Network Working Group
Request for Comments: 2748
Category: Standards Track

D. Durham, Ed.
Intel
J. Boyle
Level 3
R. Cohen
Cisco
S. Herzog
IPHighway
R. Rajan
AT&T
A. Sastry
Cisco

January 2000

The COPS (Common Open Policy Service) Protocol

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

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].

Abstract

This document describes a simple client/server model for supporting policy control over QoS signaling protocols. The model does not make any assumptions about the methods of the policy server, but is based on the server returning decisions to policy requests. The model is designed to be extensible so that other kinds of policy clients may be supported in the future. However, this document makes no claims that it is the only or the preferred approach for enforcing future types of policies.

Table Of Contents

1. Introduction

This document describes a simple query and response protocol that can be used to exchange policy information between a policy server (Policy Decision Point or PDP) and its clients (Policy Enforcement Points or PEPs). One example of a policy client is an RSVP router that must exercise policy-based admission control over RSVP usage [RSVP]. We assume that at least one policy server exists in each controlled administrative domain. The basic model of interaction between a policy server and its clients is compatible with the framework document for policy based admission control [WRK].

A chief objective of this policy control protocol is to begin with a simple but extensible design. The main characteristics of the COPS protocol include:

1. The protocol employs a client/server model where the PEP sends requests, updates, and deletes to the remote PDP and the PDP returns decisions back to the PEP.

2. The protocol uses TCP as its transport protocol for reliable exchange of messages between policy clients and a server. Therefore, no additional mechanisms are necessary for reliable communication between a server and its clients.

3. The protocol is extensible in that it is designed to leverage off self-identifying objects and can support diverse client specific information without requiring modifications to the COPS protocol itself. The protocol was created for the general administration, configuration, and enforcement of policies.

4. COPS provides message level security for authentication, replay protection, and message integrity. COPS can also reuse existing protocols for security such as IPSEC [IPSEC] or TLS to authenticate and secure the channel between the PEP and the PDP.

5. The protocol is stateful in two main aspects: (1) Request/Decision state is shared between client and server and (2) State from various events (Request/Decision pairs) may be inter-associated. By (1) we mean that requests from the client PEP are installed or remembered by the remote PDP until they are explicitly deleted by the PEP. At the same time, Decisions from the remote PDP can be generated asynchronously at any time

for a currently installed request state. By (2) we mean that the server may respond to new queries differently because of previously installed Request/Decision state(s) that are related.

6. Additionally, the protocol is stateful in that it allows the server to push configuration information to the client, and then allows the server to remove such state from the client when it is no longer applicable.

1.1 Basic Model

```
+----------------+
|                |
|  Network Node  |            Policy Server
|                |
|   +-----+      |   COPS        +-----+
|   | PEP |<-----|-------------->| PDP |
|   +-----+      |               +-----+
|    ^           |
|    |           |
|    \-->+-----+ |
|        | LPDP| |
|        +-----+ |
|                |
+----------------+
```

Figure 1: A COPS illustration.

Figure 1 Illustrates the layout of various policy components in a typical COPS example (taken from [WRK]). Here, COPS is used to communicate policy information between a Policy Enforcement Point (PEP) and a remote Policy Decision Point (PDP) within the context of a particular type of client. The optional Local Policy Decision Point (LPDP) can be used by the device to make local policy decisions in the absence of a PDP.

It is assumed that each participating policy client is functionally consistent with a PEP [WRK]. The PEP may communicate with a policy server (herein referred to as a remote PDP [WRK]) to obtain policy decisions or directives.

The PEP is responsible for initiating a persistent TCP connection to a PDP. The PEP uses this TCP connection to send requests to and receive decisions from the remote PDP. Communication between the PEP and remote PDP is mainly in the form of a stateful request/decision exchange, though the remote PDP may occasionally send unsolicited decisions to the PEP to force changes in previously approved request states. The PEP also has the capacity to report to the remote PDP that it has successfully completed performing the PDP's decision locally, useful for accounting and monitoring purposes. The PEP is responsible for notifying the PDP when a request state has changed on the PEP. Finally, the PEP is responsible for the deletion of any state that is no longer applicable due to events at the client or decisions issued by the server.

When the PEP sends a configuration request, it expects the PDP to continuously send named units of configuration data to the PEP via decision messages as applicable for the configuration request. When a unit of named configuration data is successfully installed on the PEP, the PEP should send a report message to the PDP confirming the installation. The server may then update or remove the named configuration information via a new decision message. When the PDP sends a decision to remove named configuration data from the PEP, the PEP will delete the specified configuration and send a report message to the PDP as confirmation.

The policy protocol is designed to communicate self-identifying objects which contain the data necessary for identifying request states, establishing the context for a request, identifying the type of request, referencing previously installed requests, relaying policy decisions, reporting errors, providing message integrity, and transferring client specific/namespace information.

To distinguish between different kinds of clients, the type of client is identified in each message. Different types of clients may have different client specific data and may require different kinds of policy decisions. It is expected that each new client-type will have a corresponding usage draft specifying the specifics of its interaction with this policy protocol.

The context of each request corresponds to the type of event that triggered it. The COPS Context object identifies the type of request and message (if applicable) that triggered a policy event via its message type and request type fields. COPS identifies three types of outsourcing events: (1) the arrival of an incoming message (2) allocation of local resources, and (3) the forwarding of an outgoing message. Each of these events may require different decisions to be made. The content of a COPS request/decision message depends on the context. A fourth type of request is useful for types of clients that wish to receive configuration information from the PDP. This allows a PEP to issue a configuration request for a specific named device or module that requires configuration information to be installed.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.