Internet Engineering Task Force
INTERNET DRAFT
Expires: August 2001

Lily Yang
Intel Corporation
Markus Hofmann
Lucent Technologies

OPES Architecture for Rule Processing and Service Execution

draft-yang-opes-rule-processing-service-execution-00.txt

Status of this Memo

Abstract

This document describes specific architectural components of the Open Pluggable Edge Service (OPES) framework [1]. It defines the functionality of the OPES Administration Server and the OPES Service Engine and discusses the interaction with other OPES components. In particular, the document defines the operational flow for downloading rules and proxylets and the content flow for handling web requests and responses. A list of protocols and interfaces to be specified within the OPES framework is derived from the discussion.

Table of Contents

1. Introduction

The Open Pluggable Edge Services (OPES) framework described in [1] enables the creation and the provisioning of services executed on application data by participating transit intermediaries. Figure 1 shows a diagram of the main components of the framework, with two elements added û the ôProxylet Vendorö and the ôRule Ownerö. The Proxylet Vendor is the party providing the proxylet code to be executed on the OPES Engine or the Callout Server. The Proxylet Vendor is not necessarily the developer of the proxylet code (e.g. it can also be a reseller). Rule Owner and Proxylet Vendor represent two important trust parties in the overall framework.

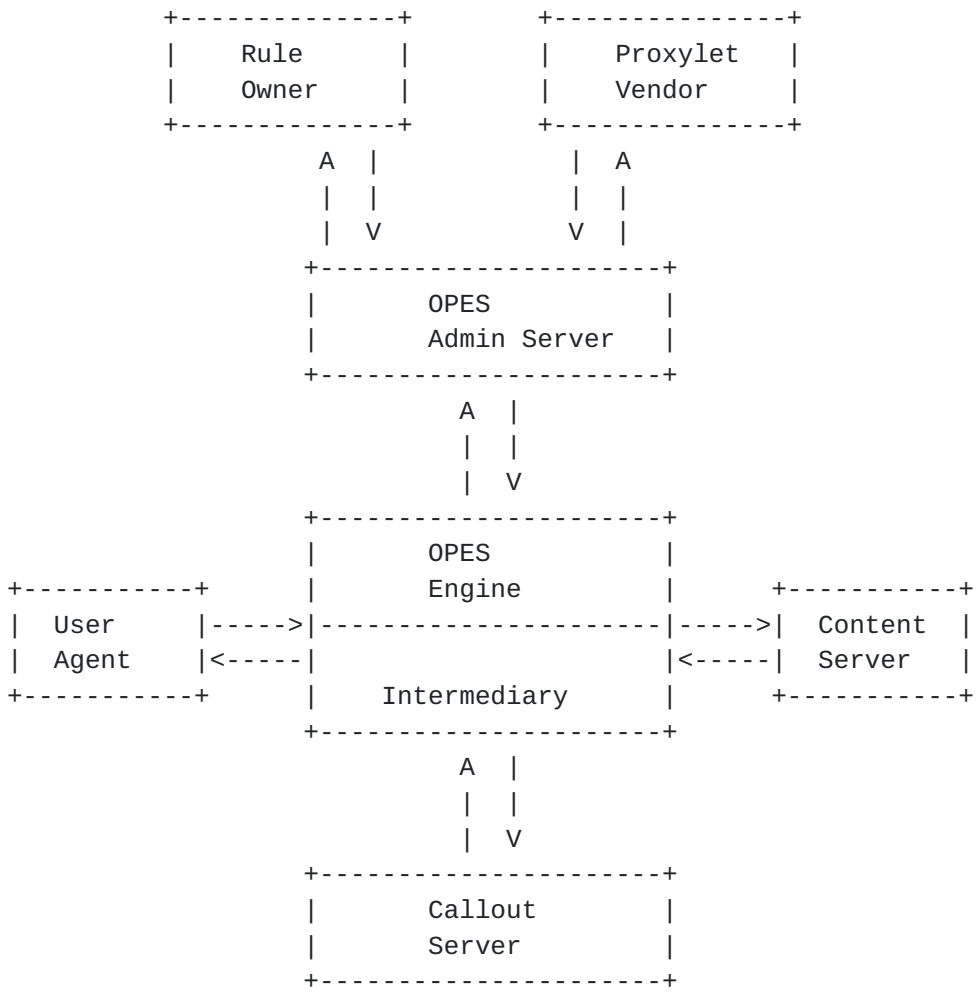


Figure 1 - OPES System Architecture Components

2. Terminology

The Terminology section of [1] provides a list of terms, many of

them being used throughout this document. Additional terms are specified in this section.

