

Internet Draft
Expiration: January 2001
File: [draft-ghani-odsi-cops-00.txt](#)

N. Ghani
Sorrento Networks
Z. Zhang
Sorrento Networks
L. Zhang
Sorrento Networks
J. Fu
Sorrento Networks

COPS Usage for ODSI

July 5, 2000

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

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

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

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

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

Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2000). All Rights Reserved.

Abstract

ODSI is a framework of protocols designed to allow internetworking devices to dynamically request bandwidth from optical networks. Within this framework, a protocol is required for policy provisioning inside the optical domain. This document defines the application of the IETF Common Open Policy Service (COPS) protocol for this purpose and details its interworking with the ODSI framework.

[1](#). Introduction

The IETF Common Open Policy Service (COPS) protocol is a well-defined policy provisioning protocol. Owing to its generic specification, COPS can also be applied within the optical networks space for authentication and resource control functions. This is a crucial requirement for optical networks and will allow operators to specifically engineer policies to meet their operational requirements. Along these lines, this document describes the application of the COPS protocol within the ODSI framework. A brief introduction to the COPS protocol is first presented, and subsequently, its interworking with the ODSI protocols framework is detailed.

2. Review of the COPS Protocol

A very brief outline of the COPS protocol framework is first given. However, this summary is only intended to serve as a background reference, and interested readers are referred to the various protocol specification documents [[Ref1](#), [Ref2](#),[Ref6](#)] for full details.

2.1 Functional Overview

COPS is a query/response protocol based upon a client/server model and has been designed for policy and admission control for a generic set of network resources. Although it is being developed by the IETF RAP (Resource Allocation Protocol) group, owing to its inherently generalized design, it can clearly be applied under a much broader context. The framework defines client entities, termed Policy Enforcement Points (PEPs), and server entities, termed Policy Decision Points (PDPs), which exchange policy information through various message types. At least one PDP entity has to be defined for an administrative domain. The messages include request, update, and delete messages from the PEP and decision and update messages from the PDP server (see [Section 2.2](#) also). COPS has been designed to support multiple policy clients [[Ref2](#)], and examples include quality of service (QoS), security, customer-specific, etc. Therefore, to distinguish between different client types, a client type must be specified in each message. Each client type can have its own specific data and policy decision rules. Note that a single PEP or PDP can support policy provisioning for multiple client types.

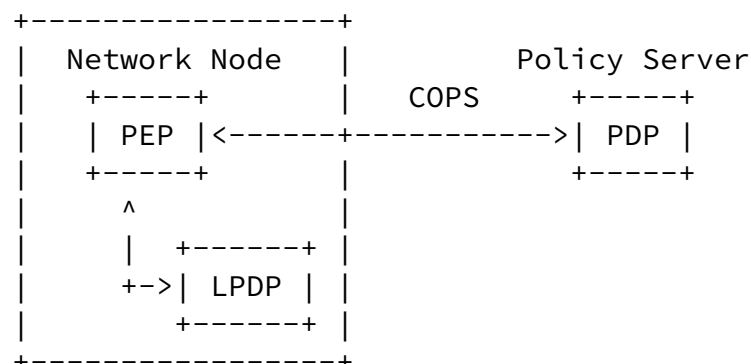


Figure 1: A COPS illustration (from [[Ref1](#)]).

The overall interaction between the respective COPS protocol entities is shown in Figure 1 (from [\[Ref1\]](#)). Specifically, the COPS protocol message set details the information exchange between the client PEP and remote PDP server. Additionally, an optional Local Policy Decision Point (LPDP) entity can also be associated with a network device to execute "local" policy decisions. However, all local decision information must be relayed to the PDP also (i.e., by the PEP in the form of an LPDP decision object), and the PDP entity remains as the binding decision point for all PEP requests. A PEP client communicates with its PDP server to implement various policy decision frameworks. As a network client, the PEP initiates this communication by opening a persistent (two-way) TCP connection to the PDP, and this connection is used for all communications between these two entities. TCP is chosen due to the important nature of the policy control information carried. This choice precludes the need for any additional mechanisms for reliable communication.

