Encrypted Public Peer File, key = hmac(<private-peer-file-secret>, "peer:" + base64(<salt>)), iv=hash("peer:" + base64(<salt>))
- Encrypted private data, key = hmac(<private-peer-file-secret>, "data:" + <salt>), iv= hash("data:" + base64(<salt>))

The format of the Private Peer File is defined so it can be stored on server (should a client desire to do so) with only clients that have the correct "private peer file secret" being able to request download of the file without the server knowing the value of the data contained within the file.

The Peer Contact's URI is used to indicate which Public Peer File the Private Peer File is correlated.

The key salts combined with hash input phrases are used to ensure that the "private peer file secret" is not directly used to encrypt more than one piece of data.

The "private peer file secret" proof is used so a server can verify a client does have the correct information to request download of the Private Peer File. Only a client that knows the "private peer file secret" would be able to generate the correct key proof in a challenge. The contact ID is combined with the secret to add extra complexity into the secret to ensure no two users have the same stored secret resulting hash should the private peer file is published into a database and two users use the same secret value.

The encrypted private key is the private key pair matching the public key in the Public Peer File.

The encrypted Public Peer File is a complete encryption of the Public Peer File (i.e. all sections), thus requiring only one file to store both the public and private key.

The encrypted private data is extension data for use for whatever purposes required by a client.

Security Considerations

The contact URI must be the computed hash based on Section "A" of the Public Peer File. The salt must be cryptographically random. Both sections of the private peer file must be signed to ensure the contents of the private peer file have not been modified by another entity and must be verified by the client before the private peer file is used.

All data in this file is considered secure thus all data must be encrypted.

The "private peer file secret" should be a cryptographically random string that is sufficiently long to protect the sensitive contents it encodes and should not derived from a user's password.

Example Private Peer File

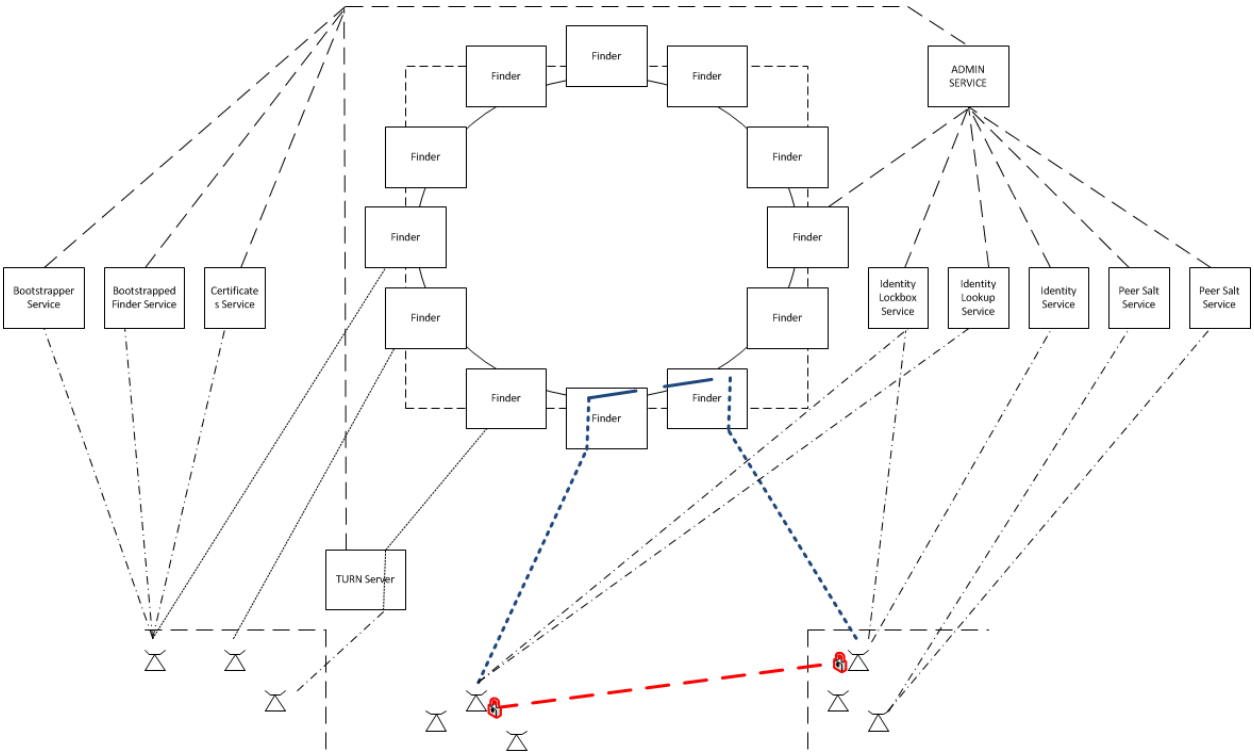
```
{
  "privatePeer": {
    "$version": "1",
    "sectionBundle": [
      {
        "section": {
          "$id": "A",
          "contact": "peer://example.com/ab43bd44390dabc329192a392bef1",
          "cipher": "sha256/aes256",
          "salt": "YzY4NWUxMGU4M2ZjNzVkZWQzZTljYWMyNzUzZDAwNGM4NzE5Yjg1",
          "secretProof": "NDlkZWI0MzFhYmUxOWQzNWJkNDkzMWZhMzFmMw=="
        },
        "signature": {
          "reference": "#A",
          "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
          "digestValue": "j6lw...x3rvEPO0vKtMup4NbeVu8nk=",
          "digestSigned": "G4Fwe...0E/YT=",
          "key": { "uri": "peer://example.com/ab43bd44390dabc329192a392bef1" }
        }
      },

      {
        "section": {
          "$id": "B",
          "encryptedContact": "cGVlcjo...NDNiZDQ0MzkwZGFiYzMyOTE5MmEzOTJiZWYx",
          "encryptedPrivateKey": "jk483n2n~3232n/34nk323j...32fsjdneen2311=",
          "encryptedPeer": "43j2332944bfdss323bjfjweke2dewbub3i...22dnnewne321~nn32n3j2/44=",
          "encryptedPrivateData": "ZGM0MzQxODBjMTgxMDY2NGQ4MWE...GUwYjMzNmI4Nzk5OWU="
        },
        "signature": {
          "reference": "#B",
```



```
          "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
          "digestValue": "UrXLDLB...Ita6skoV5/A8Q38GEw44=",
          "digestSigned": "MC0...E~LE=",
          "key": { "uri": "peer://example.com/ab43bd44390dabc329192a392bef1" }
        }
      }
    ]
  }
}
```

Overall Network Architecture

Network Diagram



Network Components

Bootstrapper – a server that acts as an introductory server to the entire peer network. The Bootstrapper exclusively talks over HTTPS to clients which must have a certificate issued from a trusted certificate authority.

Salt Service – a server that generates cryptographically strong salt and signs the salt as having come from the server. This service is useful to ensure cryptographically strong random data has been correctly used whenever it is critically important and especially when a server can't trust a client will generate random data that is intentionally or accidently non-cryptographically random. The salt service exclusively talks HTTPS to clients and the certificate authority as discovered from the Bootstrapper service will sign the certificate.

Identity Service(s) – servers that provide Identity Lookup or Asserted Identities for Identities such as LinkedIn, Facebook or other 3rd party Identities. The weak form of an Asserted Identity allows a third party like Hookflash Inc. to assert an Identity is correct when the Identity Provider does not offer an Identity signing service themselves.

TURN server – a server which is used to relay information on an 'as needed' basis when the firewall is of a type where relaying is required to penetrate the firewall. The TURN service will talk the standard TURN protocol.

Finder – a server that keeps information about each Peer Contact's Peer Locations and assists Peers in establishing the initial communication between Peers. The finder may allows a few requests over HTTPS but most request must be directed over Message/RUDP/UDP or Message/SCP protocol. The requests allowed over HTTPS are specifically mentioned with each allowed request.

Conference service – this is an example service that could be utilized as a relay point when communicating between peers in a conference scenario that would typically overwhelm a standalone client's CPU or bandwidth, but no protocol has been yet defined for Open Peer.

Peer – a peer is a client that uses the various services in the architecture to help with the establishment of communication to other Peers. Once peers establish and communicate directly with other peers, they should not required to utilize server infrastructure to maintain the communication (with the exception possibly of using a TURN server). Peers use the Message/RUDP/UDP or Message/SCP protocol as their initial connection and control protocol.

Limitations to Scope of Open Peer Specification

Open Peer defines the communication between the client and the Open Peer services but does not delegate how the servers in the infrastructure communicate amongst each other. Open Peer does not define how peer finders are allocated amongst peers. Open Peer allows Peers to treat servers as potentially untrustworthy from a privacy perspective and thus the protocol was designed to keep sensitive information from flowing through the servers or the servers having access to the keys to decrypt the sensitive data. Open Peer does not dictate how the data contained within and between servers be secured, only that the data must be secure.

RUDP Protocol

Overall Design Goals

RUDP was designed to allow bi-direction FIFO (First-In-First-Out) congestion controlled streamed data between two peers that is modelled after TCP, except that it is highly friendly to firewalls and utilizes firewall friendly protocols and techniques to connect between peers. The RUDP can be used between peers or from server to server.

TCP is a great protocol for signalling as it is reliable and messages are always delivered in order to a report party in a FIFO manner. The major problem with TCP is that it is not firewall friendly for peer-to-peer scenarios. TCP works great for peer-to-server where one entity is not located behind a firewall (i.e. the server). However, TCP between peers using today's marketed firewalls is virtually impossible.

RUDP uses STUN/ICE/TURN as the basis for establishing peer-to-peer connections then uses a UDP packaging technique to deliver application data. Additionally because RUDP is a FIFO based protocol like TCP, it can layer protocols such as TLS directly above its transport with little to no change at all being required (other than pumping the data through an RUDP socket instead of a TCP socket).

RUDP uses ICE to perform connectivity probes between peers and utilizes a STUN extension for connecting, teardown, and reliable as well as unreliable data acknowledgements.

RUDP supports vector based acknowledgments and XOR bit parities to prevent malicious clients from being able to pretend a download stream was downloading faster than the server is truly capable of delivering.

RUDP further supports multiple channels with a single point-to-point connection and multiple connects on a single port between multiple points. This allows for an existing connectivity probe to be reused for sending additional streams of data without performing new connectivity checks.

RUDP does not offer security beyond connectivity security offered with STUN and ICE. However, TLS or other mechanisms can be layered on top of RUDP to provide security and encryption.

RUDP is designed to be firewall friendly with minimal overhead.

Comparison to Other Protocols

DCCP / DTLS

DCCP is a good message based protocol that allows for reliable connecting and tear down and congestion control but offers a lossy data stream. DCCP was not chosen as it was desirable to have a reliable transport protocol between peers. To add security DTLS can be layered on top of DCCP.

DCCP is not readably available on all platforms (requiring a layering over UDP on major platforms like Windows. To utilize DCCP on windows would require manipulating RAW sockets and implementing the full DCCP protocol from scratch. Alternatively, DCCP would have to run on top of UDP, which is an option but adds overhead.

As such to utilize DCCP would have required a layer as:

[application level reliability layer] -> DTLS -> DCCP (optionally over UDP)

Further using DCCP would still require utilizing extension protocols to perform peer to peer firewall probing in a manner like ICE performs.

DCCP was not chosen as the lack of data reliability, overhead and work involved to support all the layers was not considered viable.

TCP

TCP is very similar to RUDP in capabilities with one exception: the ease to penetrate firewalls between peers. TCP does not offer an ICE like mechanism to perform connectivity probes like ICE and thus was not chosen as an acceptable protocol.

SCP

SCP is an alternative TCP like peer-to-peer communication protocol that messages can be relayed over as an alternative to RUDP protocol should the underlying framework support SCP.

RUDP Protocol Specification

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890


 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Channel Number        |          Data Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Flags     |      Lower 24bits of Sequence Number          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Reserved    |             Lower 24bits of GSNR              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/                      Options and Padding                      /
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
/                       Application Data                        /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
Channel Number - in the same range as allowed by TURN (0x4000 -> 0x7FFF)
Data Length - how much application data is included in this packet after the
              header (0 is a valid length)
Flags - See definition below
Lower 24bits of Sequence Number - lower 24 bits of the 64bit sequence number
Reserved - must be set to "0" on send and ignored upon receipt
Lower 24bits of GSNR - Lower 24bits of the 64bit GSNR (Greatest Sequence
                       Number Received). If no packets have been received this
                       should be set to the NEXT-SEQUENCE-NUMBER as received
                       from the remote party.
Options and padding - If the EQ bit is set to zero in the flags then the
                      vector/GSNFR is included as part of the header.
Application data - Application data at the length of the data length


Flags are defined as follows
 0
```

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|P|P|X|D|E|E|A|R|
|S|G|P|P|C|Q|R| |
+-+-+-+-+-+-+-+-+
PS = Parity bit of sending packet (this packet)
PG = Parity of the Greatest Sequence Number Received (if no packets have been
     received yet then this value is "0")
XP = XOR'd parity of all packets received up-to-and-including the GSNFR (if
     no packets have been received then this value is "0")
DP = Duplicate packets have been received since the last ACK packet was sent.
EC = ECN (Explicit Congestions Notification) received on incoming packet
     since last packet in sequence sent. If no packets have been received then
     this value is set to "0".
EQ = GSNR == GSNFR (Greatest Sequence Number Received equals Greatest
     Sequence Number Fully Received). If no packets have been received then
     this value is set to "1".
AR = ACK required (must send a STUN "RELIABLE-CHANNEL-ACK"
     indication/request or another packet with ACK information (i.e.
     header only packet without data is okay).
R  = RFFU (Reserved For Future Use). Must be set to "0" on sending and ignored
     upon receipt


--------------------------------------------------------------------------------
This header is present in packet after the header if EQ flag is "0" (and
therefor cannot be present if no packets were received from the remote party
as the EQ value in this case must be "1").

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|P|Vector Length|             Lower 24bits of GSNFR             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/   [0...vector length] vector RLE information                  .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.            [padded RLE to next DWORD alignment (if required)] /
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
P - XOR'ed parity of all packets marked as received in the vector
    (starting at the calculated XORed to-date-up-to-and-including the GSNFR)
Vector Length - Total vector size included after header as expressed in
                DWORDs (if the only packet missing is the GSNFR+1 then
                vector can be zero)
Lower 24 bits of GSNFR - The lower 24 bits of the 64bit GSNFR (Greatest
                         Sequence Number Fully Received).

The entire packet including header cannot be over the PMTU or 512 bytes
if not known.

Sender will estimate the receiver's packet window. The sender will only
send packets that are in the sequence number range from the last reported
GSNFR up to the end of the receiver's
estimated packet window, i.e.:
((sequence_number > GSNFR) &&
 (sequence_number < (GSNFR + estimated_receivers_window)).

The sender will use a fairness algorithm to estimate the receiver's packet
window and adjust the window up and down according to its own policy on how
much data can be outstanding in the path in at half the RTT. The sender
can verify that the receiver has in fact received packets by way of the XOR'd
bit validation in a way that the receiver can't cheat and lie that it has
received packets when it has not. This prevents the receiver from maliciously
pretending it has received packets where it has not and causing the sender
to over-estimate the capacity of the path or miscalculate RTT by the
receiver acknowledging packets faster than it has actually received them.

The sender will cryptographically randomly choose a parity bit for every
packet is sends over the wire.

The sender will include the parity bit of the packet representing the GSNR
packet. The sender will include the XORed parity-to-date up-to-and-including
```

the GSNFR packet.

If the GSNFR equals the GSNR then the sender will set the EQ bit on the packet to 1 otherwise it will set it to 0 and include the vector/GSNFR additional header.

If the network in which the sender receives packets is ECN aware and marks packets with ECN, the sender will set the EC flag on a packet to 1 if the packet

The sender will mark the last packet in a series when no more data is available for sending at the moment with an AR flag.

The sender will automatically resend unacknowledged packets that are beyond haven't been acknowledged in the 2 times the total estimated RTT (Round Trip Time). The estimated RTT must never be lower than the negotiated MINIMUM-RTT.

A receiver can ACK packets received in two ways:
1) Any channel data packet sent in a series acts as an ACK for the channel
2) Send a STUN RELIABLE-CHANNEL-ACK indication or request

The receiver will ignore incoming packets with a sequence number that is less than the receiver's start window (the last fully ACKed packet) plus the receiver's window size. The receiver will ignore packets that have the GSNR parity incorrect for their sent packet. The receiver will close the connection if it receives any packets with the incorrect GP parity or the XP parity wrong from the same source:port:connection bound to the connection as this is an attempt either by a spoofer to inject data into the stream or by a client attempting to fake acknowledgements on packets it never received.

An IP spoofer could attempt to inject data into a stream by randomly flooding a receiver in attempt to hit within the sequence number window but they would have to fake the source IP:port and channel number in order for the attack to succeed. Thus it is recommended that the channel number is randomly chosen to make a spoof flood attack less likely to succeed.

Be aware that an IP spoofer may use the XP flag as an attack to attempt to close a connection to which they don't own by broadcasting packets but they are unlikely to know the correct sequence number window and channel number and thus would have to attempt to broadcast many packets in order to obtain a packet within the window. Obviously if they were sniffing and interfering with the network directly they could launch an attack but they already could interfere on a much deeper level in such situations which no protocol can stop but only detect. Adding security, such as TLS on top of RUDP is recommended to prevent faked data from being injected into a stream.

The receiver will ignore packets that are outside it's own receiving window (i.e. from the last fully ACKed packet to the last valid received packet plus the receiver's window). The receiver will ignore packets beyond its own receiving buffer capacity (i.e. total packets beyond a missing packet is greater than the receiver is willing to buffer).

If the AR flag was set on an accepted incoming packet for a packet with a sequence number greater than the last acknowledged, the sender will send an ACK packet immediately. The receiver can use a data packet for the ACK as long as it doesn't violate its own sending rules.

The receiver must send an ACK packet for packets that it didn't acknowledge yet within the window of the oldest unacknowledged packet plus one calculated RTT time frame. The calculated RTT must never be lower than the negotiated MINIMUM-RTT.

The receiver will acknowledge all packets it can every single data packet it sends out.