Intermediary

An intermediary is a device that is located in the middle of the client-to-server transit path and has a basic understanding of the application protocol. Caches are probably the most commonly known and used intermediaries today.

OPES Engine

An OPES engine allows new services to be defined and executed on intermediaries according to the policies set up by an OPES admin server and the rules specified by rule owners (e.g. content providers, user agents, or access providers). An OPES Engine contains a message parser, a rule processor and a service execution component that executes proxylets or makes calls to a basic proxylet library or a remote call-out server (e.g. using (ICAP).

OPES Device

An intermediary integrating an OPES engine is called OPES device. We generally use OPES device to refer to a physical intermediary that has an OPES engine built in it.

Proxylet

A proxylet is a piece of code that runs on the (local) OPES device and provides a service on the transit requests or responses. For example, a proxylet could be a piece of JAVA code that does a simple URL filtering û it allows only traffic to a short list of work-related sites during work hours.

Proxylet Library

Proxylet library is a library provided to support some basic functionality to the proxylet code programmers. The library also provides a set of commonly useful services that every OPES device would need, like simple traffic accounting functions etc.

OPES Service

An OPES service is any service that can be provided within the OPES framework on behalf of content providers, access providers or end users. The service is provided within the OPES architecture by executing code either locally on an OPES device or remotely on other service engines. Currently the services supported by OPES are either proxylets, calls to the proxylet library, or remote procedure calls like ICAP.

OPES Admin Server

An OPES admin server performs downloading of proxylets and rules from other parties at a higher trust level, authorization and authentication for services, the collection of accounting and log data, and other administrative tasks for the OPES devices.

Rule Owner

The rule owner is the party that authors the rule module. The rules specify which services have to be executed under what condition. The rule owner can be one of the three types û content providers, client, and access providers. There are certain

constraints as to which services the rule modules from a particular owner can initiate û the rule owner can only instruct on how services are invoked on its behalf. For example, a rule module provided by a content provider should only affect requests for content owned by the same content provider; and a rule module from a client agent should only affect requests from that client. If the rule owner is an access provider, it usually is the same party that owns the OPES device and hence there is no similar constraint at the rule module level for access providers.

Proxylet Vendor
Proxylet vendor is the party that provides the proxylet code to run in the OPES engine.

Proxylet Meta-Data
Proxylet meta-data describes the characteristics, features and requirements associated with a proxylet. Examples for such meta-data are the name of the proxylet, its functionality, its version number, where to get it, license related information, execution environments, etc. The meta-data can physically be separated from the proxylet code, but it must be possible to uniquely associate meta-data with proxylets and vice versa.

Content Path
The content path describes the path that content requests and responses take through the network. Typically, content requests and responses flow between a client, an OPES device, a content server and optionally a remote call-out server.

Configuration Path
Rules and proxylet code (and its associated meta-data) is downloaded into the OPES admin server from rule owners and proxylet vendors, respectively, and then distributed to the OPES devices. This flow is referred to as configuration path, as the data being transferred along this path is used to configure the OPES devices for providing the requested services.

3. Basic Assumptions

The OPES architecture shown in Figure 1 is based on certain assumptions. These assumptions are adopted in this document and have impact on the outlined architecture for rule processing and service execution. We discuss these assumptions, having in mind that most of them are still open for debate.

3.1. Administrative Domains

It is assumed that each of the components in Figure 1 may belong to a different administrative domain, except for the OPES admin server

and OPES devices, which are assumed to belong to the same administrative domain. Note, however, that the OPES admin server and OPES devices may be manufactured by different vendors.

3.2. Ownership and Deployment Scenarios

The general issue of OPES ownership is addressed in [2]. In summary, the OPES service is likely to be deployed by either the access provider (e.g. ISP or enterprise) on the behalf of their users (i.e. subscribers), or the content provider (e.g. surrogate proxies in front of the origin server farm, or edge servers in a CDN). In general, two OPES service deployment scenarios seem to be most typical:

- ISP scenario: One or multiple OPES devices deployed in the access provideræs network. ISPs are more likely to be interested in value-added services that they can resell to their subscribers and in services that would simplify their general administrative tasks.

- CDN scenario: CDN providers can deploy OPES engines on surrogates that provide CDN services on behalf of content providers (i.e. the customers of the CDN). CDN providers are likely to be more interested in services that they can offer to content providers, in particular to enable secure and profitable distribution of valuable content.

Other scenarios are not excluded, but the above-mentioned scenarios seem to be most likely and are the focus of this document. In particular, it is assumed that a single OPES admin server should be able to support multiple OPES devices, all being in the same administrative domain. We do not consider a scenario in which a single OPES device is controlled by multiple OPES admin servers. This does not exclude scenarios in which a provider deploys multiple OPES admin servers for fail-over.

