Network Working Group
Internet Draft
Intended status: Standards Track
Expires: July 2012

A. Dalela
Cisco Systems
M. Hammer

January 4, 2012

Service Orchestration Protocol
draft-dalela-sop-00.txt

Abstract

Cloud services need to interoperate across cloud providers, service vendors and private/public domains. To enable this interoperability, there is need for a standard wire-format for exchanging service information. This document describes a Service Orchestration Protocol (SOP) to be used as a standard wire-format for cloud exchanges. Similar to widely used protocols like HTTP, SIP and SMTP, SOP uses text-based messages, which are easily extensible and may be inspected at cloud proxies. While SOP carries service-independent information, service-dependant information is attached as a Service Description Framework [SDF] payload to SOP packets. This is similar to how HTML is transported over HTTP. SDF is a XML schema for describing services. SOP and SDF enable any kind of service to be discovered and orchestrated across private and public domains. Simple protocol compliance tests can be employed to ensure interoperability across domains. SOP wire-formats can be used with existing cloud APIs. Using these, it would be possible to interoperate diverse APIs and cloud services across service providers, service vendors and service users.

Table of Contents

1. Introduction

This document describes a protocol for discovering, publishing and orchestrating services in the Internet. A detailed requirement set for such a protocol is described separately [REQT].

The present protocol scheme comprises of two protocols - the Service Orchestration Protocol (SOP) and a Service Description Framework (SDF). SOP carries service-independent information while SDF encodes service-dependant information. The relation between SOP and SDF is similar to that between SIP and SDP or between HTTP and HTML. This document describes SOP. SDF is described separately [SDF]. SDF payloads can be attached in SOP messages, and this document illustrates that. Through separation of service-independent and service-dependant pieces, the scheme described here can be extended to any service type in an interoperable way.

SOP uses text-based headers similar to HTTP, SIP and SMTP. This makes SOP very easy to implement. To extend SOP, one only needs to add new text headers, or occasionally new methods. These messages may be inspected at cloud Proxies and modified by them.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

3. Terms and Acronyms

The key words Provider, Vendor, User, Orchestration, Client, in this document have the same meaning as defined in SOP requirements [REQT].

The key words Proxy, Workflow Server (WS), Service Node (SN) in this document have the same meaning as defined in SOP architecture [ARCH].

The key words Service Description Framework (SDF), Service Domain Name (SDN), SDN Attributes, Vendor Specific Attributes (VSA) in this document have the same meaning as define the SDF specification [SDF].

4. Overview of Operation

4.1. Service Discovery

Before Users can orchestrate services, SNs, Proxies and Users must discover each other, and their service-related capabilities. If the number of users, SNs and Proxies is small, this discovery is achieved by configuring addresses manually. But, in a large network with a variety of services, automated discovery procedures are needed.

This section describes SOP automated discovery procedures in three steps: (a) discovery of the Proxy, (b) registration to the Proxy and Identity validation, and (c) exchanging service information via the Proxy through service subscription and publishing.

4.1.1. Proxy Discovery

A Proxy orchestrates services on a SN. To do this orchestration, the Proxy needs to discover SNs. A simple approach to discovery is that we can configure the SN with the Proxy or the Proxy with the SN address. The Proxy can then query the SN to know if it exists or the SN can heartbeat with the Proxy to announce its presence. This configuration is very cumbersome and error prone at scale and not a preferred method of service discovery.

A more appropriate approach is that we use SOP messages for dynamic discovery. A Proxy needs to advertize its presence along with the ability to proxy for certain services. A SN can discover service-specific Proxies by listening to Proxy advertisements. A SN should also be able to trigger service discovery because the SN may have been out of service when the Proxy advertized its presence.

Two messages are used in SOP - ADVERTISE and DISCOVER - to support proxy discovery. The Proxy sends ADVERTISE to announce its presence and ability to handle some services. The SN uses DISCOVER to trigger the ADVERTISE, if an ADVERTISE has not been transmitted or if it was transmitted but the SN was not operational to receive it.

These two messages are independent requests, and not related as request-response. A Proxy SHALL periodically send an ADVERTISE to announce its presence. A SN SHALL send a DISCOVER whenever it starts up and if an ADVERTISE has not already been received. A Proxy SHOULD send a ADVERTISE in response to a DISCOVER if the service types in the DISCOVER match with the service types supported by the Proxy.

The DISCOVER SHOULD be always broadcast. The ADVERTISE may be unicast or broadcast. Periodically sent ADVERTISE messages SHOULD be

broadcast. ADVERTISE sent in response to DISCOVER SHOULD be unicast (the unicast address is derived from the received DISCOVER).

```
          +--------+                         +--------+
          | Proxy  |                         |   SN   |
          +--------+                         +--------+
               |             DISCOVER             |
               |<---(I need Proxy for XYZ svcs.)---|
               |                                  |
               |            ADVERTISE             |
               |---(I can Proxy for XYZ svcs.)---->|
               |                                  |
```

4.1.2. Service, User and Proxy Registrations

After receiving the ADVERTISE message from a Proxy, and if the service interests match, a SN, a WS, User or Proxy SHOULD register with the Proxy. If a SN supports multiple services, it SHOULD register with those Proxies that support those services. If a Client is interested in services supported by multiple Proxies, it SHOULD register with all of them. If a Proxy is interested in exchanging service information with multiple Proxies, it should register with them. Registration identifies the SN, WS, User or Proxy to the registering Proxy.

```
      +-------------------+                 +--------+
      | SN/WS/User/Proxy |                 | Proxy  |
      +-------------------+                 +--------+
                |                                |
                |           REGISTER             |
                |-----(Here is my Identity)---->|
                |                                |
                |<---------- 200 OK ------------|
                |                                |
```

The Proxy receiving registration MAY initiate authentication during the registration. The Proxy SHOULD interact with a AAA server to authenticate and/or challenge the registration identity.

Registrations MUST be carried out periodically, and they act as a keep-alive mechanism with the Proxy. A registration MAY be used to determine the location of a User, Proxy, WS or SN, and location based policies can be applied in service orchestration.

4.1.3. Service Publishing and Subscribing

There are two kinds of service publishing. First, the SN publishes its capabilities to the Proxy. Second, the Proxy publishes the aggregate of SN capabilities to other Proxies and Users. These publications build up the service routing tables in network and Users and Proxies can know how to route a request to a service.

Service publishing uses the PUBLISH message. This message uses request-response, and a response SHOULD be sent by the receiver.

A SN SHOULD send PUBLISH because of service virtualization. Each SN may be capable of hosting multiple virtualized service instances. As those instances are allocated, the total virtualized capacity in the SN reduces. The SN needs to inform the Proxy about its current capacity to host services. A Proxy WOULD aggregate these capacities and can use them to determine the placement of services.

A SN SHOULD send a PUBLISH to a Proxy in the following three cases, whichever comes first:

a. Periodically based on a time indicated by the Proxy (in the ADVERTISE message). A Proxy SHOULD null the SN's capabilities if 3 successive PUBLISH messages have not been received.

b. When the SN is started, to inform about its available capacity.

c. Whenever its service capabilities change; this may be typically after a new service allocation or partial failure/outage.