The receiver will calculate the latest RTT based on the acknowledgement of its last packet flagged with the AR bit. The calculated RTT must never be set lower than the negotiated MINIMUM-RTT.

With packets the receiver accepts, the receiver will look for

```
acknowledgements that it can verify as accurate with the parity bits.
Older packets containing acknowledgements where the data is still available
to validate the parities can be used to acknowledge packets but never be
used to mark already acknowledged packets as having not been received.

Vector format is as follows:
+--------+--------+--------+--------
|SSLLLLLL|SSLLLLLL|SSLLLLLL|  ...
+--------+--------+--------+--------

 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|Sta| Run Length|
+-+-+-+-+-+-+-+-+

Sta[te] occupies the most significant two bits of each byte and can
have one of four values, as follows:

State  Meaning
-----  -------
0    Received
1    Received ECN Marked
2    Reserved
3    Not Yet Received

A "0" vector byte is used at the end of a RLE series for padding to the next
DWORD alignment (and if interpreted would be seen as "0" packets received).

NOTE: There is no guarantee that a "0" byte will be contained at the end
      of any vector RLE series as it is only used for padding.


-------------------------------------------------------------------------------
STUN REQUEST: RELIABLE-CHANNEL-OPEN

This STUN request is used to open a channel to a remote party. In an ICE
environment, the requester is always the ICE controlling party.

Will contain following attributes:

If sent over non-ICE to open an anonymous channel:
First send request without any attributes with get 401 back with
NONCE/REALM back.

USERNAME - is set to userRandomFragRemote:userRandomFromLocal. The random frag
           should be globally unique to not cause conflict and unguessable.
PASSWORD - is set to the userRandomFragRemote. The password is not included
           directly but instead used in the MESSAGE-INTEGRITY calculation.
           When the server issues a request it will reverse the fragments
           and use the userRandomFromLocal as the password.

NONCE - as indicated by server
REALM - as indicated by server

If sent over an established ICE channel:
USERNAME - is set to the userFragRemote:userFragLocal of the nominated ICE
           pairing (just like ICE BINDING requests).
PASSWORD - is set to the ICE password of the remote party of the nominated
           ICE pairing (just like ICE BINDING request). The password is not
           included directly but instead used in the MESSAGE-INTEGRITY
           calculation. Short-term credential calculation is used.
NONCE/REALM - not used.

The request will always contain:
LIFETIME - set to how long the channel should remain open before it is
           automatically closed (in seconds). Setting to zero will
           cause the channel to close immediately and there is no need to
           contain NEXT-SEQUENCE-NUMBER, MINUMIM-RTT or CONGESTION-CONTROL.
           Any data received on the channel or RELIABLE-CHANNEL-ACK will
           cause the LIFETIME attribute timeout countdown to be reset to the
           default.
```

```
          If not specified, a LIFETIME of 10 minutes is assumed.
CHANNEL-NUMBER - set to the channel number the local party wishes the remote
                 party to use in all packets the remote party will send to
                 itself.
NEXT-SEQUENCE-NUMBER - set to the first sequence number-1 that
                       will be sent from this location (the first sequence
                       number must be at least 1 but less than 2^48-1)
MINUMIM-RTT - set to the number of milliseconds for the minimum RTT (Round Trip
              Time). The request may contain the MINUMIM-RTT attribute to
              indicate a minimum RTT it wishes to negotiate with the remote
              party.
CONGESTION-CONTROL - A list of congestion control algorithms available to use
                     by the sender with the preferred listed first. There
                     must be two of these attributes, one representing the
                     local (requester) congestion to use and one representing
                     the remote (responder) congestion algorithm.
CONNECTION-INFO - A string representing whatever additional information is
                  required to exchange upon connection.

Response will contain (signed with message integrity):
LIFETIME - The responder can always choose a value lower than the requested
           LIFETIME of the requester but never can respond with "0" unless
           the requester sent "0". This is a negotiated value between requester
           and responder. The channel is kept alive by any channel data being
           sent or by RELIABLE-CHANNEL-ACK requests/indications. If the
           responder wishes to close the channel at a later date the responder
           can chose to issue its own CHANNEL open in the reverse direction
           with a LIFETIME of "0" with the CHANNEL-NUMBER being set to the
           CHANNEL-NUMBER the responder is currently expecting to receive
           from the remote party.

           If not specified, a LIFETIME of what the requester asked is assumed.
NEXT-SEQUENCE-NUMBER - set to the first sequence number-1 that
                       will be sent from this responder (the first sequence
                       number must be at least 1 but less than 2^48-1).
CHANNEL-NUMBER - set to the channel number the responder party wishes the
                 requester to use in all packets it will send to the
                 responder.
MINUMIM-RTT - set to the number of milliseconds for the minimum RTT (Round Trip
              Time) that the response party will accept. If the response
              agrees with the minimum value by the requester it does not
              need to include this attribute. The response may contain this
              attribute value containing a larger than the requester if it
              wishes to negotiate a larger minimum RTT between the two parties
              but can never choose a shorter minimum RTT than the requester.
CONGESTION-CONTROL - A list of congestion control algorithms available to use
                     by the receiver with the selected algorithm listed first.
                     The selected algorithm must be within the list offered
                     by the requester. If the attribute is missing then the
                     responder is assumed to use the algorithm preferred by
                     the requester. Typically, two of these attributes are
                     present in the response, one for the local (i.e.
                     responder) and one for the remote (requester).
CONNECTION-INFO - A string representing whatever additional information is
                  required to exchange upon connection.


When a channel is open for the first time, the responder does not start
sending data until the requester first sends data or sends an ACK. This is
required to ensure the response actually arrived to the requester and thus
proving the negotiation completed.

If either party responds to a renegotiation attempt (i.e. a new
RELIABLE-CHANNEL-OPEN with changed attributes on the same channel, it must
cease sending channel data until a data packet or ACK is received with a
remote sequence number equal or than the sequence number in the request.

If both parties attempt a simutanious renegotiation attempt a
"487 Role Conflict" should result unless the negotiated request from the
remote party is completely compatible with the outstanding negotiated
```

request issued from the local party.

If either party was attempting to close the channel but an error was received as a response, the channel should therefor be considered closed (except in the case where the NONCE is reported as stale).

STUN REQUEST/INDICATION: RELIABLE-CHANNEL-ACK

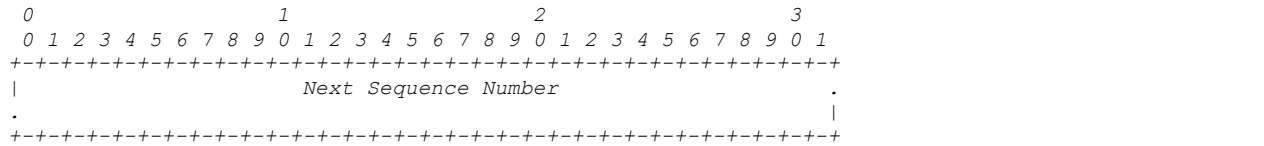
Either party can send this as a request or indication. The NONCE/REALM are only needed on a non-ICE situations. All other attributes must be present in request, except the ACK-VECTOR if it is not needed (i.e. when the only packet sequence number missing is the GSNR-1). If not send as an indication then a response is required and the response must contain the same attributes as listed for the request except the USERNAME, NONCE and REALM.

USERNAME - same logic as RELIABLE-CHANNEL-OPEN
PASSWORD - same logic as RELIABLE-CHANNEL-OPEN
REALM/NONCE - same logic as RELIABLE-CHANNEL-OPEN
CHANNEL-NUMBER - set to the channel number the local party wished the remote party to use in all packets the remote party sent to itself.
NEXT-SEQUENCE-NUMBER - set to the next sequence number the requester will send over the wire (but has not sent yet).
GSNR - set to the greatest sequence number seen from the remote party.
GSNFR - set to greatest sequence number up to which all packets have been received.
RELIABLE-FLAGS - Flags indicating the parity or other useful information
ACK-VECTOR - Vector RLE in the same fashion as in the data packet.

A successful response (MESSAGE-INTEGRITY is not required) will indicate closure is complete. A failure response indicates the request/channel was not understood properly and the client should consider it closed, except a 483 where a packet must be re-issued to satisfy the NONCE being stale.

STUN ATTRIBUTE: NEXT-SEQUENCE-NUMBER

This is a 64bit unsigned integer attribute indicating the next sequence number the requester or responder expects to send (but has not sent).



STUN ATTRIBUTE: GSNR

This is a 64bit unsigned integer attribute indicating the Greatest Sequence Number Received by the requester/responder encoded in the same method as the NEXT-SEQUENCE-NUMBER attribute.

STUN ATTRIBUTE: GSNFR

This is a 64bit unsigned integer attribute indicating the Greatest Sequence Number Fully Received by the requester/responder encoded in the same method as the NEXT-SEQUENCE-NUMBER attribute. In other words, all the packets that have been received to date up to a certain sequence number. If the GSNR is the same value as the GSNFR then this attribute is optional. If this attribute was not received on a RELIABLE-ACK then the GSNFR is assumed to be the same value as the GSNR.


```
--------------------------------------------------------------------------------
STUN ATTRIBUTE: MINIMUM-RTT

This is a 32bit unsigned integer representing the minimum RTT in milliseconds
negotiated by the two parties.

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Minimum-RTT                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


--------------------------------------------------------------------------------
STUN ATTRIBUTE: CONNECTION-INFO

An encoded string used at channel open to add additional information about
the connection. The interpretation is entirely dependant on the context.


--------------------------------------------------------------------------------
STUN ATTRIBUTE: RELIABLE-FLAGS

The reliable flags are flags needed to indicate the parity bits and other
acknowledgement flags encoded in 4 bytes. The first byte is the only byte used
at this time.

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V|P|X|D|E|   R   | RFFU (Reserved For Future Use)                |
|P|G|P|P|C|       | (must be set to "0" on send and ignored)      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

VP - XOR'ed parity of all packets marked as received in the ACK-VECTOR
     attribute (starting with the calculated XORed to-date-up-to-and
     including the GSNFR) - Same meaning as the "P" flag on the vector/GSNFR
     header in the data packet.
PG = Parity of the Greatest Sequence Number Received
XP = XOR'd parity of all packets received up-to-and-including the GSNFR
DP = Duplicate packets have been received since the last ACK packet was sent.
EC = ECN (Explicit Congestions Notification) received on incoming packet
     since last packet in sequence sent
R  = RFFU (Reserved For Future Use). Must be set to "0" on sending and ignored
     upon receipt


--------------------------------------------------------------------------------
STUN ATTRIBUTE: ACK-VECTOR

Has the same meaning and encoding as the optional vector encoded after the
vector/GSNFR header. The "P" flag from the vector header is contained in the
VP flag of the RELIABLE-FLAGS attribute.


--------------------------------------------------------------------------------
STUN ATTRIBUTE: CONGESTION-CONTROL

 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|D|     RFFU    |     RFFU      | Profile preferred or selected |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
/ Profile preferred or selected | Profile preferred or selected /
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
/ Profile preferred or selected | [Profile/padding as required] /
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+

The first byte is reserved for flags with only one flag available at this time.
D = Direction. If "0" the congestion control list applies to the "local"
               party. If "1" the congestion control list applies to the
```

```
                "remote" party.
                When receiving a request, the remote will apply to the responder
                and the local will apply the requester. When receiving a reply
                the remote will apply to the requester and the local will apply
                to the responder.
RFFU - All bits should be set to "0" and ignored upon receipt.

The second byte is RFFU.

This is a list of unsigned 16bit integers representing the congestions
profile algorithms offered or accepted. The preferred or selected algorithm
must be listed first. The order of the algorithms is assumed to be the
preferred order of the requester or responder. The responder must select an
algorithm within the list offered by the remote party.
```

Open Peer Signalling Protocol

Non Encrypted JSON Signalling Protocol

The signalling protocol used for Open Peer is simple JSON based protocol and uses RUDP or SCP as the transport.

The packaging of an JSON message for deliver to a peer entity is extremely simple when no encryption is used (known as "text/x-open-peer-json-plain"):

- Message Size [4 bytes in network order] - The length of the raw JSON message about to be received
- JSON data – the JSON message data to be receive of exactly the message size specified (no NUL termination is required or expected).

Encrypted JSON Signalling Protocol Using TLS

Open Peer can utilize standard TLS to connection from a peer to a server. In this mode the messaging is encoded exactly the same except delivered through a TLS pipe over RUDP, also using the RUDP mime type of "text/x-open-peer-json-tls".

Encrypted JSON Signalling Protocol Using Messaging and Not Using TLS

Open Peer has one important expectation difference from the typical TLS scenario and thus offers an an alternative offering to TLS for signalling called "text/x-open-peer-json-mls", where MLS = Message Layer Security as opposed to TLS which is Transport Layer Security).

TLS is majority used by HTTPS (although not exclusively) under a scenario where an anonymous client without any public/private key connects to a server that has a public/private key whose identity is validated from a trusted authority chain, such as VeriSign or Thawte. In such a situation, TLS requires negotiation to establish a bidirectional channel with known trust chains to verify the servers identity and prevent man-in-the-middle attacks without ever being able to validate the identity of the client (unless prearranged private data is exchanged additionally in the application layer).

In the Open Peer case, all peers have a public and private key, without exception, be it peer to peer or peer to server. In all cases, a peer initiating a connection always knows the public key of the designation peer in advance (thus avoiding man-in-the-middle attacks) of the connection itself. Trust is established via the Bootstrapper's introduction to services as well as Identity Providers that provide Asserted Identities.

Open Peer connections can take advantage of this situation by utilizing the pre-known public key in the initiating side to simplify the negotiation scenario and offer unidirectional encrypted streams.

The format for the unidirectional message is as follows:

- Encryption key algorithm selection (16 bits network byte order, upper 8 bits reserved and must be set to "0") – When negotiating, each number represents selected keys / algorithm pair for use by the number chosen but "0" is used to represent a key/algorithm negotiation. Every "0" key causes a reset of all encryption algorithms in progress to substitute with the values specified in the "0" package. Each key / algorithm selected is selected from the supported keys/algorithms offered to the remote party, but can only be select using algorithms the remote party supports. As such, there is one mandated algorithm to ensure compatibility, "http://openpeer.org/2012/12/14/jsonmls#aes-cfb-32-16-16-sha1", where the AES (Rijndael-128) in CFB mode with a 32 byte key size, 16 byte block size, and a 16 byte feedback size with a SHA1-HMAC.
- 32 bit - encrypted data bundle size
- Data bundle, consisting of:
    - JSON message encrypted using the algorithm/key selected
    - hmac of the data encrypted using the algorithm/key selected

The advantage of pre-knowing the public key by the sender allows for unidirectional encryption with different keys being used in each direction and allows for encryption to begin in both directions in a single round trip negotiation.

Message level security is not to be used except for this specific scenario and only when the keys are pre-known by the initiator of the connection and where both parties have public and private keys. Further, this is a message centric encryption and not stream level encryption as offered by TLS. Any received public key and Asserted Identities must be validated at the application layer by exchanging Asserted Identity information in correlation with Public Peer Files.

The "0" package is sent in plain text in JSON format and the data bundle has a 0 byte size hmac since the signature as part of the JSON package is used instead.

The "0" package contains the following:

- Signed keying bundle including:
    - Nonce – this nonce should be validated as having only been seen once by the receiving client
    - Expiry – this package must be verified as valid before the expiry or it's considered invalid
    - Algorithms – preference ordered set of algorithms supported by the client
    - List of keys, with each key containing:
        - key ID – this corresponds to the algorithm selection ID of which key to use in any subsequent decryption
        - algorithm – the algorithm to use when decrypting payloads using this key ID
        - algorithm input data – each algorithm requires its own set of keying information required for decryption, which is contained here. All sensitive data is encrypted using the public key of the remote party

For the mandatory "aes-cfb-32-16-16-sha1" algorithm, the following algorithm input information is used:

- key – the 32 byte AES key, base64encode(remote_public_key_encrypt(<32-byte-aes-key>))
- iv – the 16 byte AES initialization vector, base64encode(remote_public_key_encrypt(<32-byte-aes-key>))
- hmacSecretKey – the initial secret key input string, base64encode(remote_public_key_encrypt(<secret-key-string>))

When the mandatory key is used, the AES CFB is initialized with these values and will continue to encrypt payloads using this keying until a new "0" package arrives which would reset the algorithms with new keying information for subsequent messages in the message stream. The hmac is calculated using the following algorithm, hmac(<secret-key-string> + ":" + <sequence-number>, <decrypted-message>)), where the sequence number starts at 1 and increments by 1 each time the same "encryption key algorithm" is used (and resets back to 1 the next time a new "0" package is received). The sequence number is appended to prevent the same message repeated from containing the same hmac hash proof in the result.

Any sensitive data contained in the "0" package must be encrypted using the public key of the receiving party. The entire package must be signed by the public key of the party sending the "0" package and the receiver must validate this package's signature before it assumes any of the data sent in the package is considered valid.

The party receiving the connection request cannot respond until it knows the corresponding public key of the initiator of the request, which must match and verify the signature used in the initiator's original "0" package. The "0" package must be the first package sent on the wire in either direction. The party receiving the connection request must assume all data received via encrypted messages may not be valid until it can verify the signature in the original "0" package and all subsequent "0" packages. The public key used in the "0" packages must never change throughout the lifetime of the connection.

All keying information and salts must be generated using cryptographically random algorithms only.

The algorithms used for encrypting must be limited to the algorithms supported by the remote party, but the mandatory "aes-cfb-32-16-16-sha1" algorithm must always be considered a valid algorithm available in any minimal implementation.