3.3. Types of OPES devices

In principle, an OPES Engine can be implemented on top of any intermediary, as long as it meets certain requirements (e.g. understanding the required set of protocols and/or MIME types).

Caching proxies are probably the most commonly thought-of intermediaries when talking about OPES devices. Although their built-in caching capability can provide additional benefits, the OPES architecture does not depend on it. Other types of intermediaries, such as web switches or firewalls, can also be used as a basis for OPES devices.

3.4. Physical Boundary between OPES Admin Server and OPES Device

The OPES admin server and the OPES device represent separate logical components in the OPES architecture. While the OPES admin server is

off the content path, the OPES device is placed right in the middle of the content flow. However, no assumption is made regarding the physical boundary between these two functional components. In particular, the following physical configurations are possible:

- Appliance Model: The OPES admin server and the OPES engine/OPES device are built into a single appliance.

- Toolbox Model: The OPES admin server and the OPES device are physically separate boxes. This toolbox approach seems to be beneficial in large-scale CDN environments, where a few admin servers can administer a large number of OPES devices.

4. Loading Proxylets into OPES Devices

In general, there are two choices in loading proxylets into OPES devices. Proxylets could be requested and loaded dynamically together with the content, or they could be loaded and verified a-priori, i.e. independent from the content.

In this document, we explicitly assume that proxylets are NOT requested and loaded dynamically along with the content. Instead, they are loaded a-priori and independent from the content. In other words, the configuration path is different from the content path. It is assumed that rules and proxylets are loaded into the OPES admin server via a separate mechanism before they are transmitted to the OPES device.

5. Configuration Path: Getting Rules and Proxylets

This section addresses issues along the configuration path and identifies protocols and APIs to be specified by OPES.

5.1. Rule Loading

5.1.1. Description

Rule owners (e.g. content providers, access providers, or users) specify what kind of services should be invoked under what conditions. They specify these rules using an open, standardized rule specification language such as IRML [3]. It is assumed that the rule owner has a certain trust relationship and/or business arrangement with the owner of the targeted OPES devices.

After the rules have been specified by the rule owner using IRML, they can be loaded into the OPES admin server via any secure file transfer mechanism (e.g. secure file copy, email with PGP, etc.). Once the IRML rule is loaded into and authenticated by the OPES admin server, the OPES admin server transmits the IRML rule module to the OPES devices. As above, any secure standard transfer mechanism can be used here.

5.1.2. Protocols and Interfaces required for Rule Loading

- An open and standardized language for rule specification is needed. A standard format allows exchange of rule sets between different parties and different vendor appliances. See [3] for

a work-in-progress specification on the Intermediary Rule Markup Language (IRML).

- Although no specific transfer mechanism is required to ship rules from the rule owner to the OPES admin server and further to the OPES devices, it seems favorable to agree on a specific existing transfer mechanism within the OPES framework in order to simplify vendor interoperability.

5.2. Proxylet Loading

5.2.1. Description

A proxylet is software code to be executed on an OPES device to provide a specific service on the same OPES device. Similar to rules, the proxylet is downloaded from the proxylet vendor or proxylet owner to OPES admin servers. In addition to the authentication process, the OPES admin server also wants to perform proxylet ôsandboxö validation to make sure that the proxylet conforms to local policies (e.g. what resource it is allowed to access, etc.). Only after successful validation, the proxylet code would be loaded into the targeted OPES devices and be triggered by the rules.

Since a proxylet itself is a piece of binary code, it is necessary to have meta-data associated that describes the important features and requirements of the proxylet. For example, it is necessary to learn about the required execution environment (e.g. operating system, runtime interpreters, etc.) before loading a proxylet code into an OPES device. It might also be possible that a proxylet owner wants to specify a policy about which rule owners are allowed to call the proxylet (e.g. only rule modules from abc.com and xyz.com are allowed to call this proxylet, or any rule modules can all this proxylet, etc.). It would also be helpful to indicate the message format that the proxylet can handle (e.g. an video adaptation proxylet might require the underlying video stream to be in MPEG). This meta-data information has to be provided in a standardized format, so that different parties (e.g. rule owners, OPES device managers etc.) can access it. The meta-data can be kept separately, but it must be possible to associate meta-data with the correct proxylet and vice versa.

Since meta-data can be kept separate from the proxylet code, a standard API for querying meta-data of proxylets is required.