A Proxy SHOULD also send PUBLISH to other Proxies and Users to propagate service capabilities in the network. As new services are made available, or existing ones are removed, Users need to know the available list of services. A Proxy can dynamically publish its catalogue of discovered services to the user.

These publications can carry two kinds of information. First, they can have an aggregated view of service capabilities available via a Proxy. This information is useful to a receiver to determine which services can be accessed via a publisher. Second, the PUBLISH can indicate the Workflows that can be requested through a Proxy. The Workflows will be tagged by service types, so a user can know which Workflows will result in which kinds of services.

In certain cases publishing an aggregated view of services may be undesirable at it exposes a provider's internal details to other providers or users. The PUBLISH may however carry aggregated services

to a limited set of trusted Proxies. It SHOULD always carry Workflows to all Users or Proxies that may need to use the Workflows.

```
             +----+                        +-------+
             | SN |                        | Proxy |
             +----+                        +-------+
                |           PUBLISH            |
                |-------(Update on svc X)------>|
                |                              |
                |<---------- 200 OK ------------|
                |                              |

             +----+                        +-------+
             | WS |                        | Proxy |
             +----+                        +-------+
                |           PUBLISH            |
                |---(List of Avlbl Workflows)-->|
                |                              |
                |<---------- 200 OK ------------|
                |                              |

             +-------+                      +-------+
             | Proxy |                      | Proxy |
             +-------+                      +-------+
                |           PUBLISH            |
                |---(List of Avlbl Workflows--->|
                |     and Aggregate Services)   |
                |                              |
                |<---------- 200 OK ------------|
                |                              |

             +-------+                      +-------+
             | Proxy |                      | Client|
             +-------+                      +-------+
                |           PUBLISH            |
                |---(List of Avlbl Workflows)-->|
                |                              |
                |<---------- 200 OK ------------|
                |                              |
```

Of course, not every Proxy or User may be interested in every kind of service. As the service catalogue grows, the total amount of publishing can be an overhead as no User will be interested in all services and a Proxy or User may be interested in only a subset of these services. To avoid flooding, a User or Proxy can indicate its service interests by sending a SUBSCRIBE to the Proxy. The SUBSCRIBE

SHALL indicate the sender's service interests, which act as a filter for a Proxy to forward only those service-specific details.

```
             +---------+                      +--------+
             | Client |                      | Proxy |
             +---------+                      +--------+
                 |            SUBSCRIBE          |
                 |-----(Interested Svc List)---->|
                 |                               |
                 |<---------- 200 OK ------------|
                 |                               |

             +--------+                     +--------+
             | Proxy |                     | Proxy |
             +--------+                     +--------+
                 |            SUBSCRIBE          |
                 |-----(Interested Svc List)---->|
                 |                               |
                 |<---------- 200 OK ------------|
                 |                               |
```

A Proxy MAY use information gleaned during registration to determine which Users and Proxies should be forwarded which information. For instance, it might forward aggregated service information to Proxies within a trust zone, but send only Workflows to external Users.

4.1.4. Disambiguating Discovery Functions

There are similarities in DISCOVER/ADVERTISE and PUBLISH/SUBSCRIBE mechanisms. A SN indicates its service class to the Proxy via DISCOVER and the Proxy advertizes its service class to SN via ADVERTISE. Similarly, a User indicates its service interest via SUBSCRIBE to the Proxy. The Proxy indicates its service capabilities with the PUBLISH to the User. This similarity leads to the question of why we need separate messages for these two functions.

The reason for separating DISCOVER/ADVERTISE from PUBLISH/SUBSCRIBE is that the ADVERTISE/DISCOVER messages may be broadcast and there are no responses to them whereas the PUBLISH/SUBSCRIBE messages are always unicast and every message has a response. It is simpler to keep these messages and preserve their broadcast/unicast and request-only versus request-response behaviors than combining them.

There is similarity in REGISTER/PUBLISH and REGISTER/SUBSCRIBE messages (these messages are unicast and follow the request-response pattern). Both REGISTER and PUBLISH indicate a SN's information to

the Proxy. The difference between them is that a REGISTER informs a Proxy about the SN's identity (Domain Name and IP Address) whereas a PUBLISH informs the Proxy about its Service Domain Name (SDN) and capabilities. Similarly, a REGISTER informs a Proxy about the User's Identity (Domain Name and IP Address) whereas a SUBSCRIBE informs the Proxy about the User's SDN interests. So, we have a 3-way distinction between Capability and Interest which are described in SDN and Identity which is in terms of DNS and IP.

The reason to keep the REGISTER separate from PUBLISH/SUBSCRIBE is to separate Identity (DNS and IP) from Service (SDN). Identity requires authentication, and if a User or SN have been authenticated, this Identity should be usable across all service requests. Authentication should not be required in every service transaction. By confining Identity validation to REGISTER, other requests can use that Identity and don't have to authenticate the User or SN separately. We can think of REGISTER as a single sign-on used for services.

4.2. Service Management

Service management comprises of five main functions: CREATE, DELETE, UPDATE, GET and TRANSFER. The first four messages have a close parallel to the CRUD methods defined in HTTP as shown below.

SOP Message	CRUD Method	HTTP Message
CREATE	CREATE	POST
GET	READ	GET
UPDATE	UPDATE	PUT
DELETE	DELETE	DELETE
TRANSFER	-	-

```
      +--------+                      +--------+
      | Proxy  |                      |   SN   |
      +--------+                      +--------+
          |             CREATE             |
          |----(Services of Domain X)----->|
          |                                |
          |<---------- 200 OK ------------|
          |                                |
```

The CREATE, DELETE and UPDATE messages have obvious semantics in relation to services. The GET message requests a description of the service or workflow or tasks depending on the context. This GET is

not related to the service but to its meta-attributes used in orchestration. For instance, the GET will fetch the total allocated size of a storage disk but not the content of that disk.

The TRANSFER message is unique to SOP, and arises because of service virtualization and movement of services. The TRANSFER message sets up the context in which service information may be transferred from one point to another, such as the source and destination addresses and the type of information to be transferred. It does not deal with the actual transfer of information. The latter kind of transfer may be determined by the service itself (such as use of SFTP to transfer memory snapshots in case of VM mobility).

```
          +--------+                      +--------+
          | Source |                      | Target |
          | Proxy  |                      | Proxy  |
          +--------+                      +--------+
              |           TRANSFER             |
              |-------(src=SN1, dst=?)------->|
              |                                |
              |            200 OK              |
              |<-----(src=SN1, dst=SN2)-------|
              |                                |
```

4.3. Service Reliability

Service orchestration requires reliability that spans multiple SNs. A typical usable service may require orchestration across compute, network, storage, security, etc. Orchestration failure in one of these SNs should trigger a rollback of actions in all other SNs. This is achieved by sending the CANCEL message to the SNs that are impacted by a failure. If however all SNs have been orchestrated successfully, then a COMMIT message is sent to all the SNs.

```
          +--------+                      +--------+
          | Proxy  |                      |   SN   |
          +--------+                      +--------+
              |             COMMIT             |
              |----(Services of Domain X)---->|
              |                                |
              |<---------- 200 OK ------------|
              |                                |
```

Every service transaction in SOP must be committed within a certain time period. If the time for commit has elapsed, the service will be automatically rolled back by the SN. This mechanism is essential to

handle scenarios where the User, the Proxy, some SNs or a combination of these fail. If the Proxy has failed and does not send the COMMIT to a SN, the SN SHALL rollback the transaction. If the SN has failed and does not respond to the Proxy, the Proxy SHALL send a CANCEL to other Proxies and/or SNs. If Proxy has failed and does not respond to the User, the User can be assured that they are not billed for the service. The provider can be assured that unless a COMMIT has been sent, service resources will not be leaking due to failures.