Example of the "0" package is JSON in place text with the following data in the bundle:

```
{
  "keyingBundle": {
    "keying": {
      "$id": "f25f588141f7232e40b1529667b8ea626d078d20",
      "nonce": "11a9960ebfe2287c1e235aceb912d8d54532be05",
      "expires": "348498329",
      "algorithms": {
        "algorithm": [
          "http://openpeer.org/2012/12/14/jsonmls#aes-cfb-32-16-16-sha1",
          "http://openpeer.org/2012/12/14/jsonmls#aes-cfb-16-16-16-md5"
        ]
```

```
      },
      "keys": {
        "key": [
          {
            "$id": 1,
            "algorithm": "http://openpeer.org/2012/12/14/jsonmls#aes-cfb-32-16-16-sha1",
            "inputs": {
              "key": "Y21wclpXd...HFjbXgzWlhKbA==",
              "iv": "Y21wclpXd...HFjbXgzWlhKbA==",
              "hmacSecretKey": "VjFSSmVHUXlUbk5qU...aHVaV3hrYzJGRmRHbFJWREE1"
            }
          },
          {
            "$id": 2,
            "algorithm": "http://openpeer.org/2012/12/14/jsonmls#aes-cfb-32-16-16-sha1",
            "inputs": {
              "key": "WTIxd2NscFhkSEZq...YlhneldsaEtiQT09",
              "iv": "V1RJeGQyTnNjRmhrTGk...bGhuZWxkc2FFdGlRVDA5",
              "hmacSecretKey": "ZmpGU1NtVkhVW...MQ=="
            }
          },
          {
            "$id": 3,
            "algorithm": "http://openpeer.org/2012/12/14/jsonmls#aes-cfb-16-16-16-md5",
            "inputs": {
              "key": "Wm1wR1d4Vlltc...mtwWFVrVkZNUT09",
              "iv": "V20xd1IxZDRWbGx...dGVnJWa1pOVVQwOQ==",
              "hmacSecretKey": "VjIweGQxSXhaRFJX...WUXdPUT09"
            }
          }
        ]
      }
    },
    "signature": {
      "reference": "#f25f588141f7232e40b1529667b8ea626d078d20",
      "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
      "digestValue": "OzIyMmI3NG...WJlOTY4NmJiOWQwYzkwZGYwMTI=",
      "digestSigned": "G4Fwe...0E/YT=",
      "key": { "uri": "peer://example.com/ab43bd44390dabc329192a392bef1" }
    }
  }
}
```

General Request, Reply, Notify and Result Formation Rules

All request types and results use a simplified JSON format. The messages are sent either over HTTPS/TLS/MLS/Message or over RUDP/UDP or SCP protocols. Alternative protocols are acceptable so long as they maintain the integrity and public/private key aspects of these protocols.

Every request type must include an ID and a method being invoked. Every result message must mirror the request type's ID and include an epoch (whereas the epoch is optional on request types).

Even though all requests/responses are written in human readable form in this document, all requests/responses are sent on the wire in their Open Peer canonical form using the algorithm for the canonicalization of JSON signatures as specified in this document. This allows any clients or servers receive JSON requests that have non-compliant JSON parsers/generators to be able to easily validate signatures even if they cannot convert from raw JSON to canonical JSON easily. Further, this ensures the wire format is optimal on the wire since the canonical form is fairly compact (although if compression is applied in a higher layer then the wire savings might become moot).

Open Peer JSON signatures are used to verify signatures within the JSON. The canonical form of JSON is as follows (unless otherwise specified within the JSON signature):
http://openpeer.org/2012/12/14/jsonsig#rsa-sha1

This algorithm allows only JSON objects to be signed. The digest value is sha1(<message>), where the message is the canonical version of the JSON object written to a string. The object must be rendered to a string as if it were a standalone rendered JSON object where the final message is in the format:
{"name":{...}}

If the rendered object doesn't have a string part in the JSON lexical string:value pairing (as it's a value-only) then the object name is based upon the parent array's/object string in its string:value pair (or the parent's parent as need be). This canonical rendered string requires absolutely no white space between tokens. All strings must be in normalized to UTF-8 format with only these escape sequences used:
\" \\ \f \n \r \b

Unicode escape sequences are converted to UTF-8 format. Number sequences must not have unnecessary leading or trailing zeros. Numbers are rendered "as is", i.e. in the format they are put inside the original JSON package where the signature is applied.

Any object's string:value pairing that begin with "$" in the string part are placed in the order they appear in the JSON package where the signature is applied before any other string:value pairs inside the object. Next any object containing a string:value pair who's lexical string name is "#text" is placed after those strings starting with "$". Finally, all remaining string:value pairs follow in the order they appear in the JSON package where the signature is applied.

The "$", "#text" and 'other' ordering is used to ensure the implementations which wish to process the JSON in XML form can still render back/forth from XML to JSON and still be capable of generating

signatures properly. The "$" prefix was chosen specifically because it is a non special variable name in JavaScript where JSON has been most prominently adopted but yet still denotes a special character for the sake of JSON/XML conversions.

JSON objects that have member string:value pairs that start with "$" in their string parts are only allowed to contain string, number, true, false or null values. JSON objects that have a member string:value pair with the string part equal to "#text" can only have a string as the corresponding value. This restriction is put into Open Peer to allow for easy JSON/XML conversions.

All base64 encoding/decoding routines must use standard/common encoding scheme, without line breaks and with padding used for unused bytes.

The client may receive an error complaining that a request has expired if the client's clock was set wrong (401). Hence in every result, the epoch of the server will be sent for the client to detect potential clock errors. To reduce issues, the client should use a secure NTP service to set its own clock at some point before initiating contact to a server or another peer.

The client is responsible for disconnecting at all times from the server. In the case of peer to peer, the initiating peer is considered the client role and the receiving peer plays the server role.

There are exceptions to this rule. The server will close a connection without warning based on two inactivity time-outs. The larger timeout is based upon an expiry window when the entity is known or "registered" to the server. The smaller timeout window of inactivity (chosen and unspecified at the discretion of the server) is based on not having received any request or notification on a channel within that defined timeframe. If either of those two timeouts occurs, the server may disconnect which is typically the responsibility of the client. The server may disconnect any client it sees as likely malicious behaviour.

If a client disconnects without sending the unregister request, the server should assume the client disconnected prematurely and will discard any associated sessions.

Other disconnection rules are specified whenever they are exceptions to the rule or the exceptions.

General Result Error Code Reasons

Error replies appear as follows:

```
{
  "result": {
    "$method": "method",
    "$timestamp": 439439493,
    "$id": "abc123",

    "error": {
      "reason": {
        "$id": 404,
        "#text": "Not Found"
      }
    }
  }
}
```

The reasons codes closely match the HTTP error code specification for familiarity. Error results may contain additional information depending on the error and requirements of the error result.

301 – Moved Permanently

The requested Peer Contact has changed and is now registering itself under a new Peer Contact. The client's response should be to resolve again the Identity that pointed to the original Peer Contact in an attempt to locate the new Peer Contact.

400 – Bad Request

The method in the request is not valid.

401 – Unauthorized

One of the security checks has failed to pass in the request. The server may issue information on what exactly failed to authorize to assist debugging the issue.

403 – Forbidden

The data specified is invalid and fixing any security check issues will not fix the situation. The server may issue information on what exactly was the issue with the data to assist debugging the issue.

404 – Not Found

This error is returned if the requested Peer Contact, session or other resource is not found. This error is also returned from any request where the session or Peer Contact was valid but is no longer valid, e.g. situations where requests have been made to sessions which have already been unregistered yet unknown to the connected client due to the nature of asynchronous eventing.

A 404 error does not mean the resource never existed or will not exist in the future. The contact may not be registered at this time and that would cause a 404 error even though in the past it may have been registered.

409 - Conflict

A conflict has occurred, such as an edit conflict with version numbers.

426 - Upgrade Required

A client has requested a method that is not accessible since an upgrade is required. This will be sent if a client's certificates have expired and attempt to access a method or may be sent if a client is using an out-dated request.

480 – Temporarily Unavailable

The request may optionally include an expiry when the request can be tried again.

Bootstrapper Service Requests

The communication to the Bootstrapper is done over HTTPS exclusively whose HTTPS server certificate was signed by one of the trusted root Internet certification authorities. For security purposes, the Bootstrapper is the introducer to all other services within the network including appropriate security credentials to connect to each network component.

Locating the Boostrapper

Overview

The Bootstrapper is the introductory service into the domain responsible for a Peer Contact and DNS is used to locate the Bootstrapper.

A peer contact is written in the following form:
peer://domain.com/e433a6f9793567217787e33950211453582cadff

And an identity is written in the following form:
identity://domain.com/alice

In both cases, an HTTPS request is this performed on "domain.com", using the following URL:
https://domain.com/.well-known/openpeer-services-get

Clients must confirm the HTTPS certificates comes from the same domain as the original domain request and reject any records that do not.

Services Get Request

Purpose

This request is required to obtain a list of services available on the peer network as well as establish a hierarchy of certificate trust.

Inputs

None.

Returns

- Service ID
- Service Type
- API Version
- URI – (optional) – if the service is based on a single URI then this is returned, otherwise individual methods are specified and each service type will determine if a single URI is needed versus listing individual methods involved.
- List of requests methods, which includes
    - Method name

    - URI
    - Other request specific information
- Public key information – (optional) used when protocol used is not based on a root certificate authority

Security Considerations

The client must ensure the server has an HTTPS certificate that was issued by a root certificate authority and that the certificate offered by the server is still within the X.509 validity date. The certificate authority for some Bootstrapped Networks may be pre-built into a client application and verified as accurate and can respond to mismatch as deemed appropriate by the client. The client may issue more than one certificate per service should an overlap window of X.509 certificate validity be required.

The server does not need to know or verify a client's intentions.

A 302-redirect error response can be returned by this response to allow the request to be redirected to another server. This allows this service to be easily hosted as needed.

The request nor the response should have an ID associated with the request/response and should not include an epoch. This is the only request in the system that has this exception. This allows for a hard-coded file to be uploaded on a server as the response to any request on the system to allow for easy service delegation without installing any server side scripting.

Example

```
{
  "request": {
    "$domain": "example.com",
    "$handler": "bootstrapper",
    "$method": "services-get"
  }
}
```

```
{
  "result": {
    "$domain": "example.com",
    "$handler": "bootstrapper",
    "$method": "services-get",
    "$timestamp": 439439493,

    "services": {
      "service": [
        {
          "$id": "9bdd14ddad8465b6ee3fdd174b5d5bd2",
          "type": "bootstrapper",
          "version": "1.0",
          "methods": {
            "method": {
              "name": "services-get",
              "uri": "https://bootstrapper.example.com/services-get"
            }
          }
        },
        {
          "$id": "596c4577a4efb2a13ded43a3851b7e51577ad186",
          "type": "bootstrapped-finders",
          "version": "1.0",
          "methods": {
            "method": {
```



```
            "name": "finders-get",
            "uri": "https://finders.example.com/finders-get"
          }
        }
      },
      {
        "$id": "596c4577a4efb2a13ded43a3851b7e51577ad186",
        "type": "certificates",
        "version": "1.0",
        "methods": {
          "method": {
            "name": "certificates-get",
            "uri": "https://certificates.example.com/certificates-get"
          }
        }
      },
      {
        "$id": "0c16f792d6e0727e0acdd9174ae737d0abedef12",
        "type": "identity-lockbox",
        "version": "1.0",
        "methods": {
          "method": [
            {
              "name": "public-peer-files-get",
              "uri": "https://peer-contact.example.com/public-peer-files-get"
            },
            {
              "name": "peer-contact-login",
              "uri": "https://peer-contact.example.com/peer-contact-login"
            },
            {
              "name": "private-peer-file-get",
              "uri": "https://peer-contact.example.com/private-peer-file-get"
            },
            {
              "name": "private-peer-file-set",
              "uri": "https://peer-contact.example.com/private-peer-file-set"
            },
            {
              "name": "peer-contact-identity-associate",
              "uri": "https://peer-contact.example.com/peer-contact-identity-associate"
            },
            {
              "name": "peer-contact-identity-association-update",
              "uri": "https://peer-contact.ex.com/peer-contact-identity-association-update"
            },
            {
              "name": "peer-contact-services-get",
              "uri": "https://peer-contact.example.com/peer-contact-services-get"
            }
          ]
        }
      },
      {
        "$id": "d0b528b3f8e66455d154b1deac1e357e",
        "type": "identity-lockbox",
        "version": "1.0",
        "methods": {
          "method": [
            {
              "name": "lockbox-access",
              "uri": "https://lockbox.example.com/lockbox-access"
            },
            {
              "name": "lockbox-identities-update",
              "uri": "https://lockbox.example.com/lockbox-identities-update"
            },
            {
              "name": "lockbox-permissions-grant-inner-frame",
              "uri": "https://lockbox.example.com/lockbox-permissions-grant-inner-frame"
            },
```

```
            {
              "name": "lockbox-content-get",
              "uri": "https://lockbox.example.com/lockbox-content-get"
            },
            {
              "name": "lockbox-content-set",
              "uri": "https://lockbox.example.com/lockbox-content-set"
            },
            {
              "name": "lockbox-admin-inner-frame",
              "uri": "https://lockbox.example.com/lockbox-admin-inner-frame"
            }
          ]
        }
      },
      {
        "$id": "d0b528b3f8e66455d154b1deac1e357e",
        "type": "identity-lookup",
        "version": "1.0",
        "methods": {
          "method": [
            {
              "name": "identity-lookup-check",
              "uri": "https://identity-lookup.example.com/identity-check"
            },
            {
              "name": "identity-lookup",
              "uri": "https://identity-lookup.example.com/identity-lookup"
            }
          ]
        }
      },
      {
        "$id": "f98b4d1ff0f1acf3054fefc560866e61",
        "type": "identity",
        "version": "1.0",
        "methods": {
          "method": [
            {
              "name": "identity-access-inner-frame",
              "uri": "https://identity.example.com/identity-access-inner-frame"
            },
            {
              "name": "identity-access-validate",
              "uri": "https://identity.example.com/identity-access-validate"
            },
            {
              "name": "identity-lookup-update",
              "uri": "https://identity.example.com/identity-lookup-update"
            },
            {
              "name": "identity-sign",
              "uri": "https://identity.example.com/identity-sign"
            }
          ]
        }
      },
      {
        "$id": "2b24016d58b04f0a3b157a82ddd5f18b44d8912a",
        "type": "peer",
        "version": "1.0",
        "methods": {
          "method": {
            "name": "peer-services-get",
            "uri": "https://peer.example.com/peer-services-get"
          }
        }
      },       {
        "$id": "db144bb314f8e018f103033cbba7d52e",
        "type": "salt",
        "version": "1.0",
```

```
          "methods": {
            "method": {
              "name": "signed-salt-get",
              "uri": "https://salt.example.com/signed-salt-get"
            }
          }
        },
        {
          "$id": "db144bb314f8e018f103033cbba7d52e",
          "type": "example",
          "version": "1.0",
          "key": {
            "$id": "8cd14dda3...d5bd2",
            "domain": "example.com",
            "service": "something"
          },
          "methods": {
            "method": {
              "name": "example-method",
              "uri": "peer://example.com/5ff106c7db894b96a1432c35c246f36d8414bbd3"
            }
          }
        }
      ]
    }
  }
}
```

Example (redirect)

```
{
  "request": {
    "$domain": "example.com",
    "$handler": "bootstrapper",
    "$method": "services-get"
  }
}
```

```
{
  "result": {
    "$domain": "example.com",
    "$handler": "bootstrapper",
    "$method": "services-get",

    "error": {
      "$id": 302,
      "#text": "Found",
      "location": "http://someserver.com/services-get"
    }
  }
}
```

Bootstrapped Finder Service Requests

Finders Get Request

Purpose

This request returns a list random possible peer finders that a client can attempt a connection for the sake of registering or finding other peers.

Inputs

The total number of server entries desired (which the server can choose to ignore and return less servers than requested, but never more).

Returns

Returns a list of Finders containing the following information for each Finder:

- Finder ID – the unique ID that represents this finder in the system
- Transport
- SRV record [or pre-resolved comma separated IP:port pair locations] – SRV lookup type is _finder._udp.domain.com
- Public key for the finder – Can be either the full X.509 certificate or a key name lookup for certificates returned from the certificate server
- Weight / priority - default values for SRV like weight/priority when SRV entry is pre-resolved IP:port pairs
- Geographic region ID – (optional) each server belongs to a logical geographic region (clients can organize servers into geographic regions for fail over reasons)
- Created – the epoch when the finder registered itself to the Bootstrapped Finder service. A finder with the same ID but a newer created date should replace an existing finder with the same ID.
- Expires – the epoch when this finder information should be discarded and a new finder fetched to replace the existing one. There is no guarantee the finder will remain online for this period of time as this is a recommendation only. Should initiated communication to a finder server fail, the finder information might be considered no longer valid as the finder server might be gone.

Security Considerations

The client must ensure the server has an HTTPS certificate that was issued by a root certificate authority and that the certificate offered by the server is still within the X.509 validity date. The client should check the validity of each finder by verifying each finder was signed by a "Finder" service for the same domain as the requested Bootstrapper. The server does not need to verify a client's intentions.

The finder information can be cached and the client can reconnect to the same finder at will in the future. The finder server is considered transient though and may at any time disappear from offering service.

Each Finder should have its own X.509 certificate that it generates upon start-up and reports through whatever secure mechanism to the Bootstrapper Finder Service. This causes each finder to use it's own keying information which is not shared across finders. However, this is not a hard requirement and the Finders may use a common key across all Finders.