COPS can function in two basic models, outsourcing and provisioning [\[Ref2\]](#) and defines various objects for its functioning, see [\[Ref1\]](#). In the outsourcing model represents a client-driven approach, where the PDP server actively responds to incoming policy requests from PEP clients. This model works well for signaled protocols, e.g., as in RSVP-COPS [\[Ref6\]](#). Client requests are carried via COPS-PR defined data objects, which communicate Client Specific Information objects [\[Ref7\]](#). The format of this data is independent of the actual messaging protocol, and hence this decouples the information model from the actual signaling semantics. For example, RSVP-COPS requires all RSVP Path message data to be directly encapsulated into associated COPS message types (see [\[Ref6\]](#) for details). Examples of various RSVP PEP-PDP message exchange sequences are also detailed in [\[Ref6\]](#).

Meanwhile, the provisioning model represents a server-driven approach, where the PDP downloads (pre-provisions) policy information to client PEPs beforehand. Such policy information is based upon predicted configurations/demands and works well for non-signaled protocols/scenarios. Specifically, after the PEP-PDP connection is setup, the PEP sends a configuration request message to the PDP, which in turn replies with a message containing all provisionable policies for the PEP device. Alternatively, the PDP can also asynchronously "push" policy state onto the PEP. Meanwhile, the actual COPS policy data is represented via a named "self-identifying" data structure, termed the Policy Information Base (PIB) [\[Ref2\]](#). This structure specifies the type and purpose of the policy information arriving from the PDP, and hence must also be client specific. This information is installed by the PEP. Conceptually, PIB can be described using a tree structure, consisting of Policy Rule Classes (PRCs) and their associated Policy Rule Instances (PRIs), see [\[Ref2\]](#). The overall PIB concept is very flexible, and the class/rule definitions can be further modified between PEP/PDP nodes to reflect new policy requirements, e.g., expiration quotas, time-of-day restrictions, and other SLA details (see [\[Ref2\]](#)).

In the provisioning model, the PDP can re-issue updated policies to PEPs at any time, i.e., asynchronous updates. After any installation/deletion of such configuration data from the PDP, PEP nodes must send appropriate report messages to the PDP for accounting and monitoring purposes.

COPS relies upon a state-based model, where the request/decision state is shared between the PEP and PDP. Specifically, PEP requests are stored on the PDP until explicitly deleted (signaled) from the PEP. These stored requests can have an impact on how the PDP servers process future requests (i.e., based upon current request/decision states). PEP clients whose request states change must notify their PDP servers to the effect and delete non-applicable state information. However, PDP servers can also issue unsolicited messages (e.g., state updates) to PEP clients in order to force existing states. This may be necessary during policy upgrades or SLA changes. Depending upon the client type, multiple states can be installed for a single PEP/PDP relationship (as identified by the Handle object).

Due to the important role of the COPS protocol, fault-tolerance features are also provided. These capabilities allow to "re-synchronize" state between the PEP and PDP in the event of a communications failure. The PEP-PDP connection is constantly verified using keep-alive messages, and upon failure detection, the PEP falls back to local decisions (via an LPDP). While disconnected from the PDP, the PEP tries to reconnect to the original PDP and/or to an alternative PDP [[Ref1](#)]. Upon connection re-establishment, the PEP must provide the PDP with new state and/or event information. Furthermore, the PDP can resynchronize all the PEP's internal state. See [[Ref2](#)] also for additional details on fault tolerance. Finally, security provisions for COPS are provided via an Integrity object, and all COPS implementations must support this object. Additionally, client (PEP) and server (PDP) entities can also use IP Security [[Ref8](#)] mechanisms to authenticate and secure the communications channel between the PEP and the PDP entities, as described in [[Ref1](#)].

