Network Working Group
Internet-Draft
Expires: January 31, 2003

Abbie. Barbir
Nortel Networks
R. Chen
AT&T Labs
M. Hofmann
Bell Labs/Lucent Technologies
H. Orman
Purple Streak Development
R. Penno
Nortel Networks
G. Tomlinson
The Tomlinson Group
August 2, 2002

An Architecture for Open Pluggable Edge Services (OPES)
draft-ietf-opes-architecture-03

Status of this Memo

Copyright Notice

Abstract

This memo defines an architecture for a cooperative application service in which a data provider, a data consumer, and zero or more

application entities cooperatively realize a data stream service.

Table of Contents

1. Introduction 3
2. The Architecture 4
2.1 OPES Entities 4
2.1.1 Data Dispatcher 4
2.2 OPES Flows 5
2.3 OPES Rules 6
2.4 Callout Servers 7
2.5 Tracing Facility 8
3. Security and Privacy Considerations 10
3.1 Trust Domains 10
3.2 Callout protocol 11
3.3 Privacy 11
3.4 Establishing trust 11
3.5 End-to-end Integrity 12
4. Summary 13
References 14
Authors' Addresses 14
A. Acknowledgements 16
Full Copyright Statement 17

1. Introduction

When supplying a data stream service between a provider and a consumer, the need may arise to provision the use of other application entities, in addition to the provider and consumer. For example, some party may wish to customize a data stream as a service to a consumer. The customization step might be based on the customer's resource availability (e.g., display capabilities).

In some cases in may be beneficial to provide a customization service at a network location between the provider and consumer host rather than at one of these endpoints. For certain services performed on end-user behalf this may be the only option of service deployment. In this case, one or more additional application entities may participate in the data stream service. There are many possible provisioning scenarios which make a data stream service attractive. In [1] a description of several scenarios is provided.

This document presents the architectural components of Open Pluggable Edge Services (OPES) that are needed in order to perform a data stream service. The architecture addresses the IAB considerations described in [2]. These considerations are covered in the various parts of the document, specifically with respect to tracing (Section 2.5) and security considerations (Section 3).

The document is organized as follows: Section 2 introduces the OPES architecture. Section 3 discusses security considerations. Section 4 provides a summary of the architecture and the requirements for interoperability.

2. The Architecture

The architecture of Open Pluggable Edge Services (OPES) can be described in terms of three interrelated concepts, mainly:

o OPES entities: processes operating in the network;

o OPES flows: data flows that are cooperatively realized by the OPES entities; and,

o OPES rules: these specify when and how to execute OPES intermediary services.

2.1 OPES Entities

An OPES entity is an application that operates on a data flow between a data provider application and a data consumer application. OPES entities can be:

o an OPES service application, which analyzes and possibly transforms messages exchanged between the data provider application and the data consumer application;

o a data dispatcher, which invokes an OPES service application based on OPES ruleset and application-specific knowledge.

In the network, OPES entities reside inside OPES processors. The cooperative behavior of OPES entities introduces additional functionality for each data flow provided that it matches the OPES rules.

In the current work, the data provider and data consumer applications are not considered as OPES entities. The OPES architecture is largely independent of the protocol that is used by the OPES entities to exchange data. However, this document selects HTTP [4] as the example for the underlying protocol in OPES flows.

2.1.1 Data Dispatcher

Data dispatchers include a ruleset that can be compiled from several sources and must resolve into an unambiguous result. The combined ruleset enables an OPES processor to determine which service applications to invoke for which data flow. Accordingly, the data dispatcher constitutes an enhanced policy enforcement point, where policy rules are evaluated, service-specific data handlers and state information is maintained, as depicted in Figure 1.

```
                                       +----------+
                                       |  callout |
                                       |  server  |
                                       +----------+
                                            ||
                                            ||
                                            ||
                                            ||
                        +--------------------------+
                        | +----------+      ||     |
                        | |   OPES   |      ||     |
                        | |  service |      ||     |
                        | |   appl   |      ||     |
                        | +----------+      ||     |
                        | +----------------------+ |
        OPES flow <---->| | data dispatcher and  | |<----> OPES flow
                        | |   policy enforcement | |
                        | +----------------------+ |
                        |           OPES           |
                        |         processor        |
                        +--------------------------+
```

Figure 1: Data Dispatchers

The architecture allows more than one policy enforcement point to be present on an OPES flow.