4.4. Service Abstraction

For complex Workflows, there is need for a Workflow Anchor [ARCH]. Each Workflow may have a different Anchor. The Anchor controls the Workflow execution, and is responsible for validating the Workflow prior to execution. A Workflow MUST NOT be branched into Tasks by any network element upstream from the Anchor. The Proxy that advertizes the Workflow into the service network SHALL indicate through the PUBLISH messages that it is the Anchor of the Workflow. This allows a request to be routed to the correct Proxy (Anchor) for execution.

Since a Workflow cannot be branched prior to the Workflow Anchor, all network elements upstream from the Anchor must request the Workflow as an atomic element. The WORKFLOW message is used to make this atomic request. This message MUST contain the Workflow Name being requested. The request MAY also contain a complete or partial Workflow description. All network elements prior to the Anchor MUST forward the WORKFLOW request without branching it into Tasks.

On the receiving the WORKFLOW message, the Anchor SHALL extract the Workflow and send it to the relevant WS along with the Workflow Name through the GET message, requesting the WS to validate and complete the Workflow specification. The Workflow in the GET request represents the input to the WS to construct a complete Workflow specification. The WS may override the Workflow description, or may reject the GET request, depending on configured policies.

```
          +--------+                     +-------+
          | Proxy  |                     | Proxy |
          +--------+                     +-------+
              |           WORKFLOW           |
              |--------(Workflow Name)------->|
              |                              |
              |<---------- 200 OK ------------|
              |                              |
```

If the Workflow has been accepted, the complete Workflow specification SHALL specify individual Tasks as CREATE, DELETE,

TRANSFER, COMMIT, etc. that the Anchor can execute. On receiving the completed Workflow specification, the Anchor SHALL proceed to executing the Workflow. If the Workflow has been rejected, the Anchor SHALL forward the reject to the upstream network element.

The WORKFLOW message allows service bundles to be treated as atomic services for upstream network elements. A customer or provider can create a Workflow description and PUBLISH it through the service network using a unique service name. Users or Providers can request this service through the WORKFLOW message. By anchoring the Workflow in a particular location, the correct execution of the Workflow can be guaranteed. The Anchor ensures that the Workflow is validated and authorized prior to execution and accounted after completion.

5. Protocol Description

SOP is a text-based protocol used for discovering, requesting, updating, moving and tearing down services. The protocol has a number of similarities with SIP, HTTP and SMTP:

```
-  SOP uses text-based headers

-  SOP separates control and data, as in SIP/SDP, HTTP/HTML

-  Uses 1xx-6xx responses
    o 1xx   Informational Responses
    o 2xx   Successful Responses
    o 3xx   Redirection Responses
    o 4xx   Client Failure Responses
    o 5xx   Server Failure Responses
    o 6xx   Global Failure Responses

-  Defines unique request types
    o DISCOVER  :: Used by a SN to discover a Proxy
    o ADVERTISE :: Used by a Proxy to Advertize its Presence
    o REGISTER  :: Used by a SN to register itself with a Proxy
    o PUBLISH   :: Used by a SN to indicate its Capabilities
    o SUBSCRIBE :: Used by a WS/Client to subscribe to service info
    o CREATE    :: Used by a Proxy to request service creation
    o DELETE    :: Used by a Proxy to request service deletion
    o UPDATE    :: Used by a Proxy to request service update
    o GET       :: Used by a Proxy/SN to get service/workflow details
    o TRANSFER  :: Used by a Proxy to request service mobility
    o COMMIT    :: Used by a Proxy to commit service changes
    o CANCEL    :: Used by a Proxy/Client to cancel operations
    o WORKFLOW  :: Used by a Proxy/Client to execute a workflow
```

5.1. Transport

SOP implementations can use UDP or TCP transports. Service orchestration creates, deletes, configures and moves services. These transactions need to be highly reliable and scalable. The combination of reliability and scalability is hard to achieve at the transport layer. If a proxy manages millions of services, persistent TCP connections don't scale. If millions of users connect to a Proxy to obtain services, transporting them over UDP can be error prone.

Keeping in mind these constraints, it is ideal to use TCP for short-lived user interactions and UDP for long-lived service management interactions. SOP has built in timers to COMMIT / CANCEL procedures to deal with network reliability issues in case the Proxy fails or the connection to the Proxy fails. As such, SOP will work correctly with use of either TCP or UDP. The choice of transport therefore depends on the desired scale and reliability of deployment.

Limitations of transport may also be overcome through additional means. For instance it is possible to deploy a set of TCP Proxies that offloads TCP scale outside the SOP Proxy. Alternately we can use UDP transport in combination with shorter retransmit timers and increased retransmit counters. SOP has procedures to dynamically distribute timer and counter values across the network.

5.2. Addressing

All entities in SOP are addressed by a user@domain name. The user "default" can be used in the following circumstances:

- For addressing Proxies and WSs
- For addressing a service when it does not have a name assigned (this can happen prior to SN registration)
- For addressing entities in a broadcast message
- If a sender does not know which user to address it to

Outside of above cases, SNs SHOULD be addressed by a name such as service@sn.provider.com, where "sn" is a service node and "service" is a virtual service instance at the SN. If there is one SN for many virtualized service instances, the SN WOULD receive all messages for virtualized service instances, even though they are addressed to a particular "service". The SN MAY take actions on a request on behalf of the service or pass on information to the service.

When messages are being broadcast, an address of the type default@default.provider.com may be used, and the packet will use a broadcast IP address.

5.3. Transactions and Exchanges

A "transaction" identifies as pair-wise message transfer between SOP network entities. Transactions correlate requests with responses. The "branch" parameter in the Via header identifies a transaction between two SOP network entities. The branch parameter is added by the request initiator to identify a new request and MUST be included by the responding entity in its response messages.

A service request may branch out into multiple SOP network nodes, such as to allocate different resources (compute, network, storage and security) through the same request. The requests and responses are part of the same "exchange". The Exchange identifies a set of transactions that must be executed to complete a single service request. The combination of a "branch" and "exchange" uniquely identifies a message in the service network.

5.4. Service Accounting

Accounting responsibilities rest with the Proxy. The Proxy SHALL start service accounting by sending an Accounting Start request to the AAA server. It SHALL stop service accounting by sending an Accounting Stop request to the AAA server. These messages (or enhancements to these messages for supporting service specific accounting functions) are outside scope of current document.

6. SOP Requests

Every SOP request MUST have the Message Type, From and To headers. Messages with request-response MUST have the Exchange and Via headers. Messages without request-response (ADVERTISE and DISCOVER) MAY NOT have Exchange and Via headers. Other headers vary from message to message as described below. The meanings of all headers are described in Section 8.