5.2.2. Protocols and Interfaces required for Proxylet Loading

- A standard format for proxylet meta-data. See [4] for a work-in-progress specification on the OPES Meta-data Markup Language

(OMML).

- An API for the standard library that can be used by proxylet developers.

- An API for a base class proxylet or a standard library that provides fundamental functions, such as querying meta-data of installed proxylets.

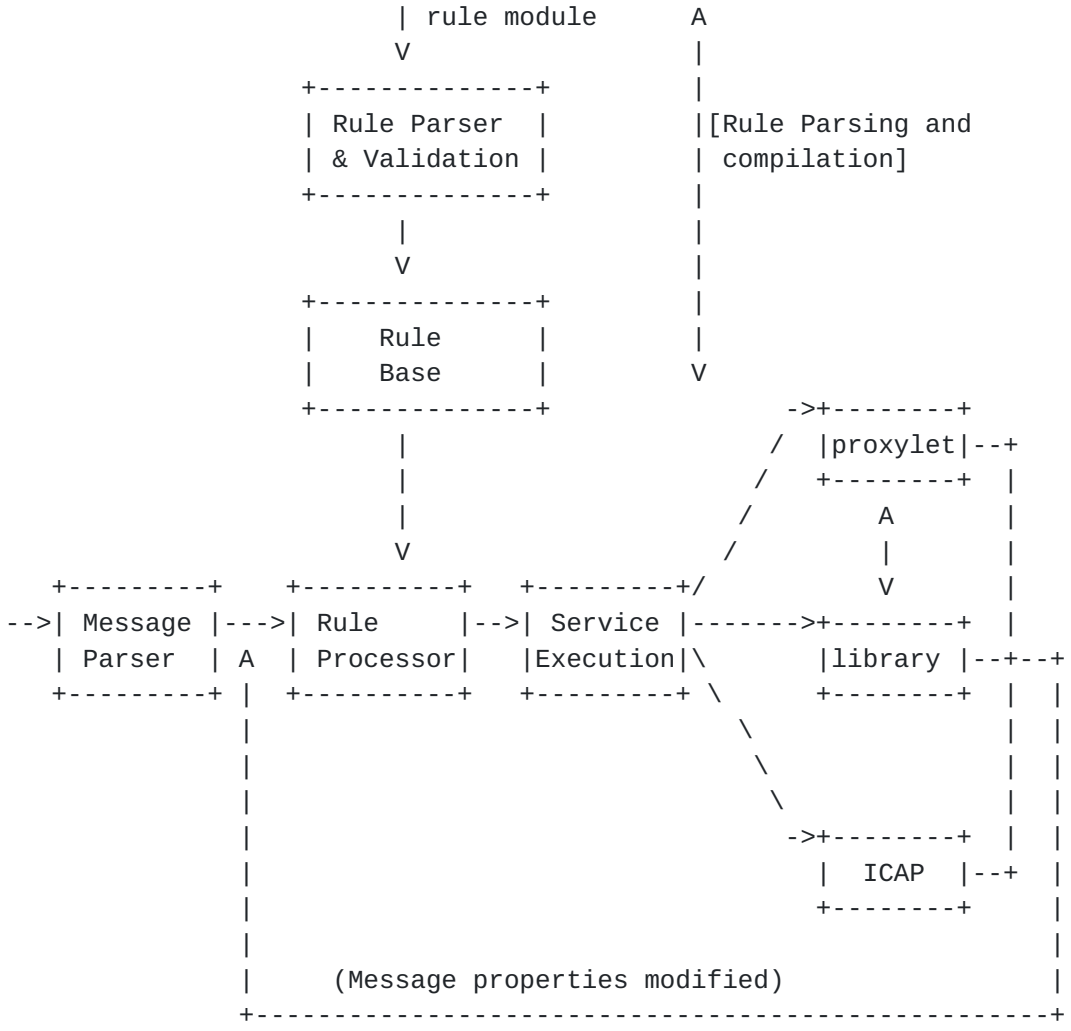
- A protocol for proxylet loading between proxylet vendor and OPES admin server if automation of the loading (i.e., no human involved) is desirable (see also comments above on loading rules).

6. Content Path: Rule Processing and Service Execution

This section describes how messages (i.e. client requests and responses) flow through the OPES service engine.

6.1. Data Flow

Figure 2 illustrates the data flow within an OPES device.



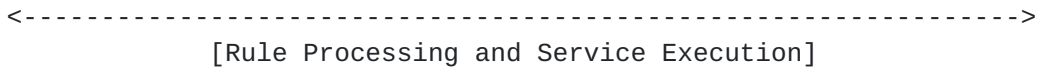


Figure 2 - Data Flow within an OPES Device

Figure 2 illustrates the separation of rule parsing and compilation from rule processing and service execution. Rule parsing and compilation is off the content path and refers to the process of generating an efficient internal representation of the rules (i.e. the rule base). Rule processing and service execution is on the content path invoking OPES services on messages as defined by the rules.