[2.2](#) Message Set Summary

A very brief summary of the COPS message set is presented to give a high-level view of some of the protocol's internals. Interested readers are referred to [[Ref1](#)], [[Ref2](#)] for more details:

- o Request (REQ) (PEP-to-PDP): PEPs send REQ messages to request policy provisioning data from the PDP. This message includes client-specific information to assist the PDP and also a unique Client Handle (to identify request states at the PEP and PDP entities).

- o Decision (DEC) (PDP-to-PEP): PDPs respond to client REQ messages via DEC messages. These messages contain multiple policy decisions that are to be installed on the PEP. DEC messages can be used to delete/update existing rules (i.e., state) in the PEP. DEC messages use the

lient Handle to identify the REQ being responded to. These messages contain command codes which instruct various operations on client-specific PIB entities.

- o Report State (RPT) (PEP-to-PDP): This message is sent from the PEP to the PDP to report accounting information. Also, a PEP must always report back to the PDP the outcome of action taken for an earlier DEC (install) message, i.e., indicating success or failure in applying the configuration action [[Ref1](#)].

- o Delete Request State (DRQ) (PEP-to-PDP): This message is sent to the PDP in order to delete a request (or related information) pertaining to the specified Client Handle. PEPs can issue DRQ messages in response to status changes on their end (e.g., connection requests withdrawn by clients), and the PDP will take appropriate actions to remove state. It is important for PEPs to issue this command to ensure state consistency with the PDP.

- o Synchronize State Request (SSQ) (PDP-to-PEP): This message requests the PEP to re-send state information. The Client Handle field here is optional, and if specified, only the respective state needs be conveyed (via a PEP RPT message). If, however, no Client Handle is specified, the PEP must send all its state information to the PDP.

- o Client-Open (OPN) (PEP-to-PDP): This message is sent to the PDP server to specify various PEP-related information: client-types supported, last PDP connected to per client type, etc. Multiple such messages can be sent at anytime. PEP clients normally send an OPN message after connection establishment with a PDP server.

- o Client-Accept (CAT) (PDP-to-PEP): In response to the PEP OPN message, the PDP can respond "positively" with a CAT message. This reply contains various timer values which are used to generate keep-alive and/or accounting report messages.

- o Client-Close (CC) (PEP-to-PDP, PDP-to-PEP): This is a bi-directional message and indicates that a particular client type is no longer supported. This can be sent from the PDP in response to a PEP OPN message, in which case an alternative PDP server may also be specified.

- o Keep-Alive (KA) (PEP-to-PDP, PDP-to-PEP): These messages are transmitted by the PEP in order to maintain the PEP-PDP connection (i.e., independent of any particular client-type). Associated timer values for this message are specified by the PDP (e.g., via CAT response). The PDP must echo a keep-alive ACK message per incoming PEP keep-alive message.

- o Synchronize State Complete (SSC) (PEP-to-PDP): This message is sent by the PEP after receiving a PDP SSQ message, and indicates that (state) synchronization between the PEP and PDP is complete. This

message may or may not include a Client Handle (as per the original PDP SSQ request message).

3. COPS Applied to the ODSI Framework

Clearly, COPS provides a very generic policy control framework that is very extensible to the optical networking domain. With recent trends towards IP-based control for optical networks, the use of this framework will further provide for a more seamless interworking with P-controlled devices. Moreover, the built-in reliability and security features of this protocol ensure its applicability to robust services-provisioning scenarios. Hence, it is logical to interwork the COPS and ODSI frameworks together in order to provide a comprehensive policy provisioning setup for optical networks and extend optical policy provisioning closer to the network edge. In particular, the ODSI interface on edge optical devices must communicate with a COPS PEP entity in order to relay policy provisioning request/responses messages. Since the PEP deals with optical network policy administration, it must reside in an edge optical device. This overall framework is illustrated more clearly in Figure 2, where the PEP client communicates with the ODSI interface.