6.1. ADVERTISE

This message SHALL be sent by a Proxy to advertize its presence and willingness to proxy for certain types of services. The SDF payload in the message indicates which service domains it can support. In the example message below, the Proxy supports the iaas.compute domain of services. The message SHOULD carry the Registration-Timeout header. The message may also carry other Timer values and Retry counts as a way to globally configure all SNs in the network uniformly.

```
  ADVERTISE 1 SOP/1.0
```

```
From: default@p.provider.com
To: default@default.provider.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p.provider.com;branch=k9DjR5lbcw
Timestamp: 1285162132
Sequence-ID: 13224 ADVERTISE
Registration-Timeout: 1000
Content-Type: application/sdf; charset=utf-8
Content-Length: 147

<?xml version="1.0" encoding="UTF-8"?>
<domain name="iaas.compute" type="capability" def="sdn"/>
```

6.2. CANCEL

This message is used to cancel a transaction that has not been committed in case of errors detected or in case of a timeout. The cancellation is a new transaction and it requires a response. The CANCEL refers to the Task to be cancelled through the Task-ID header. The receiver of the request MAY obtain a description of the Task again by querying the WS with the Task-ID provided in the message.

```
CANCEL 1 SOP/1.0
From: default@p.provider.com
To: default@4357254.provider.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p.provider.com;branch=khewui6GDw
Sequence-ID: 134 CANCEL
Task-ID: 67439375
Workflow-Server: ws.provider.com
Requestor: consumer@customer.provider.com
```

6.3. COMMIT

This message MUST be used to commit CREATE, DELETE, UPDATE or TRANSFER transactions. This message SHOULD be sent by the Proxy to each SN involved in a Workflow after all Tasks have been successfully completed. However, the Workflow specification may override this through a specification of when the COMMIT should be sent. The COMMIT refers to the Task to be committed through the Task-ID header. The receiver of the request MAY obtain a description of the Task again by querying the WS with the Task-ID provided in the message.

The Proxy SHOULD also send a COMMIT to the WS to commit the Workflow. This is an indication to the WS that the Workflow execution was successful. The WS upon receipt of a COMMIT SHALL store the Workflow instance for reference later, and return a Workflow-ID. This

Workflow-ID SHOULD be used to recreate services (in case of disaster recovery) or reverse them (in case of service deletion).

```
     COMMIT 1 SOP/1.0
     From: default@p.provider.com
     To: default@4357254.provider.com
     Exchange: 43shXui7236
     Via: SOP/1.0/UDP default@p.provider.com;branch=khewui6GDw
     Sequence-ID: 134 COMMIT
     Task-ID: 67439375
     Workflow-Server: ws.provider.com
     Requestor: consumer@customer.provider.com
```

6.4. CREATE

This message SHOULD be used by a Proxy to request a service creation to a SN. The CREATE refers to the Task to be executed through the Task-ID header. The receiver of the request SHOULD obtain a Task description by querying the WS with the Task-ID provided in message.

```
     CREATE 1 SOP/1.0
     From: default@p.provider.com
     To: default@4357254.provider.com
     Exchange: 43shXui7236
     Via: SOP/1.0/UDP default@p.provider.com;branch=k9DjR5lbcw
     Sequence-ID: 134 CREATE
     Task-ID: 67439375
     Workflow-Server: ws.provider.com
     Requestor: consumer@customer.provider.com
```

6.5. DELETE

This message SHOULD be used by a Proxy to request a service deletion to a SN. The DELETE refers to the Task to be executed through the Task-ID header. The receiver of the request SHOULD obtain a Task description by querying the WS with the Task-ID provided in message.

```
     DELETE 1 SOP/1.0
     From: default@p.provider.com
     To: default@4357254.provider.com
     Exchange: 43shXui7236
     Via: SOP/1.0/UDP default@p.provider.com;branch=k9DjR5lbcw
     Sequence-ID: 134 DELETE
     Task-ID: 67439375
     Workflow-Server: ws.provider.com
     Requestor: consumer@customer.provider.com
```

6.6. GET

This message MUST be used by a SN to obtain a Task description from a WS. It SHOULD be used by a Proxy to request a SN description prior to a service transfer. It MUST be used by a Proxy to obtain a Workflow or Task description from a WS. The context of the Transaction determines which of these functions need to be achieved.

Depending on the context, appropriate headers must be present. To request a SN description, a Proxy MUST include the Service-ID header to point to refer to the specific service whose description is required. To get a workflow or task description, the appropriate Workflow-Name, Workflow-ID or Task-ID headers MUST be present. The Query-Type header MUST be present to describe the type of query.

```
GET 1 SOP/1.0
From: default@p.provider.com
To: default@ws.provider.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p.provider.com;branch=k9oluElbcw
Sequence-ID: 286 WORKFLOW
Query-Type: workflow-name
Workflow-Name: gTyuI82Zx@provider.com
```

6.7. DISCOVER

This message SHOULD be sent by a SN to discover any Proxies that might be willing or capable to act as Proxy for its services. The SDF content in the message indicates service capabilities in the SN. In the example below, the SN is capable of iaas.compute domain of services.

```
DISCOVER 1 SOP/1.0
From: default@default.com
Via: SOP/1.0/UDP default@default.com;branch=k9DjR5lbcw
Timestamp: 1285162130
Sequence-ID: 1 DISCOVER
Content-Type: application/sdf; charset=utf-8
Content-Length: 147

<?xml version="1.0" encoding="UTF-8"?>
<domain name="iaas.compute" type="capability" def="sdn"/>
```

6.8. PUBLISH

This message SHOULD be sent by a SN whenever its service capabilities change or when the Publish-Timeout timer expires or after a service restart (after a registration), whichever comes earlier.

This message indicates the current service availability of the SN to the Proxy. The Proxy will use the information given in the SDF payload to update its service database. The capabilities indicated here would be used by the Proxy to route service requests towards the SN. The "capability" and "availability" inside the SDF indicate the total capacity and the capacity that is available currently.

```
  PUBLISH SOP/1.0
  From: default@4357254.provider.com
  To: default@p.provider.com
  Exchange: 43shXui7236
  Via: SOP/1.0/UDP default@4357254.provider.com;branch=k9DjR5lbcw
  Sequence-ID: 13432 PUBLISH
  Distance: 1
  Content-Type: application/sdf; charset=utf-8
  Content-Length: 513

  <?xml version="1.0" encoding="UTF-8"?>
  <domain name="iaas.compute" type="capability" def="sdn">
      <!-- list of domain elements -->
  </domain>
  <domain name="iaas.compute" type="availability" def="sdn">
      <!-- list of domain elements -->
  </domain>
```

6.9. REGISTER

This message SHOULD be used by a SN, a Proxy, a WS or a User to register with a Proxy after a new Proxy discovery, or when the Registration-Timeout expires, whichever comes earlier. The Node-Type header informs the Proxy about the type of registering entity. If the Node-Type header indicates a service-node, then the message MAY have the Transfer-Node header, indicating type of mobility the SN supports. For other node-types, Transfer-Node header MUST be absent.

The REGISTER is used by the Proxy to establish the Identity of the SN (its Domain Name and IP Address). The Proxy MAY initiate authentication procedures with the registering entity after the REGISTER is received as described in Section. If the registration fails, the Proxy SHALL NOT register the entity.