6.2. Rule Parsing and Compilation

In the rule parsing and compilation process, the rule module file is loaded (from OPES admin server) to the OPES device in IRML format, and it is parsed, validated and inserted into the rule base. The rule base is the ultimate output for the rule parsing and compilation process. It is an internal representation of all the rules accepted into the OPES rule engine. This representation is internal to the implementation of the OPES service engine.

A rule can reference a proxylet, a call to a proxylet library, or a remote callout service like ICAP as its action. So in order to validate a rule, the corresponding service (i.e. action) referenced by the rule must exist (either locally for proxylet and library, or remotely for ICAP). For local proxylet code, that means the proxylet must have passed the validation at the OPES admin server and have been loaded onto the OPES device.

6.3. Rule Processing and Service Execution

In the rule processing and service execution process, value-added services are invoked according to the rules in the rule base.

First, the Message Parser parses the incoming user requests and server responses and feed the relevant message properties (like HTTP headers for HTTP) into Rule Processor. This step might be optimized by looking at the already compiled rule base to find out what headers are relevant so only relevant headers are fed into the rule processor. The rule processor takes one rule out of the rule base at a time and attempts to match the relevant properties against the regular expression pattern specified by the rule. If a match is found, an action is fired. The Service Execution component binds the action element in the IRML rule [3] with a proxylet, a call to the proxylet library, or to the ICAP client for invoking a remote callout service. When the action is completed, the control is back to the OPES engine with a set of possibly modified message properties. The modified message properties are then fed back to the rule processor again for the next rule in the rule base. Infinite loop is avoided because each rule in the rule base is checked once and only once per request or response.

The Message Parser usually is provided as a base functionality of the intermediary. A standardized API between the Message Parser and the rule processor would help facilitate easy implementation of OPES

engine over a variety of different intermediary devices from different vendors.

6.4. Protocols and Interfaces required for Rule Processing

- It would be desirable to have a standard API to the message parser allowing for standardized access to message properties.

7. Accounting and Management

The OPES admin server has to collect information from OPES devices in order to perform accounting and billing services and to provide statistics for management purposes. (In principle, accounting and billing can also be done on a separate component, but for simplicity of this document, we consider this functionality to be integrated into the OPES admin server. This assumption does not limit the generality of this document).

OPES devices collect relevant information and save it in a standard logging format yet to be specified. The standard log files can be transferred to the OPES admin server (or any other accounting and billing system) for later processing. The transfer of log files can be done via existing file transfer protocols, although a certain level of security is desirable.

8. Security Considerations

Security is an important aspect within the OPES framework and is discussed in the section on "Security Considerations" of [1].

9. Intellectual Property

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11.

Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary

rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

10. Acknowledgments

The authors would like to thank all the active participants in the OPES mailing list for their thought-provoking discussion. Many of the ideas and suggestions have been incorporated into this document. In particular, we want to acknowledge the following people for their significant contributions: Christian Maciocco, Rob Erickson, Michael Condry, and Andre Beck.

11. References

[1] G. Tomlinson, et al., "Extensible Proxy Services Framework", Internet-Draft draft-tomlinson-epsfw-00.txt, work in progress, July 2000.

[2] R. Erickson, et al., ôOPES Ownershipö, Internet-Draft, work in progress, February 2001.

[3] A. Beck, M. Hofmann, "IRML: A Rule Specification Language for Intermediary Services ", Internet-Draft draft-beck-opes-irml-00.txt, work in progress, February 2001.

[4] C. Maciocco, M. Hofmann, "OMML: OPES Meta-data Markup Language", Internet-Draft draft-maciocco-opes-omml-00.txt, work in progress, February 2001.

12. Disclaimer

The views and specification herein are those of the authors and are not necessarily those of their employers. The authors and their employer specifically disclaim responsibility for any problems arising from correct or incorrect implementation or use of this specification.

13. Author's Address

Lily Yang
Intel Corporation
MS JF3-206
2111 NE 25th Ave.
Hillsboro, OR 97124, USA
Phone: +1-503-264-8813
E-Mail: lily.l.yang@intel.com

Markus Hofmann
Bell Labs/Lucent Technologies
Room 4F-513
101 Crawfords Corner Road
Holmdel, NJ 07704, USA

Phone: +1 732 332 5983
Email: hofmann@bell-labs.com

14. Full Copyright Statement