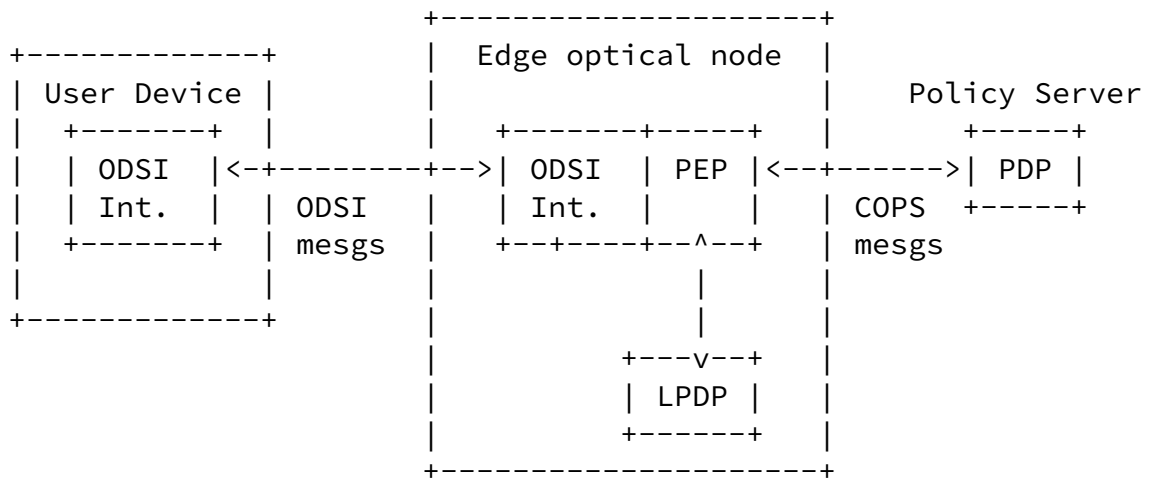


Figure 2: COPS applied to the ODSI framework

Initially, the ODSI user device-to-optical interface is setup by a series of procedures (service discovery, address registration, signaling, see [Ref3],[Ref4],[Ref5]). After this procedure is complete, the COPS startup procedures can be initiated. In the above interworking, the ODSI interface residing in the edge optical device passes messaging between the (optical domain) PEP and the (electronic domain) user device, e.g., such as an IP router or SONET digital cross-connect. In particular, this entity translates ODSI actions into appropriate PEP client actions. Subsequently, the optical PEP entity sends back appropriate reply messages to the ODSI interface. Here, the outsourcing policy management model is applied for COPS-ODSI interworking (akin to RSVP-COPS interworking [Ref6]), and ODSI actions are transmitted to the PDP via the local PEP entities. In this case,

the PDP (LPDP) returns policy responses to the PEP, which are in turn translated into appropriate ODSI interface actions and/or messages. Therefore, when PEPs open communications with PDP servers, their initial Client-Open (OPN) messages have the client-type set to indicate a signaled client. In particular, a new signaled client type ODSI is defined. If the PDP supports this client-type, it responds with a Client-Accept (CAT) message or otherwise with a Client-Close (CC) message. Alternatively, the PDP can also redirect the PEP to other PDPs in the network domain (via a Redirect address in the CC message). After receiving a CAT message, the PEP can issue the first (ODSI-driven) request message to the PDP server.

Meanwhile, the information transfer model assumes that all objects received in the ODSI messages are encapsulated in a Client Specific Information Object (Signaled ClientSI) and sent from the PEP to the PDP. Specifically, ODSI trail request messages will contain topological information (source, destination IP addresses and port numbers), bandwidth requirements, protection levels, preemption priorities, and setup priority. It assumed that the (ODSI policy server) PDP is fully ODSI-aware and can handle these message contents and generate appropriate policy control decisions.