The REGISTER is used by the Proxy as a heartbeat or keep-alive mechanism. The Proxy SHALL de-register the entity if 3 successive Registration-Timeouts have expired, and remove all information related to that service. The Proxy MAY initiate service recreation procedures for all services under that SN at another SN, if a SN has been de-registered due to Registration-Timer expiry.

```
REGISTER 1 SOP/1.0
From: default@default.com
To: default@p.provider.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@default.com;branch=k9DjR5lbcw
Sequence-ID: 1 REGISTER
Transfer-Mode: stateless
Node-Type: service-node
```

6.10. SUBSCRIBE

This message allows a SOP network element to request updates on a particular type of service capability. The SDF payload MUST be present to know the capabilities for which the requestor is subscribing. A WS SHALL subscribe with the Proxy for all updates to services and all updates to user registrations. A User MAY subscribe with the Proxy to receive Workflow and service related updates.

```
SUBSCRIBE SOP/1.0
From: default@ws.provider.com
To: default@p.provider.com
Via: SOP/1.0/UDP default@ws.provider.com;branch=k9DjR5lbcw
Exchange: 43shXui7236
Timestamp: 1285162130
Sequence-ID: 1 SUBSCRIBE
Distance: 1
Content-Type: application/sdf; charset=utf-8
Content-Length: 154

<?xml version="1.0" encoding="UTF-8"?>
<domain name="iaas.compute" type="capability" def="sdn"/>
<domain name="users.company.com" type="capability" def="sdn"/>
```

6.11. TRANSFER

This message is sent by a Proxy to initiate a service transfer. The From and To headers SHALL indicate the source and destination Proxies. The Source and Destination headers MUST be present and they indicate the source and destination SNs. The Requestor header MUST be present to identify the initiator of the transfer (this may be a

Proxy). The TRANSFER message sets up the session to initiate a service transfer. It is not responsible for actually transferring service state from one SN to another. The latter is outside the scope of the current document, and each service may choose its own methods to transfer state (such as FTP to transfer a file).

The SDF payload in the TRANSFER message SHALL indicate to the Destination SN the capabilities expected in the transfer. For instance, these capabilities may indicate the total amount of memory or storage required to accept a moving virtual machine. The content of that memory is outside scope for SOP and to be handled separately.

```
TRANSFER SOP/1.0
From: default@p1.provider.com
To: default@p2.provider.com
Exchange: 4j253TyXuM6
Via: SOP/1.0/UDP default@p1.provider.com;branch=XsMf634d2W
Sequence-ID: 1 TRANSFER
Source: service1@4357254.provider.com
Requestor: default@p1.provider.com
Content-Type: application/sdf; charset=utf-8
Content-Length: 142

<?xml version="1.0" encoding="UTF-8"?>
<domain name="iaas.compute" type="capability" def="sdn">
    <!-- list of domain elements -->
</domain>
```

6.12. UPDATE

This message SHOULD be used by a Proxy to request a service update to a SN. The UPDATE refers to the Task to be executed through the Task-ID header. The receiver of the request SHOULD obtain a description of the Task by querying the WS with the Task-ID provided in the message.

```
UPDATE 1 SOP/1.0
From: default@p.provider.com
To: default@4357254.provider.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p.provider.com;branch=k9DjR5lbcw
Sequence-ID: 134 CREATE
Task-ID: 67439375
Workflow-Server: ws.provider.com
Requestor: consumer@customer.provider.com
```

6.13. WORKFLOW

This message is used by a Client or a Proxy to initiate a Workflow execution. It MUST contain the Workflow-Name that needs to be executed. It MUST contain a reference to the prior Workflow-ID when the request involves deletion or transfer of a prior executed Workflow.

```
WORKFLOW 1 SOP/1.0
From: consumer@customer.com
To: default@p.provider.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p.customer.com;branch=k9DjR5lbcw
Sequence-ID: 5 WORKFLOW
Workflow-Name: gTyuI82Zx@provider.com
```

7. SOP Responses

7.1. 100 TRYING

This message MUST be sent by a Proxy on receiving a Workflow request. It indicates that the Proxy has received the request and is attempting to execute it. This message SHOULD be sent by a receiving SN upon a CREATE, UPDATE, DELETE or TRANSFER request, as these operations can take a longer period of time. This message informs the Proxy that the request has been received and it being processed.

```
100 TRYING 1 SOP/1.0
From: default@p.provider.com
To: consumer@customer.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p.provider.com;branch=k9DjR5lbcw
Sequence-ID: 1 TRYING
Workflow-Name: gTyuI82Zx@provider.com
```

7.2. 183 WORKFLOW PROGRESS

The Proxy may send these messages periodically to the Client keeping it informed about the update. These message MAY have a Workflow description that describe the progress in execution.

```
183 SESSION PROGRESS 1 SOP/1.0
From: default@p.provider.com
To: consumer@customer.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p.provider.com;branch=k9DjR5lbcw
Sequence-ID: 1 TRYING
```

```
     Workflow-Name: gTyuI82Zx@provider.com
     Content-Type: application/sdf; charset=utf-8
     Content-Length: 542

     <?xml version="1.0" encoding="UTF-8"?>
     <workflow name="gTyuI82Zx" id="68743693"
         xmlns:sdf="http://sdf.org/sdf">
       <description>workflow description</description>
       <taskgroup id="1" prev="idle" next="idle">
          <description>taskgroup description</description>
          <task id="1" prev="idle" next="idle" action="CREATE"
              server="4357254.provider.com reference="67439375"
              status="pending"/>
       </taskgroup>
     </workflow>
```

7.3. 200 OK

This message is sent by multiple entities including the WS, Proxy and SNs to indicate successful completion of the requests. Depending on the context, message contents and headers will vary. Below is an example message sent by the Proxy on completion of a CREATE.

```
     200 OK 1 SOP/1.0
     From: default@4357254.provider.com
     To: default@p.provider.com
     Exchange: 43shXui7236
     Via: SOP/1.0/UDP default@sn.provider.com;branch=k9DjR5lbcw
     Sequence-ID: 134 CREATE
     Task-ID: 67439375
     Workflow-Server: ws.provider.com
```

7.4. 305 USE PROXY

This message is sent by a Proxy if it wants to redirect the requestor to another Proxy and will not service the request itself. The message MUST have the Alternate-Proxy header.

```
     305 USE PROXY 1 SOP/1.0
     From: default@p.provider.com
     To: user@customer.com
     Exchange: 43shXui7236
     Via: SOP/1.0/UDP default@p1.provider.com;branch=k9DjR5lbcw
     Sequence-ID: 1 WORKFLOW
     Alternate-Proxy: default@p2.provider.com
     Workflow-Name: gTyuI82Zx@provider.com
```

7.5. 400 BAD REQUEST

This message SHALL be sent by a Proxy or a WS or a SN if they find the request malformed. Below is an example message.

```
400 BAD REQUEST 1 SOP/1.0
From: default@p.provider.com
To: user@customer.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p1.provider.com;branch=k9DjR5lbcw
Sequence-ID: 1 WORKFLOW
Workflow-Name: gTyuI82Zx@provider.com
```