Example

```
{
  "request": {
    "$domain": "example.com",
    "$handler": "bootstrapper-finder",
    "$method": "finders-get",
    "$id": "abd23",

    "servers": 2
  }
}
```

```
{
  "result": {
    "$domain": "example.com",
    "$handler": "bootstrapper-finder",
    "$method": "finders-get",
    "$id": "abc123",
    "$timestamp": 439439493,

    "finders": {
      "finderBundle": [
        {
          "finder": {
            "$id": "4bf7fff50ef9bb07428af6294ae41434da175538",
            "transport": "rudp/udp",
            "srv": "finders.example.com",
            "key": { "x509Data": "MIIDCCA0+gA...lVN" },
            "priority": 1,
            "weight": 1,
            "region": "1",
            "created": 588584945,
            "expires": 675754754
          },
          "signature": {
            "reference": "#4bf7fff50ef9bb07428af6294ae41434da175538",
            "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
            "digestValue": "jeirjLr...ta6skoV5/A8Q38Gj4j323=",
            "digestSigned": "DE...fGM~C0/Ez=",
            "key": {
              "$id": "9bdd14dda3f...dd174b5d5bd2",
              "domain": "example.org",
              "service": "finder"
            }
          }
        },
        {
          "finder": {
            "$id": "a7f0c5df6d118ee2a16309bc8110bce009f7e318",
            "transport": "rudp/udp",
            "srv": "100.200.100.1:4032,5.6.7.8:4032",
            "key": { "x509Data": "MIID5A0+gA...lVN" },
            "priority": 10,
            "weight": 0,
            "region": 1
```

```
          },
          "signature": {
            "reference": "#a7f0c5df6d118ee2a16309bc8110bce009f7e318",
            "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
            "digestValue": "YTdmMGM1ZGY2Z...DExOGVlMmExNjMwJjZTAwOWY3ZTMxOA==",
            "digestSigned": "OjY2OjZl...OjZhOjcyOjY2OjcyOjcyIChsZW5ndGg9OSk=",
            "key": {
              "$id": "9bdd14dda3f...dd174b5d5bd2",
              "domain": "example.org",
              "service": "finder"
            }
          }
        }
      ]
    }
  }
}
```

Certificates Service Requests

Certificates Get Request

Purpose

This request returns a list of public key X509 certificates used for signing in the domain for every service.

Inputs

None.

Returns

Returns a list of service certificates containing the following information for each certificate:

- Certificate ID
- Service name
- Expiry
- X.509 public key certificate

Security Considerations

The client must ensure the server has an HTTPS certificate that was issued by a root certificate authority and that the certificate offered by the server is still within the X.509 validity date. The server does not need to verify a client's intentions. The client should verify that each key was signed correctly from the Bootstrapper service key. This allows the clients to cache the certificate bundles while avoiding potential tampering.

Example

```
{
  "request": {
    "$domain": "example.com",
    "$id": "abd23",
    "$handler": "certificates",
    "$method": "certificates-get"
  }
}
```

```
{
  "result": {
    "$domain": "example.com",
    "$id": "abc123",
    "$handler": "certificates",
    "$method": "certificates-get",
    "$timestamp": 439439493,

    "certificates": {
      "certificateBundle": [
        {
          "certificate": {
            "$id": "4bf7fff50ef9bb07428af6294ae41434da175538",
            "service": "finder",
            "expires": 48348383,
            "key": { "x509Data": "MIIDCCA0+gA...lVN" }
          },
```

```
          "signature": {
            "reference": "#4bf7fff50ef9bb07428af6294ae41434da175538",
            "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
            "digestValue": "jeirjLrta6skoV5/A8Q38Gj4j323=",
            "digestSigned": "DEf...GM~C0/Ez=",
            "key": {
              "$id": "9bdd14...dda3fddd5bd2",
              "domain": "example.com",
              "service": "boostrapper"
            }
          }
        },
        {
          "certificate": {
            "$id": "9bdd14...dda3fddd5bd2",
            "service": "bootstrapper",
            "expires": 48348383,
            "key": { "x509Data": "OWJkZD...GQ1YmQy=" }
          },
          "signature": {
            "reference": "#9bdd14...dda3fddd5bd2",
            "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
            "digestValue": "amVpcmpMcn...RhNnNrb1Y1L0E4UTM4R2o0ajMyMz0=",
            "digestSigned": "YW1WcGNtcE1jb...lJoTm5OcmIxWTFMMEU0VVRNNFIybzBhak15TXowPQ==",
            "key": {
              "$id": "9bdd14...dda3fddd5bd2",
              "domain": "example.com",
              "service": "boostrapper"
            }
          }
        },
        { ... }
      ]
    }
  }
}
```

Identity Lockbox Service Requests

Lockbox Access Request

Purpose

This request obtains access to a lockbox. Access is granted by way of login proof to an identity that is allowed to have access to the lockbox.

Inputs:

- Client nonce - a onetime use nonce, i.e. cryptographically random string
- Identity information (optional, if logging in using an identity)
    - Identity access token – as returned from the "identity access complete" request
    - Proof of 'identity access secret' – proof required to validate that the 'identity access secret' is known, proof = hmac(<identity-access-secret>, "identity-access-validate:" + <identity> + ":" + <client-nonce> + ":" + <expires> + ":" + <identity-access-token> + ":lockbox-access")
    - Expiry of the proof for the 'identity access secret' – a window in which access secret proof is considered valid
    - Original identity URI
    - Identity provider (optional, required if identity does not include domain or if domain providing identity service is different)
- Grant
    - ID – a client generated cryptographic unique ID representing the agent's permission to access the lockbox. Once this ID is generated by a client, it should remain stable in subsequent accesses (or a new permission grant will be required). This ID should remain secret to the client application.
- Lockbox information
    - Lockbox domain – the domain hosting the lockbox
    - Lockbox account ID – (optional, if known) the assigned account ID for the lockbox
    - Lockbox key "lockbox half" – (optional, if known and only key was just generated), this is server side base-64 encoded XORed portion of the lockbox key that must be combined with the client side portion of the key to create the full lockbox key. If specified the identity access token and proof must be specified.
    - Lockbox key "hash" – hash of the combined XORed lockbox key. If the lockbox key "lockbox half" is specified then this lockbox hash is reset to this hash specified otherwise this hash can be used to login to the lockbox account (by specifying the original identity with the "identity access token" and "proof" information being optional).

Returns:

- Lockbox information

    - Lockbox account ID – the assigned account ID for the lockbox
    - Lockbox access token – a verifiable token that is linked to the lockbox
    - Lockbox access secret – a secret that can be used in combination to the "lockbox access token" to provide proof of previous successful login
    - Lockbox access expiry – the window in which the access key is valid
    - Lockbox domain – the domain hosting the lockbox
    - Lockbox key "lockbox half" – this is server side base-64 encoded XORed portion of the lockbox key that must be combined with the client side portion of the key to create the full lockbox key.
    - Lockbox key "hash" – hash of the combined XORed lockbox key as previously passed in.
- Grant
    - ID – ID as passed into the request
    - List of namespace URLs previously granted to the grant ID
- List of identities attached to the lockbox
    - Disposition – (optional) if the "update" disposition is specified then this identity needs its lockbox key "identity half" updated as it is out of date
    - Original identity URI
    - Identity provider (optional, required if identity does not include domain or if domain providing identity service is different)

Security Considerations

Access to the lockbox does not grant access to the contents of the lockbox. The lockbox key must be obtained through an alternative method.

The server will validate the identity login via the identity service to access the account or validate the client has the correct lockbox key hash to access the account.

If the lockbox key "lockbox half" is specified because it was regenerated then all the information associated to the account is purged. The grants to the various namespaced values still remain valid but the data contained is completely wiped out. The remaining identities that were not used to login during the regeneration need to be marked as needing update since they will have the incorrect lockbox key "identity half".

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-access",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "identity": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,

      "uri": "identity://domain.com/alice",
      "provider": "domain.com"
```

```
    },

    "grant": {
      "$id": "de0c8c10d692bc91c1a551f57a50d2f97ef67543"
    },

    "agent": {
      "product": "hookflash/1.0.1001a (iOS/iPad)",
      "name": "hookflash",
      "image": "https://hookflash.com/brandsquare.png",
    },

    "lockbox": {
      "domain": "example.com",
      "key": "Wm1SellXWmtabVJoWm1wcmFuSmlhMnB5WW1WbWEycHlaV3ByY21ZPQ==",
      "hash": "cf69f9e4ed98bb739b4c72fc4fff403467014874"
    },

    "identities": {
     "identity": [
        {
          "uri": "identity://domain.com/alice",
          "provider": "domain.com"
        },
        {
          "uri": "identity:phone:16045551212",
          "provider": "example.com"
        }
      ]
    }
  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-access",
    "$timestamp": 439439493,

    "lockbox": {
      "$id": "123456",
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecret": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretExpires": 8483943493,

      "domain": "example.com",
      "key": "Wm1SellXWmtabVJoWm1wcmFuSmlhMnB5WW1WbWEycHlaV3ByY21ZPQ=="
    },

    "grant": {
      "$id": "de0c8c10d692bc91c1a551f57a50d2f97ef67543",

      "namespaces": {
        "namespace": [
          "https://domain.com/pemissionname",
          "https://other.com/pemissionname"
        ]
      }
    }
  }
}
```

Lockbox Access Validate Request

Purpose

This request proves that a lockbox access is valid and can be used to validate a lockbox access is successful by way of a 3rd party.

Inputs:

- Client nonce - a onetime use nonce, i.e. cryptographically random string
- Purpose – reason for validation (each service using this validation should have a unique purpose string)
- Lockbox information
    - Lockbox access token – a verifiable token that is linked to the lockbox
    - Proof of lockbox access secret' – proof required to validate that the lockbox access secret' is known, proof = hmac(<lockbox-access-secret>, "lockbox-access-validate:" + <client-nonce> + ":" + <expires> + ":" + <lockbox-access-token> + ":" + <purpose>)
    - Expiry of the proof for the 'lockbox access secret' – a window in which access secret Identity information

Returns:

Success or failure.

Security Considerations

Example

```
{
  "request": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-access-validate",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "purpose": "whatever",
    "lockbox": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934
    }
  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$method": "identity-access-validate",
    "$timestamp": 439439493
  }
}
```


Lockbox Identities Update Request

Purpose

This request updates the identities that are allowed to access the lockbox account.

Inputs:

- Client nonce - a onetime use nonce, i.e. cryptographically random string
- Lockbox information
    - Lockbox access token – a verifiable token that is linked to the lockbox
    - Proof of lockbox access secret' – proof required to validate that the lockbox access secret' is known, proof = hmac(<lockbox-access-secret>, "lockbox-access-validate:" + <client-nonce> + ":" + <expires> + ":" + <lockbox-access-token> + ":lockbox-identities-update")
    - Expiry of the proof for the 'lockbox access secret' – a window in which access secret proof is considered valid
- List of identities information
    - Disposition – "update" is used to add/update an identity and "remove" removes access to an identity
    - Identity access token – (optional, required if "update" is used), as returned from the "identity access complete" request
    - Proof of 'identity access secret' – (optional, required if "update" is used), proof required to validate that the 'identity access secret' is known, proof = hmac(<identity-access-secret>, "identity-access-validate:" + <identity> + ":" + <client-nonce> + ":" + <expires> + ":" + <identity-access-token> + ":lockbox-access-update")
    - Expiry of the proof for the 'identity access secret' – (optional, required if "update" is used) window in which access secret proof is considered valid
    - Original identity URI
    - Identity provider (optional, required if identity does not include domain or if domain providing identity service is different)

Returns:

- List of identities still attached to the lockbox
    - Original identity URI
    - Identity provider (optional, required if identity does not include domain or if domain providing identity service is different)

Security Considerations

If all the identities associated to the lockbox are removed then the lockbox account is considered deleted.

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abd23",
```



```
    "$handler": "lockbox",
    "$method": "lockbox-identities-update",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "lockbox": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,
    }

    "identities": {
     "identity": [
        {
          "$disposition": "update",

          "accessToken": "a913c2c3314ce71aee554986204a349b",
          "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
          "accessSecretProofExpires": 43843298934,

          "uri": "identity://domain.com/alice",
          "provider": "domain.com"
        },
        {
          "$disposition": "remove",

          "uri": "identity:phone:16135551212",
          "provider": "example.com"
        }
      ]
    }
  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-identities-update",
    "$timestamp": 439439493

    "identities": {
     "identity": [
        {
          "uri": "identity://domain.com/alice",
          "provider": "domain.com"
        },
        {
          "uri": "identity:phone:16045551212",
          "provider": "example.com"
        }
      ]
    }
  }
}
```

Lockbox Namespace Grant Inner Frame

Purpose

This inner frame is loaded from the outer application frame. The inner frame holds the namepace grant page for the lockbox. The inner/outer frames send information to/from each other via JavaScript posted messages. The inner page can display the lockbox's namespace page, allowing the user to grant permission to the application requesting access to the namespaces. The outer and inner page are

rendered inside a browser window and contains sufficient display size to allow the user to see what information is being granted.

Inputs:
None.

Returns:
None.

Security Considerations

Example

Lockbox Namespace Grant Window Notification

Purpose
This notification is sent from the inner frame to the outer window as a posted message. This allows the inner window to notify the outer window it's ready to start processing requests.

Inputs:
- Ready
    - true – notify the login window is ready to receive messages

Returns:
Success or failure.

Security Considerations

Example

```
{
  "notify": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-namespace-grant-window",

    "browser": {
      "ready": true
    }
  }
}
```

Lockbox Namespace Grant Notification

Purpose
Once the browser window receives notification that it is ready, this request is sent to the inner frame by the outer frame to give the inner frame the needed information to start the grant process.

Inputs:
- Agent

    - Product – the user agent identification for the product, typically "name/version (os/system)" information)
    - Name – a human readable friendly name for the product
    - Image – a human visual image for the brand that must be square in shape.
- Lockbox information
    - Lockbox access token – a verifiable token that is linked to the lockbox
    - Proof of lockbox access secret' – proof required to validate that the lockbox access secret' is known, proof = hmac(<lockbox-access-secret>, "lockbox-access-validate:" + <client-nonce> + ":" + <expires> + ":" + <lockbox-access-token> + ":lockbox-permission-grant")
    - Expiry of the proof for the 'lockbox access secret' – a window in which access secret proof is considered valid
- Grant
    - ID – ID as passed into the lockbox access request
    - List of namespace URLs to be granted to the grant ID

Returns:

None.

Security Considerations

Example

```
{
  "notify": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-namespace-grant",

    "agent": {
      "product": "hookflash/1.0.1001a (iOS/iPad)",
      "name": "hookflash",
      "image": "https://hookflash.com/brandsquare.png",
    },

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "lockbox": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,
    }

    "grant": {
      "$id": "de0c8c10d692bc91c1a551f57a50d2f97ef67543",

      "namespaces": {
        "namespace": [
          "https://domain.com/pemissionname",
          "https://other.com/pemissionname"
        ]
      }
    }
  }
}
```


Lockbox Namespace Grant Complete Notification

Purpose

This notification is sent from the inner browser window to the outer window as a posted message to indicate that the grant process has completed.

Inputs:

- Grant
  - ID – ID as passed into the lockbox grant request
  - List of namespace URLs granted to the grant ID

Returns:

Security Considerations

If permission was not granted to the namespace then the namespaces array will not contain any newly requested grants (but will contain any previously granted namespaces, if any).

Example

```
{
  "notify": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-namespace-grant-complete",

    "grant": {
      "$id": "de0c8c10d692bc91c1a551f57a50d2f97ef67543",

      "namespaces": {
        "namespace": [
          "https://domain.com/pemissionname",
          "https://other.com/pemissionname"
        ]
      }
    }
  }
}
```

Lockbox Content Get Request

Purpose

This request retrieves data contained in the lockbox.

Inputs:

- Client nonce - a onetime use nonce, i.e. cryptographically random string
- Lockbox information
  - Lockbox access token – a verifiable token that is linked to the lockbox
  - Proof of lockbox access secret' – proof required to validate that the lockbox access secret' is known, proof = hmac(<lockbox-access-secret>, "lockbox-access-validate:" + <client-nonce> + ":" + <expires> + ":" + <lockbox-access-token> + ":lockbox-get")

    - Expiry of the proof for the 'lockbox access secret' – a window in which access secret proof is considered valid
- Grant
    - ID – ID as passed into the lockbox grant request
- Content list of data elements containing:
    - Namespace URL – the amespace URL is the ID where the data is stored

Returns:

- Content list of data elements containing:
    - Namespace URL – the namespace URL is the ID where the data is stored
    - List of values, each value is base 64 encode with the value encrypted with: key = hmac(<lockbox-key>, "lockbox:" + <permission-url> + ":" + <value-name>), iv=hash(<permission-url> + ":" + <value-name>)

Security Considerations

No value names within the same namespace URL should be identical.

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-content-get",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "lockbox": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,
    }

    "grant": {
      "$id": "de0c8c10d692bc91c1a551f57a50d2f97ef67543"
    }

    "content": {
      "data": [
        {
          "$id": "https://domain.com/pemissionname"
        },
        {
          "$id": "https://other.com/pemissionname"
        }
      ]
    }

  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-content-get",
    "$timestamp": 439439493,

    "content": {
```

```
      "data": [
        {
          "$id": "https://domain.com/pemissionname",
          "value1": "ZmRzbmZranNkbmF...a2pkc2tqZnNkbmtkc2puZmRhZnNzDQo=",
          "value2": "Zmpza2xham...Zsa2RzamxmYXNmYXNzZmRzYWZk"
        },
        {
          "$id": "https://other.com/pemissionname",
          "what1": "ZmRzbmllZmJocmViaX...JmcXJicg0Kc2RmYQ0KZHNmYQ0Kcw0KZg==",
          "what2": "Wm1SemJtbG...ljZzBLYzJSbVlRMEtaSE5tWVEwS2N3MEtaZz09"
        }
      ]
    }

  }
}
```

Lockbox Content Set Request

Purpose

This request retrieves data contained in the lockbox.

Inputs:

- Client nonce - a onetime use nonce, i.e. cryptographically random string
- Lockbox information
    - Lockbox access token – a verifiable token that is linked to the lockbox
    - Proof of lockbox access secret' – proof required to validate that the lockbox access secret' is known, proof = hmac(<lockbox-access-secret>, "lockbox-access-validate:" + <client-nonce> + ":" + <expires> + ":" + <lockbox-access-token> + ":lockbox-get")
    - Expiry of the proof for the 'lockbox access secret' – a window in which access secret proof is considered valid
- Grant
    - ID – ID as passed into the lockbox grant request
- Content list of data elements containing:
    - Namespace URL – the namespace URL is the ID where the data is stored
    - List of values, each value is base 64 encode with the value encrypted with: key = hmac(<lockbox-key>, "lockbox:" + <permission-url> + ":" + <value-name>), iv=hash(<permission-url> + ":" + <value-name>), or a value of "-" to remove a value. The values are merged together with existing values or the values are removed if they contain a value of "-".

Returns:

Security Considerations

No value names within the same permission URL should be identical.

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abd23",
```



```
    "$handler": "lockbox",
    "$method": "lockbox-content-set",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "lockbox": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,
    }

    "grant": {
      "$id": "de0c8c10d692bc91c1a551f57a50d2f97ef67543"
    }

    "content": {
      "data": [
        {
          "$id": "https://domain.com/pemissionname",
          "value1": "ZmRzbmZranNkbmF...a2pkc2tqZnNkbmtkc2puZmRhZnNzDQo=",
          "value2": "Zmpza2xham...Zsa2RzamxmYXNmYXNzZmRzYWZk"
        },
        {
          "$id": "https://other.com/pemissionname",
          "what1": "ZmRzbmllZmJocmViaX...JmcXJicg0Kc2RmYQ0KZHNmYQ0Kcw0KZg==",
          "what2": "Wm1SemJtbG...ljZzBLYzJSbVlRMEtaSE5tWVEwS2N3MEtaZz09",
          "what3": "-"
        }
      ]
    }

  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-content-get",
    "$timestamp": 439439493
  }
}
```

Lockbox Admin Inner Frame

Purpose

This inner frame is loaded from the outer application frame. The inner frame holds the administration page for the lockbox. The inner/outer frames send information to/from each other via JavaScript posted messages. The inner page can display the lockbox's admin page, allowing the user to control access namespaces. The outer and inner page are rendered inside a browser window and contains sufficient display size to allow the user to administer the lockbox.