ODSI clients are divided into user groups, and bandwidth creation is strictly restricted to (client) user devices belonging to the same ODSI user group [[Ref5](#)]. In particular, "admission control" is applied based upon the user group and requester and endpoints (as per functional specification [[Ref1](#)]). Hence, when fielding bandwidth requests, ODSI components must first resolve the IP addresses of the endpoints and resolve the associated user group value. This information, along with the details of the ODSI bandwidth request is then forwarded to the COPS policy server (i.e., optical network PDP), which will perform the necessary policy provisioning and accounting procedures on the request. Also note that the ODSI specification states that (electronic client) endpoints may be partially specified [[Ref3](#)]. By this it is meant that the complete endpoint information may not be given (i.e., port numbers). However, the endpoint IP addresses are always specified, and user group identification should be possible with this information alone.

As per the ODSI signaling specification [[Ref4](#)], the request is propagated from (trail) requester to (trail) head to (trail) tail, and then to an optical network controller (ONC) entity. The ONC validates the request and subsequently allocates the resources for the circuit. Now clearly, one of these ODSI entities must initiate policy provisioning via COPS. To functionally decouple the ODSI and COPS entities, it is felt here that such policy requests are best done by the trail head entity, i.e., PEP entity in Figure 2 only provisions policy for electronic client nodes connected to its optical device. Since this point in the network will source the request, it is logical for the PEP entity associated with the trail head to request policy clearance from the network PDP server. If the policy requirements are

valid, the request can be further propagated to the trail tail. Otherwise, the request must be rejected (by sending an ODSI trail notification message to the trail requester). In the case the trail head does not have ODSI-COPS interworking (i.e., no PEP entity), the request must again be rejected. In the following section, the message format will be specified in more details.

[4. Message Contents](#)

The COPS protocol provides the capability for different COPS clients to define their own "named", i.e. client-specific, information for various messages. This section describes the messages exchanged between a COPS server (PDP) and COPS ODSI clients (PEP) that carry client-specific data objects. Since ODSI is a signaled client, a new signaled client-type value is defined (TBD) for the common header field.

[4.1. Request \(REQ\) PEP -> PDP](#)

The REQ message is sent by COPS-ODSI clients to issue a 'Trail Request' to the PDP. The Trail Request REQ is used to request a new trail, to modify or non-destructively modify a previously established trail, to query and to release a trail. The Client Handle associated with the REQ message originated by a client must be unique for that client. The Client Specific info object is used to specify the requested resources and the request identifier.

Each REQ message contains a single request. The PDP responds to the resource allocation request with a DEC message containing the answer to the query. The REQ message has the following format:

```
<Request> ::= <Common Header>
               <Client Handle>
               <Context >
               [<ClientSI: all objects in ODSI Trail Create Request >]
               [<LPDPDecision(s)>]
               [<Integrity>]
```

The only ODSI message type supported by COPS is Trail Create Request (actions: create, destroy, query, modify, indicated by the _action indicator flag_). The other two ODSI messages, Trail Create Acknowledge and Trail Notification, are not supported by COPS.

All objects that were received in the ODSI Trail Create Request are encapsulated inside the Client Specific Information (ClientSI) Object sent from PEP to the remote PDP.

```
<ClientSI: Request ID
    all objects in Trail Create Request>
```

If the request is a Modify request, the previous value for starting time should be appended at the end of ClientSI object (for accounting

purpose):

```
<ClientSI: trail attribute data >:: Request ID
                                   Trail Create Request ODSI message
                                   <starting time>          (new value)
                                   <starting time>          (old value)
```

The LPDPDecision can be specified if applicable.

[4.2.](#) Decision (DEC) PDP -> PEP

The DEC message is sent from the PDP to a COPS-ODSI client in response to the REQ message received from the PEP. Unsolicited DEC messages cannot be sent for this client type (therefore the solicited decision flag is always set). The Client Handle must be the same Handle that was received in the REQ message.