7.6. 403 FORBIDDEN

This message SHALL be sent by a Proxy or a WS if they find the user is not authorized to perform an operation. This may be used by a Proxy to indicate that it needs to authenticate in a REGISTER. The message may be used if a Proxy or WS finds that a user is not authorized to execute a Workflow.

```
403 FORBIDDEN 1 SOP/1.0
From: default@p.provider.com
To: user@customer.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p1.provider.com;branch=k9DjR5lbcw
Sequence-ID: 1 WORKFLOW
Workflow-Name: gTyuI82Zx@provider.com
```

7.7. 500 SERVER INTERNAL ERROR

This message SHALL be sent by a Proxy or a WS or a SN if they tried to process a request but failed to some internal error. A SN will typically send this request if it fails to perform requested operation (like service creation, deletion, update, etc.).

```
500 SERVER INTERNAL ERROR 1 SOP/1.0
From: default@p.provider.com
To: user@customer.com
Exchange: 43shXui7236
Via: SOP/1.0/UDP default@p1.provider.com;branch=k9DjR5lbcw
Sequence-ID: 1 WORKFLOW
Workflow-Name: gTyuI82Zx@provider.com
```

7.8. 504 SERVER TIMEOUT

This message SHALL be sent by a Proxy if one of the SNs fails to respond to a request, and the Proxy times out on retransmits.

```
  504 SERVER TIMEOUT 1 SOP/1.0
  From: default@p.provider.com
  To: user@customer.com
  Exchange: 43shXui7236
  Via: SOP/1.0/UDP default@p1.provider.com;branch=k9DjR5lbcw
  Sequence-ID: 1 WORKFLOW
  Workflow-Name: gTyuI82Zx@provider.com
```

7.9. 603 DECLINE

This message SHALL be sent by a Proxy, WS or SN if it is not able to support the request, because there is no capacity available. This response SHOULD be sent if the Proxy knows that no other Proxy will be able to support this request (otherwise send 305 USE PROXY). The scope of the DECLINE is the set of Proxies that this Proxy knows of.

```
  603 DECLINE 1 SOP/1.0
  From: default@p.provider.com
  To: user@customer.com
  Exchange: 43shXui7236
  Via: SOP/1.0/UDP default@p1.provider.com;branch=k9DjR5lbcw
  Sequence-ID: 1 WORKFLOW
  Workflow-Name: gTyuI82Zx@provider.com
```

8. SOP Headers

SOP messages comprise of headers. Each header begins on a new line. Each header field consists of a field name followed by a colon (":") and the field value.

```
   field-name:field-value
```

Header fields follow the same generic header format as that given in Section 7.3.1 of RFC 3261. The order of fields is not important in a SOP message however it is recommended that fields relevant to service routing (To, From, Exchange, Via) are present at the top of the message to facilitate rapid processing.

The first line of every SOP message must have the name of the request or response, along with the SOP version number. This message name determines the subsequent processing of messages. Example message name headers are shown below.

```
CREATE 1 SOP/1.0
GET 1 SOP/1.0
```

The number following the Message Type indicates if the message is being re-transmitted as the same message in the same Transaction. This number MUST be incremented by a sender whenever a message is re-transmitted. A receiver MAY give higher priority to re-transmitted messages.

8.1. Acknowledge Timeout

This header indicates the time in seconds after which a SN MAY send a "reminder" 200 OK message to the Proxy, soliciting a COMMIT. After Retransmission Counter times transmissions of the 200 OK, the SN SHOULD consider the transaction cancelled and initiate rollback operations. This header SHOULD be sent in the Advertisement message to set the global value of the timeout. The header MAY also be sent in the CREATE, DELETE, TRANSFER requests to set the timeout per request basis depending on the time a Proxy anticipates it would take to complete certain operations.

```
Acknowledge-Timeout: 30
```

8.2. Alternate Proxy Header

This header is used to redirect a Client to another Proxy. This can happen if a Proxy knows that the Workflow Anchor (WA) for the Workflow has moved to another Proxy and the Proxy is configured not to forward requests to that Proxy.

```
Alternate-Proxy: p2.provider.com
```

8.3. Cancel Timeout Header

This header indicates the time in seconds after which a Proxy SHOULD send a cancellation for a request if a 200 OK or 100 Trying message has not been received. This header SHOULD be sent in the Advertisement message to set the global value of the timeout. The header MAY also be sent in the CREATE, DELETE, TRANSFER or UPDATE requests to set the timeout per request basis.

```
Cancel-Timeout: 30
```

8.4. Commit Timeout Header

This header defines the time in seconds after a response has been sent that a COMMIT is expected by the SN. Its behavior is described in Section 9.6.

```
Commit-Timeout: 50
```

8.5. Content Type Header

This header defines the kind of content carried in the SOP request. SOP messages are expected to carry SDF payloads. However, it is possible that other kinds of service description schemes may be used in conjunction with SOP. This header MUST be used to specify the type of application payload carried in the SOP message.

```
Content-Type: application/sdf; charset=utf-8
```

8.6. Content Length Header

This header defines length of content carried in the SOP request, computed in bytes. The length MUST NOT include SOP headers. It MUST be the length of the SDF payload carried in the SOP packet.

```
Content-Length: 158
```

8.7. Distance Header

This header helps in identifying the distance of a network entity from a receiver. It MUST be used in PUBLISH messages to advertize services. The value of this header MUST be incremented whenever a PUBLISH is forwarded. The receivers SHOULD use a shortest path to a service, and to avoid service routing loops.

```
Distance: 1
```

8.8. Destination Header

This header MUST be present in the TRANSFER messages. It is used to indicate the address of the destination SN that will be receiving a service. The address MUST conform to addressing in Section 5.2.

```
Destination: default@sn.provider.com
```

8.9. Exchange Header

This header is used to identify the exchange within which a transaction is taking place. The use of Exchange identifiers helps to correlate cross-domain, multi-device or other complex orchestrations that may involve multiple devices or even brokers. This header MUST be present in all request-response messages.

```
Exchange: 43shXui7236
```

8.10. From Header

This header identifies the originator of the service request. It MUST indicate the initiator's address. Addresses MUST conform to SOP addressing in Section 5.2.

```
From: default@p.provider.com
```

8.11. Node Type Header

This header identifies the type of node in the SOP network. The header MUST have one of the following values: (a) "service-client", (b) "workflow-server", (c) "service-proxy" and (d) "service-node". This header is used in REGISTER, PUBLISH and SUBSCRIBE messages. In the REGISTER and PUBLISH messages it helps to identify the sender type and distinguish various types of registrations or publications. In the SUBSCRIBE message it helps the Proxy to determine which node's information must be forwarded to the requestor.

```
Node-Type: service-node
```

8.12. Priority Header

This header MAY be added by requestor to prioritize its processing over other pending messages. Priorities can be any number from 1 to 65535, with 65535 being the highest. This field may be used in conjunction with policy decisions in a cloud service network to prioritize certain requests over others.

```
Priority: 5
```

8.13. Query Type Header

This header MUST be used in GET requests to obtain information about a Task-ID, Workflow-Name, Workflow-ID, List of Workflow-IDs against a Workflow-Name, List of Task-IDs against a Workflow-ID or Name, etc. The header can have the following values:

(a) "active-workflows" - all workflow-ids against a workflow-name. the Workflow-Name MUST be present in the request.