Inputs:

None.

Returns:

None.

Security Considerations

Example

Lockbox Admin Window Notification

Purpose

This notification is sent from the inner frame to the outer window as a posted message. This allows the inner window to notify the outer window it's ready to start processing requests.

Inputs:

- Ready
    - true – notify the login window is ready to receive messages

Returns:

Success or failure.

Security Considerations

Example

```
{
  "notify": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-admin-window",

    "browser": {
      "ready": true
    }
  }
}
```

Lockbox Admin Notification

Purpose

Once the browser window receives notification that it is ready, this request is sent to the inner frame by the outer frame to give the inner frame the needed information to start the administration.

Inputs:

- Agent
    - Product – the user agent identification for the product, typically "name/version (os/system)" information)
    - Name – a human readable friendly name for the product
    - Image – a human visual image for the brand that must be square in shape.
- Lockbox information
    - Lockbox access token – a verifiable token that is linked to the lockbox

- Proof of lockbox access secret' – proof required to validate that the lockbox access secret' is known, proof = hmac(<lockbox-access-secret>, "lockbox-access-validate:" + <client-nonce> + ":" + <expires> + ":" + <lockbox-access-token> + ":lockbox-admin")
- Expiry of the proof for the 'lockbox access secret' – a window in which access secret proof is considered valid

- Grant
  - ID – ID as passed into the lockbox access request

Returns:
None.

Security Considerations

Example

```
{
  "notify": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-admin",

    "agent": {
      "product": "hookflash/1.0.1001a (iOS/iPad)",
      "name": "hookflash",
      "image": "https://hookflash.com/brandsquare.png",
    },

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "lockbox": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,
    }

    "grant": {
      "$id": "de0c8c10d692bc91c1a551f57a50d2f97ef67543"
    }
  }
}
```

Lockbox Admin Complete Notification

Purpose
This notification is sent from the inner browser window to the outer window as a posted message to indicate that the administration process has completed.

Inputs:

Returns:

Security Considerations

Example

```
{
  "notify": {
```

```
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-admin-complete"
  }
}
```

Lockbox Namespace Grant Request

Purpose

This request allows a client to automatically request additional grants to namespaces from a pre-authorized source.

Inputs:

- Agent
  - Product – the user agent identification for the product, typically "name/version (os/system)" information)
  - Name – a human readable friendly name for the product
  - Image – a human visual image for the brand that must be square in shape.
- Lockbox information
  - Lockbox access token – a verifiable token that is linked to the lockbox
  - Proof of lockbox access secret' – proof required to validate that the lockbox access secret' is known, proof = hmac(<lockbox-access-secret>, "lockbox-access-validate:" + <client-nonce> + ":" + <expires> + ":" + <lockbox-access-token> + ":lockbox-permission-grant")
  - Expiry of the proof for the 'lockbox access secret' – a window in which access secret proof is considered valid
- Grant
  - ID – ID as passed into the lockbox access request
- Authorization bundle
  - Proof – hmac(<grant-id>, "grant-proof:" + <client-nonce>)
  - Expires – timestamp when the proof bundle expires
  - List of namespace URLs to be granted to the grant ID
  - Signature from authorizing domain

Returns:

Success or failure.

Security Considerations

The domain signing proof can authorize namespaces within its own domain and no others unless the lockbox has special disposition for domain to authorize other namespaces.

Example

```
{
  "request": {
    "$domain": "provider.com",
    "$id": "abd23",
```

```
    "$handler": "lockbox",
    "$method": "lockbox-namespace-grant",

    "agent": {
      "product": "hookflash/1.0.1001a (iOS/iPad)",
      "name": "hookflash",
      "image": "https://hookflash.com/brandsquare.png",
    },

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "lockbox": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,
    }

    "grant": {
      "$id": "de0c8c10d692bc91c1a551f57a50d2f97ef67543"
    },

    "authorizationBundle": {
      "authorization": {
        "proof": "bdd7cf06e641ee5be2f28bc051201565f05ef15e",
        "expires": 573454,
        "namespaces": {
          "namespace": [
            "https://provider.com/pemissionname",
            "https://provider.com/pemissionname"
          ]
        }
      },
      "signature": {
        "reference": "#b5dfaf2d00ca5ef3ed1a2aa7ec23c2db",
        "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
        "digestValue": "IUe324k...oV5/A8Q38Gj45i4jddX=",
        "digestSigned": "MDAwMDAw...MGJ5dGVzLiBQbGVhc2UsIGQ=",
        "key": {
          "$id": "b7ef37...4a0d58628d3",
          "domain": "provider.com",
          "service": "identity"
        }
      }
    }
  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "lockbox",
    "$method": "lockbox-namespace-grant",
    "$timestamp": 439439493
  }
}
```


Identity Lookup Service Requests

The communication to the Identity Lookup is done over HTTPS exclusively whose HTTPS server certificate was signed by one of the trusted root Internet certification authorities.

Identity Lookup Check Request

Purpose

This request checks to see when the identity information last changed for particular identities and also determines which identities have contact information. Request will only return identities that resolve and they are returned in the same order they were requested.

Inputs:

List of providers containing:

- Identity lookup base URI
- Separator (default is ",")
- List of:
    - Identities separated by "separator"

Returns:

List of resulting identities that resolve in the order requested as follows:

- Original identity URI
- Provider – service responsible for this identity
- Last update timestamp – when the information associated to the identity was last updated

Security Considerations

Example

```
{
  "request": {
    "$domain": "test.com",
    "$id": "abd23",
    "$handler": "identity-lookup",
    "$method": "identity-lookup-check",

    "providers": {
      "provider": [
        {
          "base": "identity://domain.com/",
          "separator": ",",
          "identities": "alice,bob,fred"
        },
        {
          "base": "identity:phone:",
          "separator": ";",
          "identities": "16045551212;3814445551212"
        },
        {...}
      ]
```

```
    }
  }
}
```

```
{
  "result": {
    "$domain": "test.com",
    "$id": "abc123",
    "$handler": "identity-lookup",
    "$method": "identity-lookup-check",
    "$timestamp": 439439493,

    "identities": {
      "identity": [
        {
          "uri": "identity://domain.com/alice",
          "updated": 5949594
        },
        {...},
        {
          "uri": "identity:phone:16045551212",
          "provider": "example.com",
          "updated": 5849594
        },
        {...},
        {
          "uri": "identity:email:alice@domain.com",
          "provider": "provider.com",
          "updated": 574443
        }
      ]
    }
  }
}
```

Identity Lookup Request

Purpose

This request resolves the identities to Peer Contacts. Request will only return identities that resolve and they are returned in the same order they were requested.

Inputs:

List of providers containing:

- Identity lookup base URI
- Separator (default is ",")
- List of:
    - Identities separated by "separator"

Returns:

List of resulting identities that resolve in the order requested as follows:

- Original identity URI
- Provider – service responsible for this identity
- Stable ID – a stable ID representing the user regardless of which identity is being used or the current peer contact ID

- Public peer file
- TTL expiry timestamp - when must client do a recheck on the identity as the associated information might have changed
- Priority / weight – SRV like priority and weighting system to gauge which identity discovered to be associated to the same peer contact have highest priority
- Last update timestamp – when the information associated to the identity was last updated
- Identity display name – (optional), the display name to use with the identity
- Identity rendered public profile URL – (optional), a webpage that can be rendered by the browser to display profile information about this identity
- Programmatic public profile URL – (optional), a machine readable vcard like webpage that can be used to extract out common profile information
- Public Feed URL – (optional), an RSS style feed representing the public activity for the user
- Optional list of avatars containing:
    - Avatar name – (optional), name representing subject name of avatar (note: avatars with the same name are considered identical and thus are used to distinguish between varying sizes for the same avatar)
    - Avatar URL – URLs to download the avatar(s) associated with the identity
    - Avatar pixel width – (optional), pixel width of the avatar image
    - Avatar pixel height – (optional), pixel height of avatar image

Security Considerations

Example

```
{
  "request": {
    "$domain": "test.com",
    "$id": "abd23",
    "$handler": "identity-lookup",
    "$method": "identity-lookup",

    "providers": {
      "provider": [
        {
          "base": "identity://domain.com/",
          "separator": ",",
          "identities": "alice,bob,fred"
        },
        {
          "base": "identity:phone:",
          "separator": ";",
          "identities": "16045551212;3814445551212"
        },
        {...}
      ]
    }
  }
}
```

```
{
  "result": {
    "$domain": "test.com",
    "$id": "abc123",
    "$handler": "identity-lookup",
    "$method": "identity-lookup",
    "$timestamp": 439439493,
```



```
    "identities": {
      "identity": [
        {
          "uri": "identity://domain.com/alice",
          "provider": "domain.com",
          "stableID": "123456",
          "peer": {...},
          "priority": 5,
          "weight": 1,
          "updated": 5949594,
          "expires": 58843493,
          "name": "Alice Applegate",
          "profile": "http://domain.com/user/alice/profile",
          "vprofile": "http://domain.com/user/alice/vcard",
          "feed": "http://domain.com/user/alice/feed",
          "avatars": {
            "avatar": { "url": "http://domain.com/user/alice/p" }
          }
        },
        {...},
        {
          "uri": "identity:phone:16045551212",
          "provider": "example.com",
          "stableID": "123456",
          "peer": {...},
          "priority": 1,
          "weight": 1,
          "updated": 5849594,
          "expires": 58843493
        },
        {...},
        {
          "uri": "identity:email:alice@domain.com",
          "provider": "provider.com",
          "stableID": "78312",
          "peer": {...},
          "priority": 1,
          "weight": 1,
          "updated": 574443,
          "expires": 58843493,
          "profile": "http://www.gravatar.com/205e460b479e2e5b48aec07710c08d50",
          "avatars": {
            "avatar": [
              {
                "name": "1",
                "url": "http://www.gravatar.com/avatar/205e460b479e2e5b48aec07710c08d50?size=1",
                "width": 100,
                "height": 100
              },
              {
                "name": "1",
                "url": "http://www.gravatar.com/avatar/205e460b479e2e5b48aec07710c08d50?size=2",
                "width": 200,
                "height": 200
              }
            ]
          }
        }
      ]
    }
  }
}
```

Identity Service Requests

Identity Access Inner Frame (webpage)

Purpose

This inner frame is loaded from the outer application frame. The inner frame holds the login page for the identity. The inner/outer frames send information to/from each other via JavaScript posted messages. The inner page can display the identity provider's login page allowing the user to enter their identity credentials. The outer and inner page are rendered inside a browser window and contains sufficient display size to allow an identity provider to enter their credential information although the web view might start hidden to allow for auto-relogin (in which case there will be no rendered page for entering credential information).

Inputs:

None.

Returns:

None.

Security Considerations

Example

Identity Access Window Notification

Purpose

This notification is sent from the inner frame to the outer frame as a posted message. This allows the inner window to notify the outer browser window that visibility is needed and/or if it's ready to start processing requests. Upon loading the inner frame must send to the outer frame that it is ready to start processing messaging.

Inputs:

- Ready
    - true – notify the login window is ready to receive messages
- Visibility:
    - true – notify the login window needs visibility

Returns:

Success or failure.

Security Considerations

This notification is allowed to be sent more than once to the outer frame as needed.

If the inner frame is being reloaded after having been replaced during the login process, the inner frame dies not need to resend this notification as it can immediately send the "Identity Access Complete" notification.

Example

```
{
  "notify": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "identity",
    "$method": "identity-access-window",

    "browser": {
      "ready": true,
      "visibility": true,
    }
  }
}
```

Identity Access Start Notification

Purpose

Once the browser window receives notification that it is ready, this request is sent to the inner frame by the outer frame to give the inner frame the needed information to start the login process.

Inputs:

- Agent
  - Product – the user agent identification for the product, typically "name/version (os/system)" information)
  - Name – a human readable friendly name for the product
  - Image – a human visual image for the brand that must be square in shape.
- Base identity URI – base URI for identity (or full identity if known in advance)
- Browser information
  - Visibility – the browser window is being shown in what state
    - "visible" – the browser window is visible
    - "hidden" – the browser window is hidden and cannot be shown
    - "visible-on-demand" – the browser window is hidden but can be rendered visible via a request posted to the outer frame (note: if rendered inside an application, the application can show the window in a hidden state to start and the browser window can become visible only when the user needs to enter some credentials)
  - Popup
    - "allow" – popups windows/new tabs are allowed to be opened
    - "deny" – popup windows/new tables are not allowed to be opened
  - Outer frame reload URL – a URL to reload the outer frame should the login process have to replace the outer frame's window with its own URL. Once the outer frame is reloaded

the inner frame page is reloaded as well allowing the inner frame to send the completion request.

Returns:

None.

Security Considerations

If the full URI of the identity is specified, the client should attempt to relogin automatically to the identity (if possible).

If the outer fame is being reloaded after haven been replaced, this notification should not be sent again.

Once the inner frame receives this notification it is allowed to replace the outer frame with its own page but it must bring back the outer page so the login process can be completed.

Example

```
{
  "notify": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "identity",
    "$method": "identity-access-start",

    "agent": {
      "product": "hookflash/1.0.1001a (iOS/iPad)",
      "name": "hookflash",
      "image": "https://hookflash.com/brandsquare.png",
    },

    "identity": {
      "base": "identity://provider.com/"
    },

    "browser": {
      "visibility": "visible-on-demand",
      "popup": "deny",
      "outerFrameURL": "https://webapp.com/outerframe?reload=true"
    }
  }
}
```

Identity Access Complete Notification

Purpose

This notification is sent from the inner browser window to the outer window as a posted message to indicate that the login process has completed.

Inputs:

- Identity information
    - Identity URI – the full identity URI of the logged in user
    - Identity provider – identity provider providing identity service
    - Identity access token – a verifiable token that is linked to the logged-in identity
    - Identity access secret – a secret that can be used in combination to the "identity access token" to provide proof of previous successful login

    - Identity access expiry – the window in which the access key is valid
- Lock box information (optional, if known)
    - Lockbox domain – if lockbox domain is known in advance, this is the domain for the lockbox to use
    - Lockbox key "identity half" – this is client side base-64 encoded XORed portion of the lockbox key that must be combined with the server side portion of the key to create the full lockbox key.
    - Lockbox reset flag – this flag is used if the lockbox must be reset with a new password and all data within to be flushed.

Returns:

Security Considerations

The lockbox key should be decrypted locally in the JavaScript using something unavailable in the server, for example the user's password. Other information should be combined to create the encryption/decryption key to ensure two unique users with the same password do not share the same encryption key.

An example formula might look like:

lockbox-key = decrypt(<lockbox-key-encrypted>, iv), where key= hmac(key_strectch(<user-password>), <user-id>), iv=hash(<user-salt>)

Key stretching should be employed whenever using a weaker user generated non-cryptographically strong password. See: http://en.wikipedia.org/wiki/Key_stretching

By using information not stored on a server, this ensures that should the server be hacked that the servers do not contain the correct information to decrypt the lockbox key. The downside is that should the password change the encryption key will need to be decrypted with the existing user password then re-encrypted using the new password. Further, if the old password is lost then the lockbox key is also lost.

Example

```
{
  "notify": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "identity",
    "$method": "identity-access-complete",

    "identity": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecret": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretExpires": 8483943493,

      "uri": "identity://domain.com/alice",
      "provider": "domain.com"
    },

    "lockbox": {
      "domain": "domain.com",
      "key": "V20x...IbGFWM0J5WTIxWlBRPT0=",
```

```
      "reset": false
    }
  }
}
```

Identity Access Lockbox Update Request

Purpose

This request is sent from the outer browser window to the inner window as a posted message to indicate that the login process has completed.

Inputs:

- Identity information
  - Client one time use nonce (cryptographically random string)
  - Identity access token – as returned from the "identity access complete" request
  - Proof of 'identity access secret' – proof required to validate that the 'identity access secret' is known, proof = hmac(<identity-access-secret>, "identity-access-validate:" + <identity> + ":" + <client-nonce> + ":" + <expires> + ":" + <identity-access-token> + ":lockbox-update")
  - Expiry of the proof for the 'identity access secret' – a window in which access secret proof is considered valid
- Lock box information
  - Lockbox domain – if lockbox domain is known in advance, this is the domain for the lockbox to use
  - Lockbox key "identity half" – this is client side base-64 encoded XORed portion of the lockbox key that must be combined with the server side portion of the key to create the full lockbox key.

Returns:

Success or failure.

Security Considerations

The lockbox key should be encrypted locally in JavaScript before being sent a server. This ensures the server does not contain the correct information to be able to decrypt the lockbox key. See "Identity Access Complete"

Example

```
{
  "request": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "identity",
    "$method": "identity-access-lockbox-update",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "identity": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
```

```
      "accessSecretProofExpires": 43843298934
    },
    "lockbox": {
      "domain": "domain.com",
      "key": "V20x...IbGFWM0J5WTIxWlBRPT0="
    }
  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$method": "identity-access-lockbox-update",
    "$timestamp": 439439493
  }
}
```

Identity Access Validate Request

Purpose

This request proves that an identity access is valid and can be used to validate an identity access is successful by way of a 3rd party.