2.2 OPES Flows

An OPES flow is a cooperative undertaking between a data provider application, a data consumer application, zero or more OPES service applications, and zero or more data dispatchers.

In order to understand the trust relationships between OPES entities, each is labeled as residing in an administrative domain. Entities associated with a given OPES flow may reside in one or more administrative domains.

For example, Figure 2 depicts a data flow (also known as an "OPES flow"), that spans two administrative domains.

```
   ---------------------------------------------------------------------


    consumer administrative domain       provider administrative domain
   +------------------------------+     +------------------------------+
   |                              |     |                              |
   |       data          OPES     |     |     OPES          data       |
   |     consumer      processor  |     |   processor     provider     |
   |                              |     |                              |
   |   +----------+   +--------+  |     |  +--------+   +----------+   |
   |   |   data   |   |  OPES  |  |     |  |  OPES  |   |   data   |   |
   |   | consumer |   |service |  |     |  |service |   | provider |   |
   |   |   appl   |   |  appl  |  |     |  |  appl  |   |   appl   |   |
   |   +----------+   +--------+  |     |  +--------+   +----------+   |
   |   |          |   |        |  |     |  |        |   |          |   |
   |   |   HTTP   |   |  HTTP  |  |     |  |  HTTP  |   |   HTTP   |   |
   |   |          |   |        |  |     |  |        |   |          |   |
   |   +----------+   +--------+  |     |  +--------+   +----------+   |
   |   |  TCP/IP  |   | TCP/IP |  |     |  | TCP/IP |   |  TCP/IP  |   |
   |   +----------+   +--------+  |     |  +--------+   +----------+   |
   |        ||         ||    ||   |     |   ||    ||         ||        |
   |        =============    =================    =============        |
   |                              |     |                              |
   +------------------------------+     +------------------------------+
            | <----------------- OPES flow -----------------> |


                         Figure 2: An OPES flow

   ---------------------------------------------------------------------
```

Figure 2 depicts two data dispatchers that are present in the OPES flow. However, the architecture allows for zero or more data dispatchers to be present in any flow.

2.3 OPES Rules

OPES policy regarding services and the data provided to them is determined by a ruleset consisting of OPES rules. The rules consist of a set of conditions and related actions. The ruleset is the superset of all OPES rules on the processor. The OPES ruleset determines which service applications will operate on a data stream. The communication of data stream elements to an application is performed by data dispatchers. In this model, all data filters are invoked for all flows.

In order to ensure predictable behavior, the OPES architecture

requires the use of a standardized schema for the purpose of defining and interpreting the ruleset. The OPES architecture does not require a mechanism for configuring a ruleset into a data dispatcher. This is treated as a local matter for each implementation (e.g., through the use of a text editor, secure upload protocol, and so on). Future revisions of the architecture may introduce such a requirement.

2.4 Callout Servers

The evaluation of OPES ruleset determines which service applications will operate on a data stream. How the ruleset is evaluated is not the subject of the architecture, except to note that it must result in the same unambiguous result in all implementations.

In some cases it may be useful for the OPES processor to distribute the responsibility of service evaluation by communicating with one or more callout servers. A data dispatcher invokes the services of a callout server by using the OPES callout protocol (OCP). The requirements for the OCP are given in [7]. The OCP is application-agnostic, being unaware of the semantics of the encapsulated application protocol (e.g., HTTP). However, the OCP must incorporate a service aware vectoring capability that parses the data flow according to the ruleset and delivers the data to the OPES service application that can be local or remote.