(b) "active-tasks" - all task-ids for a Workflow-Name or Workflow-ID. The Workflow-Name and/or Workflow-ID MUST be present in request.

(c) "workflow-name" - description of a Workflow in Tasks, specified by a Workflow-Name. The Workflow-Name MUST be present in request.

(d) "workflow-id" - description of tasks that were performed in a particular instance of a Workflow, denoted by the Workflow-ID. The Workflow-ID MUST be present in the request.

(e) "task-id" - description of the attributes that make up a specific Task referenced by the Task-ID. The Task-ID MUST be present in the request. Workflow-Name and Workflow-ID MAY be present.

```
   Query-Type: workflow-id
```

8.14. Publish Timeout Header

This timer indicates the period at which a SN must publish its services to the Proxy regardless of change in service availability. Its behavior is described in detail in Section 9.7.

```
   Publish-Timeout: 500
```

8.15. Retry Count Header

This header defines the number of times a SN SHALL send a 200 OK in response to a CREATE, DELETE, TRANSFER or UPDATE messages before receiving a COMMIT. On each retry, the Commit-Timeout MUST be expired before the retry is attempted. After completion of Retry Count number of retries, the SN MUST initiate a service rollback. This header SHOULD be sent in the ADVERTISE message and MAY be sent in individual CREATE, DELETE, TRANSFER or UPDATE messages. In the latter case, the count applies only to that particular Exchange.

```
   Retry-Count: 3
```

8.16. Requestor Header

In some cases, the originator of a request may be different than the requestor. For instance, a Proxy may initiate a request using a non-SOP trigger. In these cases, it is important to include requestor's information in SOP messages. This information may be used for

accounting, or other user-specific customizations. This header MAY be identical to the From Header.

```
      Requestor: service-client@customer.com
```

8.17. Registration Timeout Header

This header indicates to SN the time within which it must register again. Behavior of this timer is described in the Section 9.1.

```
      Registration-Timeout: 10000
```

8.18. Service ID Header

A Service ID MAY be used to refer to a particular instance of a virtual service. A Proxy MUST return a unique Service ID in response to a REGISTER. That ID MUST be used in future to identify services. A new Service ID MUST also be indicated by the WS in case of a new service creation. Moving services SHOULD retain the Service-ID.

```
      Service-ID: 8743435@compute
      Service-ID: 8743435@compute-resources.provider.com
```

8.19. Sequence ID Header

This header MUST be present in all requests and SHALL identify the sequence number of requests of the same type. By adding sequence numbers, a receiver has the ability to identify which requests have come later. The sequence numbers MUST be reset after they have reached a maximum of 32 bits unsigned integer.

```
      Sequence-ID: 8624 CREATE
```

8.20. Source Header

This header is used for service mobility and indicates the address of the source SN that will be transferring a service.

```
      Source: default@sn.provider.com
```

8.21. Subscribe Timeout Header

This timer indicates the period at which a TS or WS must SUBSCRIBE for SNs or Clients regardless of change in their availability. Its behavior is described in detail in Section 9.9.

```
      Subscribe-Timeout: 500
```

8.22. Task ID Header

This header identifies a unique type of configuration in the TS. A common ID may be used across a number of SNs. Configuration specific to SN would be passed based on the requestor's address in GET. Use of an ID allows all configurations to be stored centrally.

```
      Task-ID: 67439375
```

8.23. Timestamp Header

This header MAY be added by requestor to identify the time of the request origination. It MAY be used to drop requests that are older than the specified expiry time period. The timestamp will specify the expired seconds according to UTC since January 1, 1961.

```
      Timestamp: 1285162130
```

8.24. To Header

This header identifies the destination of the service request. It must indicate a user-id to which the service request is targeted.

```
      To: default@p.provider.com
```

8.25. Transfer Mode Header

This header is used in the REGISTER message. The SN identifies the manner in which it can support service mobility. If the SN is capable of transferring live state to another SN, it SHOULD set this header to "stateful". If the SN cannot transfer live state, it must set this header value to "stateless". In stateless transfers, a service instance is created and the old one is deleted.

```
      Transfer-Mode: stateful
```

8.26. Via Header

This header MUST be inserted by every SOP node through which the SOP request traverses, provided the SOP node is not acting as a stateless proxy. The forwarder MUST insert its SOP address and a random branch parameter to identify the transaction between the peering entities. The header will also have the SOP version and indicate the transport over which SOP message is transmitted.

```
      Via: SOP/1.0/UDP default@sb.provider.com;branch=k9DjR5lbcw
```

8.27. Workflow ID Header

This header identifies a unique instance of a Workflow. A workflow may be customized for a user based on user-specific policies. The Workflow ID is the specific customized instance of a workflow. It may reference a stored workflow in the WS, or with sufficient number of parameters so as to compute the workflow at a later time.

```
Workflow-ID: 68743693@provider.com
```

8.28. Workflow Name Header

This header identifies a unique Workflow within a provider. The Workflow Name allows a client or Proxy to request a workflow execution. A Proxy SHOULD send PUBLISH messages listing available workflows. Workflow Name MUST identify the provider who supports the workflow. This allows workflow names to overlap across providers.

```
Workflow-Name: X32mnTrUwq@provider.com
```

8.29. Workflow Server Header

This header provides the address of a workflow server that a SN or Proxy may query for processing a request, such as to obtain a task description. This header should be used in conjunction with the Task-ID header.

```
Workflow-Server: Workflow-Server.provider.com
```

9. Timers and Counters

9.1. Retransmission Counter

This counter defines the number of retransmissions for a request before a CANCEL is sent, or for a response before a COMMIT is expected. The value of counter SHOULD be set by the Retry-Count header globally in the ADVERTISE message or per transaction in CREATE, DELETE, TRANSFER or UPDATE messages.

9.2. Acknowledge-Timeout

This indicates the time after which a SN may send a "reminder" 200 OK message to the Proxy, soliciting a COMMIT. After Retransmission Counter retries of the 200 OK, the SN would consider the transaction cancelled and initiate rollback operations.

9.3. Register-Timeout

This indicates the maximum time for which a Proxy will honor a registration. The SN must register with the Proxy within that time. If REGISTER is not received before timer expiry, the Proxy SHOULD mark the SN and its services unavailable. On receiving a REGISTER that matches its Capabilities, the Proxy shall send a 200 OK.

9.4. Advertise-Timeout

This indicates the time after which a Proxy will send ADVERTISE. If the Proxy indicated time period for the ADVERTISE has expired for a successive 3 times, a SN MAY transmit a DISCOVER requesting for Proxies that can cater to its service domains. This timer SHOULD be set by the Advertisement-Timeout header in the ADVERTISE message.

9.5. Cancel-Timeout

This timer is started after sending any request and reset after the receipt of any provisional or final response (such as 100 Trying or 200 OK). If a provisional or final response is not received, the sender MUST transmit the request a total of Retransmission Counter times. If a provisional or final response is still not received, the sender SHALL send a CANCEL and will terminate the transaction.

This timer SHOULD be set by the Cancel-Timeout header in the ADVERTISE message and it may be overridden on per transaction basis in the CREATE, DELETE, TRANSFER and UPDATE messages.