Inputs:

- Client nonce - a onetime use nonce, i.e. cryptographically random string
- Purpose – reason for validation (each service using this validation should have a unique purpose string)
- Identity information
    - Identity access token – as returned from the "identity access complete" request
    - Proof of 'identity access secret' – proof required to validate that the 'identity access secret' is known, proof = hmac(<identity-access-secret>, "identity-access-validate:" + <identity> + ":" + <client-nonce> + ":" + <expires> + ":" + <identity-access-token> + ":" + <purpose>)
    - Expiry of the proof for the 'identity access secret' – a window in which access secret proof is considered valid
    - Original identity URI
    - Identity provider (optional, required if identity does not include domain or if domain providing identity service is different)

Returns:

Success or failure.

Security Considerations

Example

```
{
  "request": {
    "$domain": "provider.com",
    "$id": "abd23",
```

```
    "$handler": "identity",
    "$method": "identity-access-validate",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "purpose": "whatever",
    "identity": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,

      "uri": "identity://domain.com/alice"
    }
  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$method": "identity-access-validate",
    "$timestamp": 439439493
  }
}
```

Identity Lookup Update Request

Purpose

This request proves that an identity login is valid and can be used to validate an identity login is successful by way of a 3rd party.

Inputs:

- Client one time use nonce (cryptographically random string)
- Identity access token – as returned from the "identity access complete" request
- Proof of 'identity access secret' – proof required to validate that the 'identity access secret' is known, proof = hmac(<identity-access-secret>, "identity-access-validate:" + <identity> + ":" + <client-nonce> + ":" + <expires> + ":" + <identity-access-token> + ":identity-lookup-update")
- Expiry of the proof for the 'identity access secret' – a window in which access secret proof is considered valid
- Stable ID – a stable ID representing the user regardless of which identity is being used or the current peer contact ID
- Public peer file– the public peer file associated with the contact ID
- Priority / weight – SRV like priority and weighting system to gauge which identity discovered to be associated to the same peer contact have highest priority

Returns:

Success or failure.

Security Considerations

Example

```
{
  "request": {
    "$domain": "provider.com",
```

```
    "$id": "abd23",
    "$handler": "identity",
    "$method": "identity-lookup-update",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "identity": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,

      "stableID": "0acc990c7b6e7d5cb9a3183d432e37776fb182bf",
      "peer": {...},
      "priority": 5,
      "weight": 1
    }
  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$method": "identity-lookup-update",
    "$timestamp": 439439493
  }
}
```

Identity Sign Request

Purpose

This request requests a signed identity a given proof of a successful identity login.

Inputs:

- Original identity
- Client one time use nonce (cryptographically random string)
- Identity access token – as returned from the "identity access complete" request
- Proof of 'identity access secret' – proof required to validate that the 'identity access secret' is known, proof = hmac(<identity-access-secret>, "identity-sign:" + <identity> + ":" + <client-nonce> + ":" + <expires> + ":" + <identity-access-token>)
- Expiry of the proof for the 'identity access secret' – a window in which access secret proof is considered valid

Returns:

Signed identity bundle by the identity service.

Security Considerations

Example

```
{
  "request": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "identity",
    "$method": "identity-sign",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "identity": {
```

```
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934,

      "uri": "identity://domain.com/alice"
    }
  }
}
```

```
{
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "identity",
    "$method": "identity-sign",
    "$timestamp": 439439493,

    "identityBundle": {
      "identity": {
        "$id": "b5dfaf2d00ca5ef3ed1a2aa7ec23c2db",
        "contact": "peer://example.com/ab43bd44390dabc329192a392bef1",
        "uri": "identity://domain.com/alice",
        "created": 54593943,
        "expires": 65439343
      },
      "signature": {
        "reference": "#b5dfaf2d00ca5ef3ed1a2aa7ec23c2db",
        "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
        "digestValue": "IUe324k...oV5/A8Q38Gj45i4jddX=",
        "digestSigned": "MDAwMDAw...MGJ5dGVzLiBQbGVhc2UsIGQ=",
        "key": {
          "$id": "b7ef37...4a0d58628d3",
          "domain": "provider.com",
          "service": "identity"
        }
      }
    }
  }
}
```

Peer Service

Peer Services Get Request

Purpose

This request retrieves gets a list of peer contact services available to the peer contact.

Inputs:

- Client nonce - a onetime use nonce, i.e. cryptographically random string
- Lockbox information
    - Lockbox access token – a verifiable token that is linked to the lockbox
    - Proof of lockbox access secret' – proof required to validate that the lockbox access secret' is known, proof = hmac(<lockbox-access-secret>, "lockbox-access-validate:" + <client-nonce> + ":" + <expires> + ":" + <lockbox-access-token> + ":peer-services-get")
    - Expiry of the proof for the 'lockbox access secret' – a window in which access secret proof is considered valid

Returns:

List of services available to peer contact services, containing:

- Service type
- Version
- Expires – when the service must be refreshed because it's no longer considered valid
- List of requests methods, which includes
    - Method name
    - URI
    - Other request specific information

Security Considerations

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abd23",
    "$handler": "peer",
    "$method": "peer-services-get",

    "clientNonce": "ed585021eec72de8634ed1a5e24c66c2",
    "lockbox": {
      "accessToken": "a913c2c3314ce71aee554986204a349b",
      "accessSecretProof": "b7277a5e49b3f5ffa9a8cb1feb86125f75511988",
      "accessSecretProofExpires": 43843298934
    }

  }
}
```

```
{
```



```
  "result": {
    "$domain": "provider.com",
    "$id": "abd23",
    "$handler": "peer",
    "$method": "peer-services-get",
    "$timestamp": 439439493,

    "services": {
      "service": [
        {
          "$id": "4e6a9ca60d92dfa5e872537f408066d02bdb55f2",
          "type": "turn",
          "version": "RFC5766",
          "expires": 493943,
          "methods": {
            "method": {
              "name": "turn",
              "uri": "service.com"
              "username": "id39392",
              "password": "bdaaba26fa8ccac7807a786156b1f0fc87b2e28a"
            }
          }
        },
        {
          "$id": "5e6a9ca60d92dfa5e872537f408066d02bdb55f8",
          "type": "stun",
          "version": "RFC5389",
          "expires": 493943,
          "methods": {
            "method": {
              "name": "stun",
              "uri": "service.com"
              "username": "id39392",
              "password": "bdaaba26fa8ccac7807a786156b1f0fc87b2e28a"
            }
          }
        }
      ]
    }
  }
}
```


Peer Salt Service Protocol

Signed Salt Get Request

Purpose

This request returns random salt as derived from a server and signed to prove authenticity of the salt.

Inputs:

- Number of signed salts

Returns:

- Total number of salts, where each salt has
  - Salt id – each salt is given a unique ID within the system
  - Salt – base 64 encoded cryptographically random binary salt
  - Signed by salt service's private key

Return one or more signed salt blobs for use in the peer files.

Security Considerations

The client should verify signature was generated by the certificate was issued by the salt service if the same domain.

Example

```
{
  "request": {
    "$domain": "example.com",
    "$id": "abd23",
    "$handler": "peer-salt",
    "$method": "signed-salt-get",

    "salts": 2
  }
}
```

```
{
  "result": {
    "$id": "abc123",
    "$handler": "peer-salt",
    "$method": "signed-salt-get",
    "$timestamp": 439439493,

    "salts": {
      "saltBundle": [
        {
          "salt": {
            "$id": "f2e2ba4ba900e3b78d0d8524f0888f2b57d1bf91",
            "#text": "fdjfdsE2443lfkXEnek..343o="
          },
          "signature": {
            "reference": "#f2e2ba4ba900e3b78d0d8524f0888f2b57d1bf91",
            "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
            "digestValue": "IUe324ko...V5/A8Q38Gj45i4jddX=",
            "digestSigned": "DEf...GM~C0/Ez=",
            "key": {
```

```
              "$id": "db144bb314103033cbba7d52e",
              "domain": "example.com",
              "service": "salt"
            }
          }
        },
        {
          "salt": {
            "$id": "35959a33d4eafac97b9d068cba32a8c6c6fd463a",
            "#text": "prfd+dsE243lfkXEnek..8rz=="
          },
          "signature": {
            "reference": "#35959a33d4eafac97b9d068cba32a8c6c6fd463a",
            "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
            "digestValue": "4Jee4kfd/oV5Af8Q3g48Gjg4n5iw4jzd8k=",
            "digestSigned": "Ttzf...eky5D/GM~C0=",
            "key": {
              "$id": "db144bb314103033cbba7d52e",
              "domain": "example.com",
              "service": "salt"
            }
          }
        }
      ]
    }
  }
}
```

Peer Common Protocol

Peer Publish Request

Purpose

This method allows a peer to publish a document into the network where it can be subscribed to by other peers or groups.

Inputs

- Document name – full path, including namespace
- Document version – first version must start at 1 and all others must be +1 from previous
- Document base version – (optional), must be present if sending an "json-diff" document; the base version must be included to know what the base version was used to compute the differences. This base version must match the last published version by the receiver of the publish request or a 409 conflict will be returned.
- Document lineage – for v1 documents, it is recommend to use the epoch but the server may replace with its own value in the result (which must be used in subsequent updates); the lineage must match between updates to the same document; the lineage value is to prevent conflicts when performing document deletions
- Chunk number – (optional), "1/1" is assumed – to allow upload of multiple chunks of the document (note: not all documents support chunking)
- Scope of where the document resides – associated to "location", or "contact"; the "location" scope is a private namespace only writable to the current session location; the "contact" scope is a namespace shared by all locations for the same contact (or through a special processer is shared amongst all users
- Lifetime of document – i.e. "session" or "permanent"
- Expiry of document – (optional), epoch of when document must expire
- Encoding – (optional), "json" is assumed, options are "json", "binary-base64"
- "Publish to relationships" – list of:
    - Relationships contact file name (e.g. "/hookflash.com/authorization-list/1.0/whitelist" or "/hookflash.com/authorization-list/1.0/adhoc-subscribers"). The scope for relationship documents is always "location". However, specialized document processors can generate "on-the-fly" relationship lists.
    - Permission – one of:
        - "all" – allow all users on the list to subscribe, fetch and receive notification about this document;
        - "none" – do not allow any users on the list to fetch and receive notification about this document;
        - "some" – allow only specific users listed within the relationship list to subscribe and receive notification about this document;

Outputs

The document meta information as passed into the publish without the data itself, including an updated lineage should the server replace the lineage for version 1 documents.

Security Considerations

The lineage value can only be changed and must be changed after a previous document of the same name has been deleted. The server must verify the lineage is identical to the current lineage for versions other than version 1 chunk 1. For version 1 chunk 1 documents, the server can replace the proposed lineage with its own value that must be used in subsequent updates of the document. The lineage value must increase in value from the previous value when the lineage changes.

Clients and servers should consider the document "newer" if it has a greater lineage value regardless if the version number is smaller.

The server must delete the document at the end of the lifetime.

When the client is delivering multiple chunks, the chunks must arrive in sequence. Any other document requests mid update will fail with error code 409 until the entire document has been delivered.

If using the json or json-diff scheme, the document chunks are post-pended together into a single document until all chunks are delivered and then processed as a whole. If using binary-base64 then the documents are merged together with white space removed and then decoded to binary when all chunks have arrived.

The "all" or "some" permissions for relationships that allow a contact to receive the published documents takes precedence over the "none" or implied "deny" if they the listed contact is not contained within the "some" list.

Example

```
{
  "request": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-publish",
    "document": {
      "details": {
        "name": "/hookflash.com/presence/1.0/bd520f1...c0cc9b7ff528e83470e/883fa7...9533609131",
        "version": 12,
        "baseVersion": 10,
        "lineage": 5849943,
        "chunk": "1/12",
        "scope": "location",
        "contact": "peer://example.com/97a9f246018a491d14cee267dceedf8a4ce0367c",
        "location": "5cd9f6d9bff930edcc590a62625ba7695b4f805e",
        "lifetime": "session",
        "expires": 447837433,
        "mime": "text/json",
        "encoding": "json"
      },
      "publishToRelationships": {
        "relationships": [
          {
            "$name": "/hookflash.com/authorization-list/1.0/whitelist",
            "$allow": "all"
          },
```

```
          {
            "$name": "/hookflash.com/authorization-list/1.0/adhoc",
            "$allow": "all"
          },
          {
            "$name": "/hookflash.com/shared-groups/1.0/foobar",
            "$allow": "all"
          }
        ]
      },
      "data": {...}
    }
  }
}
```

```
{
  "result": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-publish",
    "$timestamp": 13494934,

    "document": {
      "details": {
        "name": "/hookflash.com/presence/1.0/bd520f1d...cc9b7ff528e83470e/883fa7...9533609131",
        "version": 12,
        "lineage": 5849943,
        "chunk": "1/12",
        "scope": "location",
        "lifetime": "session",
        "expires": 4839543,
        "mime": "text/json",
        "encoding": "json"
      },
      "publishToRelationships": {
        "relationships": [
          {
            "$name": "/hookflash.com/authorization-list/1.0/whitelist",
            "$allow": "all"
          },
          {
            "$name": "/hookflash.com/authorization-list/1.0/adhoc",
            "$allow": "all"
          },
          {
            "$name": "/hookflash.com/shared-groups/1.0/foobar",
            "$allow": "all"
          }
        ]
      }
    }
  }
}
```

Peer Get Request

Purpose

This method allows a peer to fetch a previously publish a document from the network.

Inputs

- Document name – full path, including namespace
- Document version in cache – (optional), if available

- Document lineage in cache – (optional), if available
- Scope of where the document resides – associated to "location", or "contact"; the "location" scope is a private namespace only readable from the current session location; the "contact" scope is a namespace shared by all locations for the same contact; the "global" namespace is shared by all contacts on the system globally
- Contact from which to load the document – (optional), if "location" or "contact" is used
- Location ID from which to load the document – (optional), if "location" is used
- Chunk number – (optional), "1/1" is assumed; to allow upload of multiple chunks of the document, the server may decide to split the result into multiple chunks for easier transport. The client should respect the server's splitting and use this value instead of its own "1/x" value.

Outputs

Previously published document split into chunks when appropriate.

Security Considerations

The server must ensure the document is published to the contact that is requesting the document otherwise the server must return a 403 Forbidden.

The server can return any version and lineage greater than the cached version. If the latest version is equal to the cached version, the document result will contain the document meta information without the data.

The server will return any version number it choses equal or greater to the request version number but must adhere to the "json-diff" mechanism and give only the differences between the versions or "json" to give the latest version only without performing the differences.

The server will ignore the chunking denominator for chunk requested for the "1/1" chunk and require the denominator to be a value it expects instead for all other chunks requested.

If using the "json" or "json-diff" scheme, the document chunks are post-pended together into a single document until all chunks are delivered and then processed as a whole. If using binary-base64 then the documents are merged together with white space removed and then decoded to binary when all chunks have arrived.

The "publish to relationships" section is only returned if the contact requesting the document is the publisher of the document.

Example

```
{
  "request": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-get",

    "document": {
      "details": {
        "name": "/hookflash.com/presence/1.0/bd520f1dbaa...9b7ff528e83470e/883fa7...9533609131",
        "version": 12,
        "lineage": 39239392,
```

```
        "scope": "location",
        "contact": "peer://example.ecom/ea00ede4405c99be9ae45739ebfe57d5",
        "location": "524e609f337663bdbf54f7ef47d23ca9",
        "chunk": "1/1"
      }
    }
  }
}
```

```
{
  "result": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-get",
    "$timestamp": 13494934,

    "document": {
      "details": {
        "name": "/hookflash.com/presence/1.0/bd520f1...c0cc9b7ff528e83470e/883fa7...9533609131",
        "version": 12,
        "lineage": 39239392,
        "chunk": "1/10",
        "scope": "location",
        "contact": "peer://example.com/ea00ede4405c99be9ae45739ebfe57d5",
        "location": "524e609f337663bdbf54f7ef47d23ca9",
        "lifetime": "session",
        "expires": 45747885743,
        "mime": "text/json",
        "encoding": "json"
      },
      "publishToRelationships": {
        "relationships": [
          {
            "$name": "/hookflash.com/authorization-list/1.0/whitelist",
            "$allow": "all"
          },
          {
            "$name": "/hookflash.com/authorization-list/1.0/adhoc",
            "$allow": "all"
          },
          {
            "$name": "/hookflash.com/shared-groups/1.0/foobar",
            "$allow": "all"
          }
        ]
      },
      "data": "..."
    }
  }
}
```

Peer Delete Request

Purpose

This method allows a peer delete a previously publish a document from the network.

Inputs

- Document name – full path, including namespace
- Document version – (optional), if specified the version number must match the last published version number or a conflict is returned
- Document lineage – (optional), if specified the lineage number must match the lineage of the last published version of the document or a conflict is returned

- Scope of where the document resides – associated to "location", or "contact"; the "location" scope is a private namespace only readable from the current session location; the "contact" scope is a namespace shared by all locations for the same contact; the "global" namespace is shared by all contacts on the system globally

Outputs

Success or failure.

Security Considerations

The contact owner of the document is the only entity allowed to delete the document.

If the document version or lineage is specified then the document version and lineage must match the last published version or the request is rejected with a 409 Conflict error.

Example

```
{
  "request": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-delete",

    "document": {
      "details": {
        "name": "/hookflash.com/presence/1.0/bd520f1db...b7ff528e83470e/883fa7...9533609131",
        "version": 12,
        "lineage": 39239392,
        "scope": "location"
      }
    }
  }
}
```

```
{
  "result": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-delete",
    "$timestamp": 13494934
  }
}
```

Peer Subscribe Request

Purpose

This method allows a peer to subscribe to all documents it is authorized to fetch within a namespace and within its relationships.

Inputs

- Documents base path/name – full path base to monitor with optional "*" for partial paths
- Relationships to subscribe, containing:
    - Name of relationships document
    - Which contacts within the relationships to subscribe
        - "all" – subscribe to all relationships

- "none" – remove all subscriptions to any relationship
- "some" – change relationships to subscribe to the listed contacts
- "add" – add some contacts to subscribe within the relationships
- "remove" – remove some contacts to subscribe with the relationships

Outputs

Returns the resulting merged active subscription.

Security Considerations

The server only allows subscriptions where permissions allow.

Example