In this model, OPES applications may be executed either on the same processor (or even in the same application environment) with the dispatcher or on a different OPES processor through OCP. The general interaction situation is depicted in Figure 3, which illustrates the positions and interaction of different components of OPES architecture.

```
   -----------------------------------------------------------------------



      +--------------------------+
      | +----------+             |
      | |   OPES   |             |
      | |  service |             |      +---------------+     +-----------+
      | |   appl   |             |      |Callout Server |     | Callout   |
      | +----------+             |      |     A         |     | Server    |
      |     ||                   |      | +--------+    |     |  X        |
      | +----------------------+ |      | | OPES   |    |     |           |
      | |     data dispatcher  | |      | | Service|    |     | +--------+|
      | +----------------------+ |      | |  App2  |    |     | | OPES   ||
      |                   ||     |      | +--------+    |     | |Service ||
      |               +-------+  |      |     ||        |     | | AppX   ||
      | +---------+   |       |  |      | +--------+    | ... | +--------||
      | |   HTTP  |   |  OCP  |=========| | OCP    |    |     |    ||     |
      | +---------|   +-------+  |      | +--------+    |     | +------+  |
      | +---------+      ||      |      +---------------+     | | OCP  |  |
      | |         |      =======================================|      |  |
      | |         |              |                            | +------+  |
      | | TCP/IP  |              |                            |           |
   =====|         |              |============== OPES ======  |           |
      | +---------+              |            Data Flow       +-----------+
      +--------------------------+



                  Figure 3: Interaction of OPES Entities

   -----------------------------------------------------------------------
```

2.5 Tracing Facility

The OPES architecture requires that each data dispatcher to provide tracing facilities that allow the appropriate verification of its operation. The OPES architecture requires that tracing be feasible on the OPES flow per OPES processor using in-band annotation. One of those annotations could be a URI with more detailed information on the transformation that occurred to the data on the OPES flow.

Providing the ability for in-band annotation MAY require header extensions on the application protocol that is used (e.g., HTTP). However, the presence of an OPES processor in the data request/

response flow SHALL NOT interfere with the operations of non-OPES aware clients and servers. The support of these extensions to the base protocol HTTP is not required by non-OPES clients and servers.

OPES processors must obey tracing, reporting and notification requirements set by the center of authority in the trust domain to which OPES processor belongs. As part of these requirements OPES processor may be instructed to reject or ignore such requirements that originate from other trust domains.

3. Security and Privacy Considerations

Each data flow must be secured in accordance with several policies. The primary stakeholders are the data consumer and the data provider. The secondary stakeholders are the entities to which they may have delegated their trust. The other stakeholders are the owners of the callout servers. Any of these parties may be participants in the OPES flow.

These parties must have a model, explicit or implicit, describing their trust policy; which of the other parties are trusted to operate on data, and what security enhancements are required for communication. The trust might be delegated for all data, or it might be restricted to granularity as small as an application data unit.

All parties that are involved in enforcing policies must communicate the policies to the parties that are involved. These parties are trusted to adhere to the communicated policies.

In order to delegate fine-grained trust, the parties must convey policy information by implicit contract, by a setup protocol, by a dynamic negotiation protocol, or in-line with application data headers.

3.1 Trust Domains

The delegation of authority starts at either a data consumer or data provider and moves to more distant entities in a "stepwise" fashion. Stepwise means A delegates to B and B delegates to C and so forth. The entities thus "colored" by the delegation are said to form a trust domain with respect to the original delegating party. Here, "Colored" means that if the first step in the chain is the data provider, then the stepwise delegation "colors" the chain with that data "provider" color. The only colors that are defined are the data "provider" and the data "consumer". Delegation of authority (coloring) propagates from the content producer start of authority or from the content consumer start of authority, that may be different from the end points in the data flow.

An OPES processor may be in several trust domains at any time. There is no restriction on whether the OPES processors are authorized by data consumers and/or data providers. The original party has the option of forbidding or limiting redelegation.

An OPES processor must have a representation of its trust domain memberships that it can report in whole or in part for tracing purposes. It must include the name of the party which delegated each

privilege to it.

