

Open Pluggable Edge Services
Internet-Draft
Expires: November 22, 2002

A. Beck
M. Hofmann
Lucent Technologies
H. Orman
Purple Streak Development
R. Penno
Nortel Networks
A. Terzis
Individual Consultant
May 24, 2002

Requirements for OPES Callout Protocols
draft-ietf-opes-protocol-reqs-00

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/lid-abstracts.txt>.

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

This Internet-Draft will expire on November 22, 2002.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document specifies the requirements that the OPES (Open Pluggable Edge Services) callout protocol must satisfy in order to support the remote execution of OPES services [1]. The requirements are intended to help evaluating possible protocol candidates and to guide the development of such protocols.

Internet-Draft

Requirements for OPES Callout Protocols

May 2002

Table of Contents

1.	Terminology	3
2.	Introduction	4
3.	Functional Requirements	5
3.1	Callout Transactions	5
3.2	Callout Channels	5
3.3	Reliability	6
3.4	Congestion and Flow Control	6
3.5	Support for Keep-Alive Mechanism	6
3.6	Operation in NAT Environments	7
3.7	Multiple Callout Servers	7
3.8	Multiple Data Processors	7
3.9	Support for Different Application Protocols	7
3.10	Capability and Parameter Negotiations	7
3.11	Meta Data and Instructions	8
3.12	Asynchronous Message Exchange	9
3.13	Message Segmentation	9
4.	Performance Requirements	11
4.1	Protocol Efficiency	11
5.	Security Requirements	12
5.1	Authentication, Confidentiality, and Integrity	12
5.2	Hop-by-Hop Confidentiality	12
5.3	Operation Across Un-trusted Domains	12
5.4	Privacy	13
6.	Security Considerations	14
7.	Acknowledgments	15
	References	16
	Authors' Addresses	16
	Full Copyright Statement	18

1. Terminology

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](#) [2].

[2](#). Introduction

The Open Pluggable Edge Services (OPES) architecture [[1](#)] enables cooperative application services (OPES services) between a data provider, a data consumer, and zero or more data processors. The application services under consideration analyze and possibly transform application-level messages exchanged between the data provider and the data consumer.

The execution of such services is governed by a set of filtering rules installed on the data processor. The rules enforcement can trigger the execution of service applications local to the data processor. Alternatively, the data processor can distribute the responsibility of service execution by communicating and collaborating with one or more remote callout servers. As described in [[1](#)], a data processor communicates with and invokes services on a callout server by using a callout protocol. This document presents the requirements for such a protocol.

The requirements in this document are divided into three categories - functional requirements, performance requirements, and security requirements. Each requirement is presented as one or more statements, followed by brief explanatory material as appropriate.

[3. Functional Requirements](#)

[3.1 Callout Transactions](#)

The OPES callout protocol MUST enable an OPES data processor and a callout server to perform callout transactions with the purpose of exchanging partial or complete application-level protocol messages (or modifications thereof). More specifically, the callout protocol MUST enable an OPES data processor to forward a complete or partial application message to a callout server so that one or more OPES services can process the forwarded application message (or parts thereof). The result of the service operation may be a modified application message. The callout protocol MUST therefore enable the callout server to return a modified application message or the modified parts of an application message to the OPES data processor.

A callout transaction is defined as a message exchange between an OPES data processor and a callout server consisting of a callout request and a callout response. Both, the callout request as well as the callout response, MAY each consist of one or more protocol messages, i.e. a series of protocol messages.

Callout transactions are always initiated by a callout request from an OPES data processor and typically terminated by a callout response from a callout server. The OPES callout protocol **MUST**, however, also allow either endpoint of a callout transaction to terminate a callout transaction prematurely, i.e. before a callout request or response has been completely received by the corresponding endpoint. The callout protocol **MAY** provide an explicit (e.g. through a termination message) or implicit (e.g. through a connection tear-down) mechanism to terminate a callout transaction prematurely. Such a mechanism **MUST** ensure, however, that a premature termination of a callout transaction does not result in the loss of application message data.

A premature termination of a callout transaction is required to support OPES services which may terminate even before they have processed the entire application message. Content analysis services, for example, may be able to classify a Web object after having processed just the first few bytes of a Web object.

The callout protocol **MUST** further enable a callout server to report back to the OPES data processor the result of a callout transaction, e.g. in the form of a status code.

[3.2](#) Callout Channels

The OPES callout protocol **MUST** enable an OPES data processor and a callout server to perform multiple callout transactions over a

callout channel. A callout channel is defined as a logical connection at the application-layer between an OPES data processor and a callout server.

Callout channels **MUST** always be established by an OPES data processor. A callout channel **MAY** be closed by either endpoint of the callout channel provided that all callout transactions associated with the channel have terminated.