```
{
  "request": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-subscribe",

    "document": {
      "name": "/hookflash.com/presence/1.0/",
      "subscribeToRelationships": {
        "relationships": [
          {
            "$name": "/hookflash.com/authorization-list/1.0/whitelist",
            "$subscribe": "all"
          },
          {
            "$name": "/hookflash.com/authorization-list/1.0/adhoc",
            "$subscribe": "add",
            "contact": "peer://example.com/bd520f1dbaa13c0cc9b7ff528e83470e"
          },
          {
            "$name": "/hookflash.com/shared-groups/1.0/foobar",
            "$subscribe": "all"
          }
        ]
      }
    }
  }
}
```

```
{
  "result": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-subscribe",
    "$timestamp": 13494934,

    "document": {
      "name": "/hookflash.com/presence/1.0/",
      "subscribeToRelationships": {
        "relationships": [
          {
            "$name": "/hookflash.com/authorization-list/1.0/whitelist",
            "$subscribe": "all"
          },
          {
            "$name": "/hookflash.com/authorization-list/1.0/adhoc",
            "$subscribe": "some",
            "contact": [
              "peer://example.com/bd520f1dbaa13c0cc9b7ff528e83470e",
              "peer://example.com/8d17a88e8d42ffbd138f3895ec45375c"
            ]
          },
```

```
            {
              "$name": "/hookflash.com/shared-groups/1.0/foobar",
              "$subscribe": "all"
            }
          ]
        }
      }
    }
  }
```

Peer Publish Notify

Purpose

This method notifies a peer that a document has been updated.

Inputs

- Document name – full path, including namespace
- Document version – if "0" then the document for the lineage is deleted
- Document lineage
- Scope of where the document resides – associated to "location", or "contact" or "global"; the "location" scope is a private namespace only writable to the current session location; the "contact" scope is a namespace shared by all locations for the same contact; the "global" namespace is shared by all contacts on the system globally
- Contact id – the contact that published the document
- Location id – the location where the document was published
- Lifetime of document – i.e. "session" or "permanent"
- Expiry of document – (optional)
- Data – (optional), at the discretion of the server, the document can be delivered as part of the notify or held back which will require the client to fetch later if the client wishes to update manually

Outputs

None.

Security Considerations

If a client wants to download a document about which notification was received, a client should attempt to use the documents from their cache rather than asking fetching the document again if the version of the document has already been fetched.

Clients should consider documents with newer lineage to be "newer" regardless of the version number. Documents of the same lineage are considered newer if they have the version number is greater.

Example

```
{
  "request": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "peer-publish-notify",
```



```
    "documents": {
      "document": {
        "details": {
          "name": "/hookflash.com/presence/1.0/bd520f1d...9b7ff528e83470e/883fa7...9533609131",
          "version": 12,
          "lineage": 43493943,
          "scope": "location",
          "contact": "peer://example.com/ea00ede4405c99be9ae45739ebfe57d5",
          "location": "524e609f337663bdbf54f7ef47d23ca9",
          "lifetime": "session",
          "expires": 44241421,
          "mime": "text/json",
          "encoding": "json"
        }
      },
      "data": {...}
    }
  }
}
```

```
{
  "result": {
    "$id": "abc123",
    "$handler": "peer-common",
    "$method": "document-publish-notify",
    "$timestamp": 13494934
  }
}
```

Peer Finder Protocol

Session Create Request

Purpose

Obtain a session token that represents the peer on the finder server so continuous proof of identity is not required for each request.

Inputs

- One time use session proof bundle, consisting of
  - Session ID
  - Finder ID – where this request is to be processed
  - Peer contact making the request
  - Client nonce – cryptographically random one time use key
  - Expiry for the one time use token
  - User agent connecting, e.g. "application/major.minor[.build] (information)"
  - Public peer file
  - Signed by peer private key
- Location details
  - Location ID
  - Device ID
  - IP
  - User agent
  - OS
  - System
  - Host

Outputs

- Expiry epoch (when next a keep alive must be sent by)

Security Considerations

The server must validate that the token bundle has not expired.

The server must verify that the request has been signed by the peer's private peer file. The contact id specified in the bundle must match the calculated contact ID based on the included public peer file Section "A". The one time key is used to prevent replay attacks to the server by ensuring the registration can only be used once on the server.

If a Section-B of the public peer file is not present, the peer does not wish to be found in the network.

Example

```
{
  "request": {
    "$domain": "domain.com",
```



```
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "session-create",

    "sessionProofBundle": {
      "sessionProof": {
        "$id": "6fc5c4ea068698ab31b6b6f75666808f",

        "finder": { "$id": "a7f0c5df6d118ee2a16309bc8110bce009f7e318" },
        "clientNonce": "09b11ed79d531a2ccd2756a2104abbbf77da10d6",
        "expires": 4848343494,

        "location": {
          "$id": "5a693555913da634c0b03139ec198bb8bad485ee",
          "contact": "peer://domain.com/920bd1d88e4cc3ba0f95e24ea9168e272ff03b3b",
          "details": {
            "device": { "$id": "e31fcab6582823b862b646980e2b5f4efad75c69" },
            "ip": "28.123.121.12",
            "userAgent": "hookflash/1.0.1001a (iOS/iPad)",
            "os": "iOS v4.3.5",
            "system": "iPad v2",
            "host": "foobar"
          }
        },

        "peer": {
          "$version": "1",
          "sectionBundle": {
            "section": {
              "$id": "A",
              ...
            }
          }
        }
      },
      "signature": {
        "reference": "#6fc5c4ea068698ab31b6b6f75666808f",
        "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
        "digestValue": "IUe324k...oV5/A8Q38Gj45i4jddX=",
        "digestSigned": "MDAwMDAw...MGJ5dGVzLiBQbGVhc2UsIGQ=",
        "key": { "uri": "peer://example.com/920bd1d88e4cc3ba0f95e24ea9168e272ff03b3b" }
      }
    }
  }
}
```

```
{
  "result": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "session-create",
    "$timestamp": 13494934,

    "server": "hooflash/1.0 (centos)"
    "expires": 483949923

  }
}
```

Session Delete Request

Purpose

This request destroys an established session gracefully.

Inputs

- Locations – (optional), if specified without any sub location ID elements, then all locations including this will be unregistered (i.e. a complete system wide unregister), if the location element is missing then the current location associated with the session is unregistered
    - Location ID – (optional), for each location to unregister

Outputs

- The list of locations which were in fact unregistered, each listed by location ID

Security Considerations

The client must have an established session to issue this request.

If the client is done with the current session it may immediately disconnect after receiving the response.

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "session-delete",

    "locations": {
      "location": [
        { "$id": "99609d8b1eb4c413813cbeb7c15137837d4037e9" },
        { "$id": "c8062df29e62d42a3dad60e57d9e84ba38e5ba47" }
      ]
    }
  }
}
```

```
{
  "result": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "session-delete",
    "$timestamp": 13494934,

    "locations": {
      "location": [
        { "$id": "99609d8b1eb4c413813cbeb7c15137837d4037e9" },
        { "$id": "c8062df29e62d42a3dad60e57d9e84ba38e5ba47" }
      ]
    }
  }
}
```

Session Keep-Alive Request

Purpose

This request keeps a previous registered location alive in the location database.

Inputs

None.

Outputs

- Expiry epoch (when next a keep alive must be sent by)

Security Considerations

The client must have an established session to issue this request.

Since the client and server are the only entities that know the session ID, the session can only be kept alive between machines without additional security. The Session Keep-Alive Request must arrive on the same Internet connection as the initial Session Create Request.

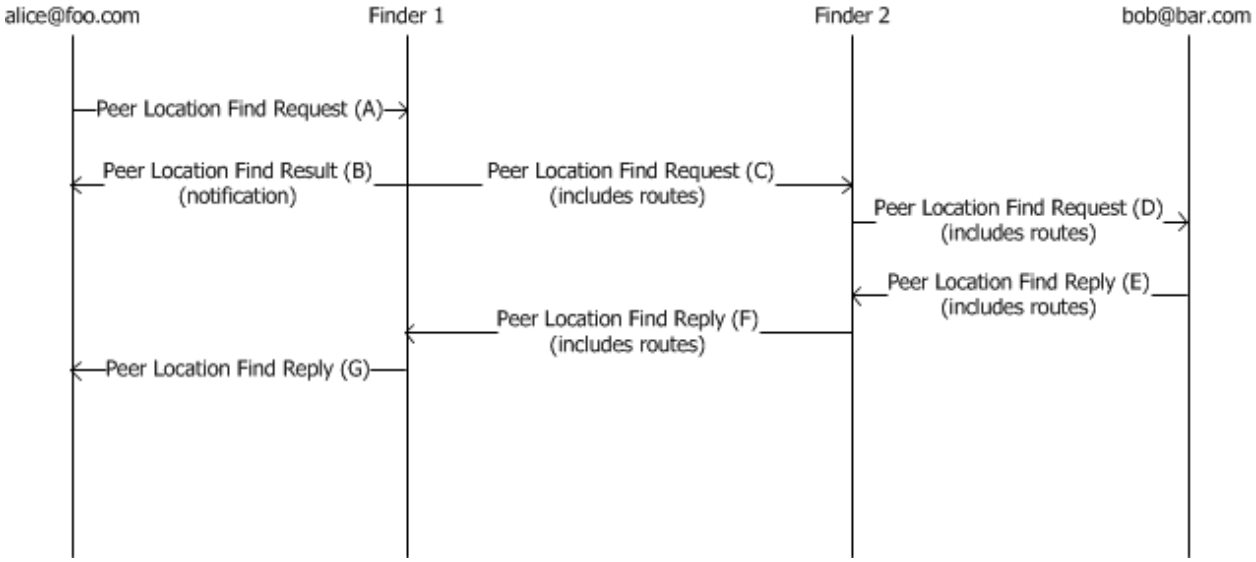
Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "session-keep-alive"
  }
}
```

```
{
  "result": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "session-keep-alive",
    "$timestamp": 13494934,

    "expires": 483949923
  }
}
```

Peer Location Find Request (single point to single point)

Request Flow



Peer Location Find Request (A)

Purpose

This is the request to find a peer that includes the proof of permission to contact the peer and the location information on how to contact the contacting peer.

Inputs

- Cipher suite to use in the proof and for the encryption
- Contact id of contact to be found
- Client nonce – cryptographically random onetime use string
- Find secret proof – i.e. hmac(<find-secret [from public-peer-file-section-B]>, "proof:" + <client-nonce> + ":" + expires))
- Find proof expires
- Peer secret (encrypted) – peer secret is a random key which is then encrypted using the public key of the peer receiving the find request
- Location details
    - Location ID of requesting location
    - Contact ID of requesting location
    - Location details
- Location candidate contact addresses for peer location, each containing transport, IP, port, usernameFrag, password and priority (note: password is encrypted using the peer secret and the IV is the hash of the username frag)
- Signed by peer making request

Security Considerations

The server must verify the server find secret proof is correct according to the information provided in the public peer file of the registered peer being contacted. At this point the one time key should be verified that it has only been seen this one time.

The "peer secret encrypted" is encrypted using the public key of the peer being contacted. Any information encrypted using this key can only be considered valid to/from the requesting peer only if the signature on the proof bundle has been validated by the peer being contacted. Otherwise, it's possible a compromised server could have compromised the "peer secret encrypted" by substituting another encrypted key in its place.

Since the peer being contact doesn't necessarily know the public key of the requesting peer in advanced, the information that is encrypted must be limited to the candidate passwords returned, which at worse can cause the peer being contacted to connect with a malicious peer. However, once the "Peer Identify Request" completes, the contacted peer can validate the requesting peer's find proof bundle at that time.

The peer being contacted will use the "peer secret encrypted" to decrypt the requesting peer's candidate's "password encrypted" and encrypt candidate's passwords in return but cannot assume any channel formed is in fact going to the correct peer until verified.

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",

    "findProofBundle" : {
      "findProof": {
        "$id": "d53255d06a17778b88501f570301e7621c5a7bc4",

        "clientNonce": "7a95ff1f51923ae6e18cdb07aee14f9136afcb9c",

        "find": "peer://domain.com/900c9cb1aeb816da4bdf58a972693fce20e",
        "findSecretProof": "85d2f8f2b20e55de0f9642d3f14483567c1971d3",
        "findSecretProofExpires": 9484848,
        "peerSecretEncrypted": "ODVkMmY4ZjJiMjBlNTVkZ...0MmQzZjE0NDgzNTY3YzE5NzFkMw==",

        "location": {
          "$id": "5a693555913da634c0b03139ec198bb8bad485ee",
          "contact": "peer://domain.com/541244886de66987ba30cf8d19544b7a12754042",
          "details": {
            "device": { "$id": "e31fcab6582823b862b646980e2b5f4efad75c69" },
            "ip": "28.123.121.12",
            "userAgent": "hookflash/1.0.1001a (iOS/iPad)",
            "os": "iOS v4.3.5",
            "system": "iPad v2",
            "host": "foobar"
          },
          "candidates": {
            "candidate": [
              {
                "transport": "rudp/udp",
                "ip": "100.200.10.20",
                "port": 9549,
                "usernameFrag": "7475bd88ec76c0f791fde51e56770f0d",
                "passwordEncrypted": "ZWJiOGM1ZTll...TY0ODRkYzZiZTg0YWZmYWExNDQ4OWMxZTU1Nw==",
                "priority": 43843
              },
              {
                "transport": "rudp/udp",
                "ip": "192.168.10.10",
                "port": 19597,
                "usernameFrag": "398ee0fca8badd89927efc52f0db0f2",
                "passwordEncrypted": "ZDJkNWY1YjU...OWZkOGE2MTM2MWM4NzJmOWQ3YzJjMDgxZTU3Nw==",
                "priority": 32932
              }
            ]
          }
        }
      },
      "signature": {
        "reference": "#d53255d06a17778b88501f570301e7621c5a7bc4",
        "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
        "digestValue": "ZDUzMjU1ZDA2YTE...NjIxYzVhN2JjNA==",
        "digestSigned": "WkRVek...TNelZoTjJKak5BPT0=",
        "key": { "uri": "peer://domain.com/541244886de66987ba30cf8d19544b7a12754042" }
      }
    },

    "exclude": {
      "locations": {
        "location": [
          { "$id": "c52591f27deab5cd48bc515e61a3df4d" },
          { "$id": "59d090f7fdd43a2a59beb2018609e2f2" }
        ]
      }
    }
  }
```



```
}
```

Peer Location Find Result (B)

Purpose

This is the result to the request and it returns a list of locations that the peer finder will attempt to contact.

Outputs

- List of locations being searched
- Additional information about the locations (as applicable)

Security Considerations

Since the request was successfully issued, the information contained in the details section of the Peer Location Register Request of the peer being contacted is returned to the requester. The requester will then know how many locations will be contacted and where to expect a reply.

Example

```
{
  "result": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",
    "$timestamp": 13494934,

    "locations": {
      "location": [
        {
          "$id": "170f5d7f6ad2293bb339e788c8f2ff6c",
          "contact": "peer://domain.com/900c9cb1aeb816da4bdf58a972693fce20e",
          "details": {
            "device": { "$id": "e31fcab6582823b862b646980e2b5f4efad75c69" },
            "ip": "28.123.121.12",
            "userAgent": "hookflash/1.0.1001a (iOS/iPad)",
            "os": "iOS v4.3.5",
            "system": "iPad v2",
            "host": "foobar"
          }
        },
        {
          "$id": "5a693555913da634c0b03139ec198bb8bad485ee",
          ...
        }
      ]
    }
  }
}
```

Peer Location Find Request (C)

Purpose

This is the forwarded request to the finder responsible for the contacted peer.

Inputs

- Same information as Peer Location Find Request (A)
- Routes (stack of routes for reversing the request)

Security Considerations

The server attaches a route message to know which peer connection to send any reply back. The route identifier for the peer connection should be cryptographically random.

In theory, a malicious peer later responding the Peer Location Find Request could change the route in the Peer Location Find Reply and cause the message to erroneously be delivered to the wrong peer attached to a finder. However, even if the malicious peer managed to misdirect to the wrong peer the reply would get dropped since request never matched any request previously sent out by that peer. The worst-case scenario is a malicious client could waste the bandwidth and processing power of another finder. Since the only way to possibly know this route in advance is by the malicious peer having established communication with targeted peer previously, the offending malicious peer already could waste the processing power and bandwidth of the targeted peer by sending bogus Peer Location Find Requests. Hence adding protection against this attack does not achieve any lessening in vulnerability.

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",

    "findProofBundle" : {
      ...
    },

    "routes": {
      "route": { "$id": "27f2e2cfc87d1f77d44afde730bfa15a" }
    }
  }
}
```

Peer Location Find Request (D)

Purpose

This is the forwarded request to the contacted peer.

Inputs

- Same information as Peer Location Find Request (C)
- Additional route(s) (stack of routes for reversing the request)

Security Consideration

The peer finder will add the route where it received the Peer Location Find Request from before sending it to the peer receiving the request. Thus when the reply comes in the reply will be directed back alone the path it was sent.

A malicious peer in theory could change the route in the Peer Location Find Reply to another peer finder on the way back by inserting a misdirected route thus wasting the peer finder's bandwidth and CPU (if this happen the Peer Location Find Reply would eventually get dropped thus is not a security whole).

Protecting against this attack is not worthwhile since a malicious peer could waste the peer's bandwidth and CPU by sending requests to the attacked peer finder directly.

A peer finder should keep track of the request to response ratio (i.e. the number of Peer Location Find Requests sent to a peer versus the number of Peer Location Find Replies received from a peer) within windows in which requests were sent. If Peer Location Find Replies are received outside the normal boundaries or outside the window of Peer Location Find Requests, this could be a peer attempting to attack another peer finder using a its peer finder as a proxy by sending bogus Peer Location Find Replies that never had requests.

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",

    "findProofBundle" : {
      ...
    },

    "routes": {
      "route": [
        { "$id": "27f2e2cfc87d1f77d44afde730bfa15a" },
        { "$id": "79434b37a8bb8408fca8c16b89e4faca" }
      ]
    }
  }
}
```

Peer Location Find Reply (E)

Purpose

This is the reply from the contacted peer to the contacting peer.