9.6. Commit-Timeout

This timer will be started by a responder after sending a 200 OK on transactions where a COMMIT is expected (CREATE, DELETE, TRANSFER, etc). After the timer expires, a 200 OK will be retransmitted up to a max of Retransmission Counter times. When the Commit-Timeout has expired Retransmission Counter times (after sending a 200 OK) the responder SHALL silently discard (or rollback) the changes it had performed earlier.

This timer SHOULD be set by the Commit-Timeout header in the ADVERTISE message and it may be overridden on per transaction basis in the CREATE, DELETE, TRANSFER and UPDATE messages.

9.7. Discover-Timeout

This timer will be started by a SN immediately after startup or after 3 successive timeouts of the Advertise-Timer (the Proxy has not sent

ADVERTISE 3 successive timeouts). When the Timer expires, the SN SHALL transmit the DISCOVER message. The timer SHALL be reset upon receipt of a ADVERTISE that matches the SN's SDN. This time SHALL have a fixed value of 15 seconds, because the timer is required to discover a Proxy and prior to receiving an ADVERTISE.

9.8. Publish-Timeout

This timer defines the time period within which a SN must send its service updates, regardless of any change in the service status. This will refresh a Proxy's service database, and prevent stale information. This timer SHOULD be set by the Publish-Timeout in the ADVERTISE message.

```
  Publish-Timeout: 500
```

9.9. Subscribe-Timeout

This timer defines the time period within which a WS or TS must send a SUBSCRIBE requesting for updates on SN or Client availabilities. This will refresh a Proxy's subscribe database, and prevent stale information. This timer SHOULD be set by the Subscribe-Timeout in the ADVERTISE message. Alternately

```
  Subscribe-Timeout: 5000
```

9.10. Retransmit-Timeout

This timer defines the time period within which a WS or TS must send a 100 TRYING to acknowledge a request. If the 100 TRYING response is not received within this time, the sender SHOULD retransmit the request up to a maximum of NRetransmit times. This timer SHOULD be set by the Subscribe-Timeout in the ADVERTISE message.

```
  Subscribe-Timeout: 5000
```

10. Message-Header Relationships

This section describes which messages depend on which headers. The following convention is used. "M" denotes "mandatory", "O" denotes "optional" and "I" denotes "ignore message" (drop it if it is seen).

Header Name	A	C	C	C	D	G	D	P	R	S	T	U	W	1	2	3	4	5	6
	D	A	O	R	E	E	I	U	E	U	R	P	O	X	X	X	X	X	X
	V	N	M	E	L	T	S	B	G	B	A	D	R	X	X	X	X	X	X
	E	C	M	A	E		C	L	I	S	N	A	K						
	R	E	I	T	T		O	I	S	C	S	A	F						

	T	L	T	E	E		V	S	T	R	F	T	L						
	I						E	H	E	I	E	E	O						
	S						R		R	B	R		W						
	E									E									
Acknowledge-Timeout	O	I	I	O	O	I	I	I	I	I	O	O	I	I	I	I	I	I	I
Altenate-Proxy	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	O	I	I	I
Content-Length	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
Cancel-Timeout	O	I	I	O	O	I	I	I	I	I	O	O	I	I	I	I	I	I	I
Commit-Timeout	O	I	I	O	O	I	I	I	I	I	O	O	I	I	I	I	I	I	I
Content-Type	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
Distance	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
Destination	I	I	I	I	I	I	I	I	O	I	O	I	I	I	I	I	I	I	I
Exchange	O	M	M	M	M	M	O	M	M	M	M	M	M	M	M	M	M	M	M
From	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Node-Type	I	I	I	I	I	I	I	I	M	M	M	I	I	I	I	I	I	I	I
Priority	I	I	I	O	O	I	I	I	I	I	O	O	I	I	I	I	I	I	I
Publish-Timeout	O	I	I	O	O	I	I	I	I	I	O	O	I	I	I	I	I	I	I
Retry-Count	O	I	I	O	O	I	I	I	I	I	O	O	I	I	I	I	I	I	I
Query-Type	I	I	I	I	I	I	M	I	I	I	I	I	I	I	I	I	I	I	I
Requestor	I	I	M	M	M	I	I	I	I	I	M	O	I	I	I	I	I	I	I
Registration-Timeout	M	I	I	O	O	I	I	I	I	I	O	O	I	I	I	I	I	I	I
Service-ID	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	O	I	I	I
Sequence-ID	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
Source	I	I	I	I	I	I	I	I	I	I	I	M	I	I	I	I	I	I	I
Subscribe-Timeout	O	I	I	O	O	I	I	I	I	I	O	O	I	I	I	I	I	I	I
Task-ID	I	I	O	O	O	O	I	I	I	I	O	O	I	I	O	I	I	I	I
Timestamp	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O	O
To	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M	M
Via	O	M	M	M	M	M	O	M	M	M	M	M	M	M	M	M	M	M	M
Workflow-ID	I	I	O	I	I	O	I	I	I	I	I	I	O	I	O	I	I	I	I
Workflow-Name	I	I	I	I	I	I	I	I	I	I	I	I	I	M	I	I	I	I	I
Workflow-Server	I	O	O	O	O	I	I	I	I	I	O	O	O	I	I	I	I	I	I

11. Security Considerations

Text-based messages can be snooped or modified in transit. Encryption and authentication are essential with SOP. SOP authentication can be done along same lines as SIP [SIP]. Similarly, SOP can use IPSec, TLS or other encryption and privacy mechanisms already being used with HTTP. A description of these mechanisms is left to a later effort.

12. IANA Considerations

An IANA port will be requested for the SOP destination port. A request to IANA to register SOP headers will be sent when the specification is closer to being finalized.

13. Conclusions

SOP describes a formal, extensible and flexible framework for orchestrating services in an interoperable way. Using SOP it should be possible to define new services and service domains, and advertise and consume them across provider and consumer boundaries.

Interoperable ways of creating, delivering and consuming services is essential for cloud. We envision that SOP can be an essential ingredient of Cloud Proxies and Border Gateways, to exchange services across multiple private, public, hosted, community and other clouds.

14. References

14.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

14.2. Informative References

[NIST] DRAFT Cloud Computing Synopsis and Recommendations http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf

[REQT] Service Orchestration Protocol Requirements http://www.ietf.org/id/draft-dalela-orchestration-00.txt

[ARCH] Service Orchestration Protocol Network Architecture http://www.ietf.org/id/draft-dalela-sop-architecture-00.txt

[SDF] Service Description Framework http://www.ietf.org/id/draft-dalela-sdf-00.txt

15. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Ashish Dalela
Cisco Systems
Cessna Business Park
Bangalore
India 560037

Email: adalela@cisco.com

Mike Hammer
Reston
Virginia
USA 20190

Email: mphmmr@gmail.com

Monique Morrow
Cisco Systems [Switzerland] GmbH
Richistrasse 7
CH-8304
Walllisellen
Switzerland

Email: mmorrow@cisco.com

Peter Tomsu
Cisco Systems Austria GmbH
30 Floor, Millennium Tower
Handelskai 94-96
A-1200   Vienna
Austria

Email: ptomsu@cisco.com