PDP performs admission control for Trail Create Request based on the User Group Id, and some other criteria. The Decision object will contain in the Client Specific info the Request ID, which is needed by the PEP to correlate the reply with the query. Each DEC message contains a single decision. The DEC message for the COPS-ODSI client type has the following format:

```
<Decision Message> ::= <Common Header>
                        <Client Handle>
                        <Decision> | <Error>
                        [<Integrity>]
```

The decision object in turn has the following format:

```
<Decision> ::= <Context>
               <Decision: Flags>
               <Decision: Client SI data>
```

The context object will be the same as contained in the REQ message. The Decision: Flags object will contain the answer in the Command-code field according to the COPS specifications. In particular the Command-code will be "Install" to mean a positive answer and "Remove" to mean a negative answer. The following text clarifies how Install and Remove Decisions map into the different request types.

```
REQ (_action_indicator flag_ = Add)
    DEC (Dec Flag = Install) -> The requested resources are
                                successfully allocated

    DEC (Dec Flag = Remove)  -> The requested resources are not
                                allocated
```

```
REQ (_action_indicator flag_ = Release)
```

DEC (Dec flag = Install) -> The resources are released

DEC (Dec Flag = Remove) -> The resources are still allocated.

REQ (_action_indicator flag_ = Modify)

DEC (Dec flag = Install) -> The modification is accepted. The newly requested resources are allocated, while the previous ones have been released

DEC (Dec Flag = Remove) -> The modification is not accepted. Previous allocation is still active.

REQ (_action_indicator flag_ = Query)

DEC (Dec flag = Install) -> The request for query is accepted.

DEC (Dec Flag = Remove) -> The request for query is not accepted.

The Error object is used to communicate COPS protocol error to the PEP, according to the definition in [\[Ref1\]](#). No client specific error sub-codes are used by COPS-ODSI.

The Decision ClientSI data object carries the information needed to correlate the decision with the answer and some optional information to explain negative Decisions. It has the following format:

```
<Decision: Client SI data> ::= <Request ID>
                               <Reject Reason>
```

Possible reasons are:

Resource unavailable

Unsupported resource type

Unacceptable source address

Unacceptable destination address

No report is sent by the PEP to confirm the reception of a Decision message. Only in case of specific errors, the PEP will send back a Report State message to the PDP.

[4.3.](#) Report State (RPT) PEP -> PDP

For COPS-ODSI client type, the Report State message is sent by the PEP to the PDP in case of problems with a received Decision message. More specifically it is used to communicate that the Decision contains a Request identifier which cannot be correlated to a previous request. This event is the manifestation of abnormal behavior. On reception of a Report State message the PDP could start a Synchronization procedure. The RPT message for the COPS-ODSI client type has the following format:

```

<Report State Message> ::= <Common Header>
                             <Client Handle>
                             <Report Type>
                             <ClientSI: Request ID>
                             [<Integrity>]

```

[4.4.](#) Synchronize State Request (SSQ) PDP -> PEP

The Synchronize State Request message is sent by the PDP to the PEP to "reset" the state information. It requests the PEP to send the set of resource allocation REQ messages needed to rebuild the state. The SSQ can apply to the whole set of PEP active reservations PEP, or to a specific resource type and source-destination couple, depending on the information contained in the Client SI object.

```

< Synchronize State> ::= <Common Header>
                          <Client Handle>
                          <ClientSI: SSQ scope>]
                          [<Integrity>]

```

[4.5.](#) Synchronize State Complete (SSC) PEP -> PDP

The Synchronize State Complete message is sent by the PEP to the PDP to inform that all the REQ messages needed to rebuild the state have been sent. The Client SI object is the same received in the SSQ message and specifies the scope of the synchronization procedure which has been completed.

```

< Synchronize State Complete> ::= <Common Header>
                                   <Client Handle>
                                   <ClientSI: SSQ scope>]
                                   [<Integrity>]

```

[5.](#) Illustrative Examples

In this section, some illustrative examples for the COPS-ODSI client interworking are presented. In the first example, the trail request is successful, and in the second example the request is rejected.