3.2 Callout protocol

The determination of whether or not OPES processors will use the measures that are described in the previous section during their communication with callout servers depends on the details of how the primary parties delegated trust to the OPES processors and the trust relationship between the OPES processors and the callout server. If the OPES processors are in a single administrative domain with strong confidentiality guarantees, then encryption may be optional. However, it is recommended that for all cases that encryption and strong authentication be used.

If the delegation mechanism names the trusted parties and their privileges in some way that permits authentication, then the OPES processors will be responsible for enforcing the policy and for using authentication as part of that enforcement.

The callout servers must be aware of the policy governing the communication path. They must not, for example, communicate confidential information to auxiliary servers outside the trust domain.

A separate security association must be used for each channel established between an OPES processor and a callout server. The channels must be separate for different primary parties.

3.3 Privacy

Some data from data consumers is considered "private" or "sensitive", and OPES processors must both advise the primary parties of the their privacy policy and respect the policies of the primary parties. The privacy information must be conveyed on a per-flow basis.

The callout servers must also participate in handling of private data, and they must be prepared to announce their own capabilities and to enforce the policy required by the primary parties.

3.4 Establishing trust

The OPES processor will have configuration policy specifying what privileges the callout servers have and how they are to be identified. This is especially critical for third-party (fourth-party, etc.) callout servers. OPES uses standard protocols for authenticating and otherwise security communication with callout servers.

An OPES processor will have a trusted method for receiving configuration information such as rules for the data dispatcher, trusted callout servers, primary parties that opt-in or opt-out of individual services, etc.

3.5 End-to-end Integrity

Digital signature techniques can be used to mark data changes in such a way that a third-party can verify that the changes are or are not consistent with the originating party's policy. This requires an inline manner of specifying policy and its binding to data, a trace of changes and the party making the changes, and strong identities and authentication methods.

Strong end-to-end integrity can fulfill some of the functions required by "tracing".

4. Summary

Although the architecture supports a wide range of cooperative transformation services, it has few requirements for interoperability.

The necessary and sufficient elements are specified in the following documents:

- o the OPES ruleset schema [6] which defines the syntax and semantics of the rules interpreted by a data dispatcher; and,
- o the OPES callout protocol (OCP) [7] which defines the requirements for the protocol between a data dispatcher and a callout server.

References

[1] McHenry, S., et. al, "OPES Scenarios and Use Cases", Internet-Draft TBD, May 2002.

[2] Floyd, S. and L. Daigle, "IAB Architectural and Policy Considerations for Open Pluggable Edge Services", RFC 3238, January 2002.

[3] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J. and S. Waldbusser, "Terminology for Policy-Based Management", RFC 3198, November 2001.

[4] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[5] OPES working group, "OPES Service Authorization and Enforcement Requirements", Internet-Draft TBD, May 2002.

[6] OPES working group, "OPES Ruleset Schema", Internet-Draft TBD, May 2002.

[7] A. Beck et al., "Requirements for OPES Callout Protocols", Internet-Draft http://www.ietf.org/internet-drafts/draft-ietf-opes-protocol-reqs-00.txt, May 2002.

Authors' Addresses

Abbie Barbir
Nortel Networks
3500 Carling Avenue
Nepean, Ontario K2H 8E9
Canada

Phone: +1 613 763 5229
EMail: abbieb@nortelnetworks.com

Robin Chen
AT&T Labs
Room E219, 180 Park Avenue
Florham Park, NJ 07932
US

Phone: +1 973 360 8653
EMail: chen@research.att.com

Markus Hofmann
Bell Labs/Lucent Technologies
Room 4F-513
101 Crawfords Corner Road
Holmdel, NJ 07733
US

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

Hilarie Orman
Purple Streak Development

EMail: ho@alum.mit.edu

Reinaldo Penno
Nortel Networks
4555 Great America Parkway
Santa Clara, CA 95054
US

EMail: rpenno@nortelnetworks.com

Gary Tomlinson
The Tomlinson Group

EMail: gary@tomlinsongroup.net

Appendix A. Acknowledgements

The authors gratefully acknowledge the contributions of: Marshall T. Rose, John Morris, Oskar Batuner, Mark Baker and Ian Cooper.

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.