Outputs

- Digest value from signature sent in original request – the reply location might not have the ability to validate the signature of the request but the reply location must validate the signature's hash value is correct and copy this value back to the original requester bundled in its own signed package (since the requester knows the original value and must have the public peer file of the reply location to validate the reply's bundle). This allows the requester to validate the original request remained non-tampered throughout and ignore replies where tampering might have occurred.
- Location identifier of location being contacted
- Candidates
    - Candidate transport
    - Candidate IP
    - Candidate port
    - Candidate username fragment
    - Candidate password encrypted

    - Candidate priority
- Signed by replying peer
- Route(s) given in request (stack of routes for reversing the request)

Security Considerations

The contacted peer will decrypt the "peer secret encrypted" using its private key and use the "peer secret" to encrypt its candidate passwords in the reply. Since these usernames and passwords are used exclusively for the sake of discovering contactable addresses between peers in an ICE fashion, the worse a compromised finder could do would be to misdirect a peer to contact a wrong address. Since a malicious finder could already misdirect peers there is no additional protection provided by securing these credentials further.

In theory a compromised finder could respond to the Peer Location Find Request attempting to direct the original requesting peer initiating a connection to malicious host. However, since the compromised finder cannot know the "peer secret", the misdirection attempt will fail.

However, a compromised finder could misdirect the contacted peer by substituting the request's "peer secret" and candidates with its own candidates. While the channel will be opened to the misdirected peer, the peer will fail to communicate as the original requesting peer must prove itself by issuing the "Peer Identity Request".

Example


```
{
  "reply": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",
    "$timestamp": 4344333232,

    "findProofBundle" : {
      "findProof": {
        "requestfindProofBundleDigestValue": "ZDUzMjU1ZDA2YTE...NjIxYzVhN2JjNA==",
        "location": {
          "$id": "1f77425b06b33bfc1d9932a0716f3f2c92ec0e5",
          "contact": "peer://domain.com/541244886de66987ba30cf8d19544b7a12754042",
          "details": {
            "device": { "$id": "e31fcab6582823b862b646980e2b5f4efad75c69" },
            "ip": "100.200.10.20",
            "userAgent": "hookflash/1.0.1001a (iOS/iPad)",
            "os": "iOS v4.3.5",
            "system": "iPad v2",
            "host": "smartie"
          },
          "candidates": {
            "candidate": [
              {
                "transport": "rudp/udp",
                "ip": "100.200.10.20",
                "port": 9549,
                "usernameFrag": "7475bd88ec76c0f791fde51e56770f0d",
                "passwordEncrypted": "MmY4MzgzMz...NWFkN2E1NDM0OTk2YzYwMDU2YjhkYzA2MmYxZA==",
                "priority": 4388438
              },
              {
                "transport": "rudp/udp",
                "ip": "192.168.10.10",
                "port": 19597,
```

```
                "username": "398ee0fca8badd89927efc52f0db0f2",
                "passwordEncrypted": "ZDg4MDNhM...dlYmEzYmFiNmE0YzNhMGQ1OGQ1MzMxZWFmNWJmYg==",
                "priority": 43923293
              }
            ]
          }
        }
      },
      "signature": {
        "reference": "#d53255d06a17778b88501f570301e7621c5a7bc4",
        "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
        "digestValue": "ZDUzMjU1ZDA2YTE...NjIxYzVhN2JjNA==",
        "digestSigned": "WkRVek...TNelZoTjJKak5BPT0=",
        "key": { "uri": "peer://domain.com/541244886de66987ba30cf8d19544b7a12754042" }
      }
    },

    "routes": {
      "route": [
        { "$id": "27f2e2cfc87d1f77d44afde730bfa15a" },
        { "$id": "79434b37a8bb8408fca8c16b89e4faca" }
      ]
    }
  }
}
```

Peer Location Find Reply (F)

Purpose

This is the forwarded reply from the contacted peer to the contacting peer.

Outputs

- Same information as Peer Location Find Reply (E)
- One less route popped off the stack as stack is reversed

Security Considerations

This is an internal communication from peer finder to peer finder. The receiving peer finder must ensure the final route maps to a connection and drop the message if it does not match an active connection.

Example

```
{
  "reply": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",
    "$timestamp": 4344333232,

    "findProofBundle" : {
      ...
    },

    "routes": {
      "route": { "$id": "27f2e2cfc87d1f77d44afde730bfa15a" }
    }
  }
}
```

Peer Location Find Reply (G)

Purpose

This is the forwarded reply from the contacted peer to the contacting peer.

Outputs

- Same information as Peer Location Find Reply (F)
- All routes are now popped of stack

Security Considerations

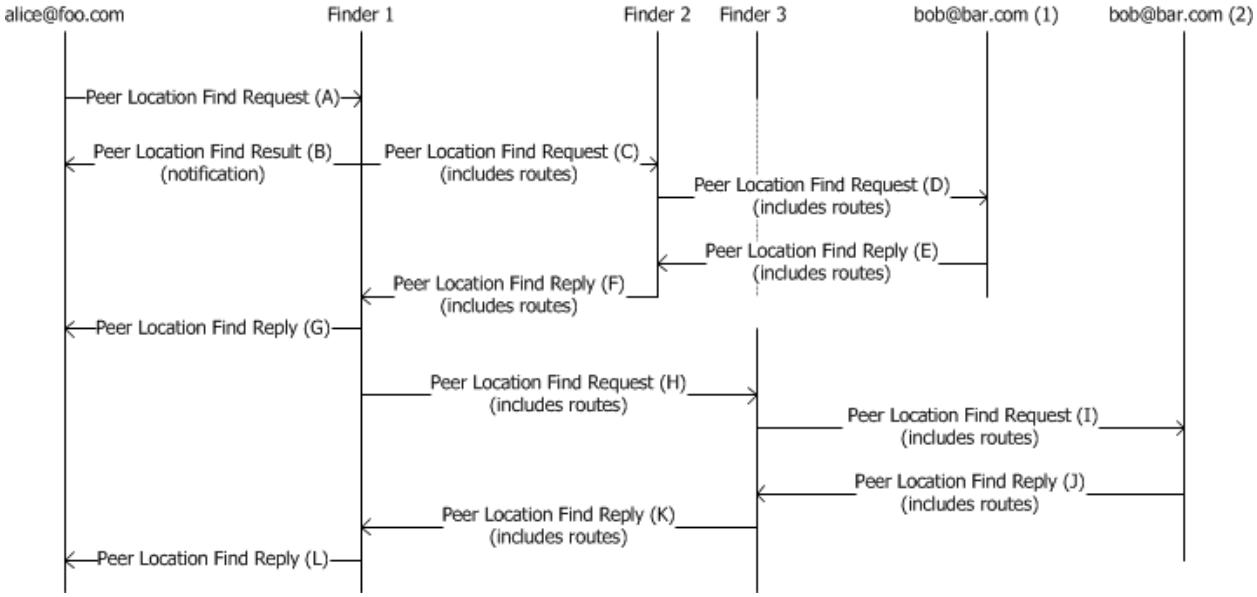
The client will receive location candidates that it believes will belong to the original peer requesting to connect. The client will issue ICE requests to discover which or the candidates are valid. Upon successful ICE discovery, the client will issue a Message/RUDP/UDP connection to the receiving peer (or the encrypted alternative).

Example

```
{
  "reply": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",
    "$timestamp": 4344333232,

    "findProofBundle" : {
      ...
    }
  }
}
```

Peer Location Find Request (single point to multipoint when challenged)

Request Flow



The request is identical to "single point to single point" except the request would fork to the two Finders responsible for the two different locations of "bob@bar.com". While above shows one request fork completing before the other request fork begins, in reality the requests would fork simultaneously. Given that another location exists for "bob@bar.com", the request start out identical but the routes would diverge and the resulting reply would be complete different.

For the sake of simplicity, Peer Location Find Request/Reply A-H are not repeated.

Peer Location Find Request (H)

Purpose

This is the forked forwarded request to the finder responsible for the secondary location of contacted peer.

Inputs

- Same information as Peer Location Find Request (A)
- Routes (stack of routes for reversing the request) – the route would probably have same identifiers as Peer Location Find Request (C) as the contacting peer exists at the same routable location.

Security Considerations

Same as Peer Location Find Request (C)

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",

    "findProofBundle" : {
      ...
    },

    "routes": {
      "route": { "$id": "27f2e2cfc87d1f77d44afde730bfa15a" }
    }
  }
}
```

Peer Location Find Request (I)

Purpose

This is the forwarded request to the contacted peer.

Inputs

- Same information as Peer Location Find Request (H)
- Additional route(s) (stack of routes for reversing the request) – the additional route would be different since the route to reverse from Finder 3 to Finder 1 is different than the route from Finder 2 to Finder 1.

Security Considerations

Same as Peer Location Find Request (D)

Example

```
{
  "request": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",

    "findProofBundle" : {
      ...
    },

    "routes": {
      "route": [
        { "$id": "27f2e2cfc87d1f77d44afde730bfa15a" },
        { "$id": "c173cc0e1653a64f31b0bf12a1f72d1d" }
      ]
    }
  }
}
```

Peer Location Find Reply (J)

Purpose

This is the reply from the contacted peer at the secondary location to the contacting peer. This looks very much like Peer Location Find Reply (E) except the identifiers would be completely different.

Outputs

- Same username as provided in Peer Location Find Request (A)
- Same information as applicable in Peer Location Find Reply (E) but with different identifiers since these identifiers come from a different contacted peer location.
- Same routes as provided in Peer Location Find Request (I)

Security Considerations

Same as Peer Location Find Reply (E)

Example

```
{
  "reply": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",
    "$timestamp": 4344333232,

    "findProofBundle" : {
      "findProof": {
        "requestfindProofBundleDigestValue": "ZDUzMjU1ZDA2YTE...NjIxYzVhN2JjNA==",
        "location": {
          "$id": "d0866fe404867a94949771bfd606f68c3c3c5bd1",
          "contact": "peer://domain.com/541244886de66987ba30cf8d19544b7a12754042",
          "details": {
            "device": { "$id": "54700644c8ce4c663457c7433d6b49ed" },
            "ip": "75.43.32.12",
            "userAgent": "hookflash/1.0.1001a (iOS/iPad)",
            "os": "iOS v4.3.5",
            "system": "iPad v2",
```

```
            "host": "foobie"
          },
          "candidates": {
            "candidate": [
              {
                "transport": "rudp/udp",
                "ip": "75.43.32.12",
                "port": 43432,
                "usernameFrag": "5158a221a95f9d1f57763f8373875807732992b0",
                "passwordEncrypted": "NTg1MjFjZmUyMDc...NzhjMDcxYzZlZDY2NDQzNDNiZGIwNmQ4Nw==",
                "priority": 39932
              },
              {
                "transport": "rudp/udp",
                "ip": "192.168.10.200",
                "port": 20574,
                "usernameFrag": "643fce25e69fd1023abb02af48e90446d7add1be",
                "passwordEncrypted": "NmU5Zj...NTAzZWNlOTc4YWQ4Njc5ZTcwNGUzNzA1Yzg5NjNiNw==",
                "priority": 488323
              }
            ]
          }
        }
      },
      "signature": {
        "reference": "#d53255d06a17778b88501f570301e7621c5a7bc4",
        "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
        "digestValue": "ZDUzMjU1ZDA2YTE...NjIxYzVhN2JjNA==",
        "digestSigned": "WkRVek...TNelZoTjJKak5BPT0=",
        "key": { "uri": "peer://domain.com/541244886de66987ba30cf8d19544b7a12754042" }
      }
    },

    "routes": {
      "route": [
        { "$id": "27f2e2cfc87d1f77d44afde730bfa15a" },
        { "$id": "c173cc0e1653a64f31b0bf12a1f72d1d" }
      ]
    }
  }
}
```

Peer Location Find Reply (K)

Purpose

This is the forwarded reply from the contacted peer at the secondary location to the contacting peer.

Outputs

- Same information as Peer Location Find Reply (J)
- One less route popped off the stack as stack is reversed

Security Considerations

Same as Peer Location Find Reply (F)

Example

```
{
  "reply": {
    "$domain": "domain.com",
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",
    "$timestamp": 435848548434,

    "findProofBundle" : {
```

```
      ...
    },

    "routes": {
      "route": { "$id": "27f2e2cfc87d1f77d44afde730bfa15a" }
    }
  }
}
```

Peer Location Find Reply (L)

Purpose

This is the forwarded reply from the contacted peer to the contacting peer.

Outputs

- Same information as Peer Location Find Reply (K)
- All routes now popped of stack

Security Considerations

Same as Peer Location Find Reply (G)

Example

```
{
  "reply": {
    "$id": "abc123",
    "$handler": "peer-finder",
    "$method": "peer-location-find",
    "$timestamp": 4344333232,

    "findProofBundle" : {
      ...
    }

  }
}
```


Peer To Peer Protocol

Peer Identify Request

Purpose

This request notifies the contacted peer of the original requesting peer's identity. This request must be the first request sent from the peer that initiated the connection (i.e. the requesting peer) to the peer that received the connection (i.e. the contacted peer).

Inputs

- Contact of the initiating contact peer
- Client nonce
- Expiry of the request
- Find secret as obtained from the Section "B" of the public peer file for the receiving peer – peer should reject peer unless this is present or unless the peer is in a common conversation thread with the peer)
- The location information of initiating peer
- The public peer file of the contacting peer – section A at minimal
- Signed by initiating peer's private key

Outputs

- Location of the receiving peer

Security Considerations

The requesting peer must send this request over a secure channel. The public certificate of the secure channel must match the contacted peer's certificate otherwise contact has been initiated to the wrong peer and the connection must be terminated immediately by the requesting peer.

The requesting peer must choose an expiry window long enough as reasonable for the contacted peer to verify that it wants to allow the initiating peer to connect but no longer. The window must allow for the time to fetch contact information about the peer and verify the identities of the peer from a 3rd party as well as potential access rights. Without this window, if the Peer Identify Request was accidently sent to the wrong contacted peer (which happens to be malicious) by the requesting peer, the malicious contacted peer could connect with the real receiving peer and replay the Peer Identify Request message. However, as long as the requesting peer verifies the receiving peer's public certificate matches what is expected then this replay attack should not be possible unless the contacted peer's private key has already been compromised.

The contacted peer must verify this is the first request it receives from the requesting peer. The contacted peer may disallow anonymous connections and require a verifiable connection ID. The

contacted peer must verify the find secret matches the find secret in its own Section "B" of its public peer file (to insure this was not a replay of a request sent to a different peer file).

The contacted peer must verify the request has not expired and should verify the one time key has not been used before. The signature on the bundle must be verified to ensure the requesting peer signed it.

Example

```
{
  "request": {
    "$id": "abc123",
    "$handler": "p2p",
    "$method": "peer-identify",

    "peerIdentityProofBundle": {
      "peerIdentityProof": {
        "$id": "ec065f4b46a22872f85f6ba5addf1e2",

        "clientNonce": "759cef14b626c9bacc9a52253fd68da29d5b6491",
        "expires": 574732832,

        "findSecret": "YjAwOWE2YmU4OWNlOTdkY2QxNzY1NDA5MGYy",

        "location": {
          "$id": "5c5fdfab4bbf8cc8345555172914b9733b2034a4"
          "contact": "peer://domain.com/db9e3a737c690e7cdcfbacc29e4a54dfa5356b63",
          "details": {
            "device": { "$id": "105f38b84d01e6d4bc60d1123c62c957" },
            "ip": "28.123.121.12",
            "userAgent": "hookflash/1.0.1001a (iOS/iPad)",
            "os": "iOS v4.3.5",
            "system": "iPad v2",
            "host": "foobar"
          },
        },
        "peer": {
          "sectionBundle": {
            "section": {
              "$id": "A",
              ...
            },
            ...
          }
        }
      },
      "signature": {
        "reference": "#ec065f4b46a22872f85f6ba5addf1e2",
        "algorithm": "http://openpeer.org/2012/12/14/jsonsig#rsa-sha1",
        "digestValue": "ZGZrbnNua2...pmZXdraiBlYnJlcnJmZXJl",
        "digestSigned": "WkdacmJuTnVhMnBtWlhkcmFpQmxZbkpsY25KbVpYSmw=",
        "key": { "uri": "peer://example.com/db9e3a737c690e7cdcfbacc29e4a54dfa5356b63" }
      }
    }
  }
}
```

```
{
  "result": {
    "$id": "abc123",
    "$handler": "p2p",
    "$method": "peer-identify",
    "$timestamp": 43848328432,

    "location": {
      "$id": "9e02827c0f43c511c30bd410bacf9a83",
      "contact": "peer://domain.com/8da6e36f8a86ba4210c008e0f0d1ba76",
      "details": {
```

```
        "device": { "$id": "e473775bf1db49090ba54f7503623c91" },
        "ip": "74.213.129.11",
        "userAgent": "hookflash/1.0.1001a (iOS/iPad)",
        "os": "iOS v4.3.5",
        "system": "iPad v2",
        "host": "momo"
      }
    }
  }
}
```

Peer Keep-Alive Request

Purpose

This request keeps a connection alive between the peers.

Inputs

None.

Outputs

- Expiry epoch (when next a keep alive must be sent by)

Security Considerations

The client must have an established peer session to issue this request.

Example

```
{
  "request": {
    "$id": "abc123",
    "$handler": "p2p",
    "$method": "peer-keep-alive"
  }
}
```

```
{
  "result": {
    "$id": "abc123",
    "$handler": "p2p",
    "$method": "peer-keep-alive",
    "$timestamp": 13494934,

    "expires": 483949923
  }
}
```

Document Specifications

The published document specifications are outside the scope of this particular document. Please refer to the "Open Peer Conversation Thread Specification" as a proposal for multiparty peer hosted conversations for chat, audio and video (and other media).