A PEP requests an OC48 trail between Head (192.234.124.1) and Tail (192.234.4.1).

```

PEP -- > PDP   REQ: = <Handle C>
                      <Context: Trail Request, Add>
                      <ClientSI:
                        Request ID:          25
** start of ODSI Trail Create Request message **
                        User Group:          X
                        Source Address:      192.234.124.1
                        Destination address: 192.234.4.1
                        Bandwidth type:      OC48

```

```
    ** end of ODSI Create message ***
        Starting time:      0.0.0
    >
```

The PDP accepts the request.

```
PDP--- > PEP      DEC: = <Handle C>
                        <Context: Trail Request, Add>
                        <Decision: Command = Install>
                        <Decision: Client SI
                            Request ID  25
                        >
```

In the following example, the trail request of an OC192 between Head (192.234.1244.1) and Tail (192.234.4.1) is rejected. The same handle and user group are used but the Request ID is different.

```
PEP -- > PDP      REQ: = <Handle C>
                        <Context: Trail Request, Add>
                        <ClientSI:
                            Request ID:      28
    ** start of ODSI Trail Create Request message **
                            User Group:      X
                            Source Address:   192.234.124.1
                            Destination address: 192.234.4.1
                            Bandwidth type:   OC192
    ** end of ODSI Create message ***
                            Starting time:    0.0.0
                        >
```

```
PDP--- > PEP      DEC: = <Handle C>
                        <Context: Request Request, Add>
                        <Decision: Command = Remove>
                        <Decision: Client SI
                            Request ID  28
                            Reason code: resource unavailable
                        >
```

5. References

- [Ref1] Durham, D., Boyle, J., R. Cohen, S. Herzog, R. Rajan, A. Sastry, "The COPS (Common Open Policy Service) Protocol", IETF [RFC 2748](#), January 2000.
- [Ref2] Chan, K., Durham, D., Gai, S., Herzog, S., McCloghrie, K., Reichmeyer, F., Seligson, J., Smith, A., Yavatkar, R., "COPS Usage for Policy Provisioning", IETF Draft [draft-ietf-rap-pr-02.txt](#), March 10, 2000.
- [Ref3] ODSI Coalition, G. Bernstein, R. Coltun, J. Moy and A. Sodder, editors, "Optical Domain Service Interconnect

(ODSI) Functional Specification", March 2000.

[Ref4] Bernstein, G., Coulton, R., Moy, J., Sodder, A.,
"Optical Domain Service Interconnect (ODSI) Signaling
Specification", April 2000.

[Ref5] Bernstein, G., Coulton, R., Moy, J., Sodder, A.,
Arvind, K., "ODSI Service Discovery and Address
Registration," April 2000.

[Ref6] Herzog, S., Boyle, J., Cohen, R., Durham, D., Rajan,
R., Sastry, A., "COPS Usage for RSVP," IETF [RFC 2749](#),
January 2000.

[Ref7] Durham, D., Khosravi, H., Weiss, W., Avri, D., "COPS
Usage for AAA," IETF Draft [draft-durham-aaa-cops-ext-00.txt](#),
May 2000.

[Ref8] Atkinson, R., "Security Architecture for the Internet
Protocol", [RFC 2401](#), August 1995.

6. Authors' Information

Nasir Ghani
Sorrento Networks Inc.,
9990 Mesa Rim Road
San Diego, CA 92121
Phone: (858) 646-7192
Email: nghani@sorrentonet.com

Zhensheng Zhang
Sorrento Networks Inc.,
9990 Mesa Rim Road
San Diego, CA 92121
Phone: (858) 646-7195
Email: zzhang@sorrentonet.com

Leah Zhang
Sorrento Networks Inc.,
9990 Mesa Rim Road
San Diego, CA 92121
Phone: (858) 450-4977
Email: leahz@sorrentonet.com

James Fu
Sorrento Networks Inc.,
9990 Mesa Rim Road
San Diego, CA 92121
Phone: (858) 450-4924
Email: jfu@sorrentonet.com

7. Full Copyright Notice

Copyright (C) The Internet Society (1997). All Rights Reserved.

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

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

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