A callout channel **MAY** have certain parameters associated with it, for example parameters that control the fail-over behavior of channel endpoints. Callout channel parameters **MAY** be negotiated between OPES data processors and callout servers (see [Section 3.10](#)).

[3.3 Reliability](#)

The OPES callout protocol MUST be able to provide ordered reliability for the communication between OPES data processor and callout server. Additionally, the callout protocol SHOULD be able to provide unordered reliability.

In order to satisfy the reliability requirements, the callout protocol MAY specify that it must be used with a lower-level transport protocol which provides ordered reliability at the transport-layer.

[3.4 Congestion and Flow Control](#)

The OPES callout protocol MUST ensure that congestion and flow control mechanisms are applied on all callout transactions. For this purpose, the callout protocol MAY specify callout protocol-specific mechanisms or refer to a lower-level transport protocol and discuss how its mechanisms provide for congestion and flow control.

[3.5 Support for Keep-Alive Mechanism](#)

The OPES callout protocol MUST provide an optional keep-alive mechanism which, if used, would allow both endpoints of a callout channel to detect a failure of the other endpoint even in the absence of callout transactions. The callout protocol MAY specify that keep-alive messages be exchanged over existing callout channel connections or a separate connection between OPES data processor and callout server.

The detection of a callout server failure may enable an OPES data processor to establish a channel connection with a stand-by callout server so that future callout transactions do not result in the loss of application message data. The detection of the failure of an OPES

data processor may enable a callout server to release resources which would otherwise not be available for callout transactions with other OPES data processors.

[3.6 Operation in NAT Environments](#)

The OPES protocol SHOULD be NAT-friendly, i.e. its operation should

not be compromised by the presence of one or more NAT devices in the path between an OPES data processor and a callout server.

[3.7](#) Multiple Callout Servers

The OPES callout protocol MUST allow an OPES data processor to simultaneously communicate with more than one callout server.

In larger networks, OPES services are likely to be hosted by different callout servers. Therefore, an OPES data processor will likely have to communicate with multiple callout servers. The protocol design MUST enable an OPES data processor to do so.

[3.8](#) Multiple Data Processors

The OPES callout protocol MUST allow a callout server to simultaneously communicate with more than one OPES data processor.

The protocol design MUST support a scenario in which multiple OPES data processors use the services of a single callout server.

[3.9](#) Support for Different Application Protocols

The OPES callout protocol MUST be application protocol-agnostic, i.e. it MUST not make any assumptions about the characteristics of the application-layer protocol used on the data path between data provider and data consumer.

The OPES entities on the data path may use different application-layer protocols, including, but not limited to, HTTP [\[3\]](#) and RTP [\[4\]](#). It would be desirable to be able to use the same OPES callout protocol for any such application-layer protocol.

[3.10](#) Capability and Parameter Negotiations

The OPES callout protocol MUST support the negotiation of capabilities and callout channel parameters between an OPES data processor and a callout server. This implies that the OPES data processor and the callout server MUST be able to exchange their capabilities and preferences and engage into a deterministic negotiation process at the end of which the two endpoints have either

agreed on the capabilities and parameters to be used for future callout channel transactions or determined that their capabilities are incompatible.

Capabilities and parameters that could be negotiated between an OPES data processor and a callout server include (but are not limited to): callout protocol version, transport-layer protocol, fail-over behavior, heartbeat rate for keep-alive messages, security-related parameters etc.

Channel parameters may also pertain to the characteristics of OPES callout services if, for example, callout channels are associated with one or more specific OPES services. An OPES service-specific parameter may, for example, specify which parts of an application message an OPES service requires for its operation.

The parties to a callout protocol MAY use callout channels to negotiate all or some of their capabilities and parameters. They MAY also use a separate control connection for this purpose. If there is a need for callout channel parameters, then they MUST be negotiated on a per-callout channel basis and before any callout transactions are performed over the corresponding channel. Other parameters and capabilities, such as the fail-over behavior, MAY be negotiated between the two endpoints independently of callout channels.

[3.11](#) Meta Data and Instructions

The OPES callout protocol MUST provide a mechanism for the endpoints of a particular callout transaction to include in callout requests and responses meta data and instructions for the OPES data processor or callout server.

Specifically, the callout protocol MUST enable an OPES data processor to include information about the forwarded application message in a callout request, e.g. in order to specify the type of the forwarded application message or to specify what part(s) of the application message are forwarded to the callout server. Likewise, the callout server MUST be able to include information about the returned application message.

The OPES data processor MUST further be able to uniquely specify one or more OPES services which are to be performed on the forwarded application message. The callout protocol MAY also choose to associate callout channels with specific OPES services so that there is no need to identify OPES service on a per-callout transaction basis.

Additionally, the OPES callout protocol MUST allow the callout server

to indicate to the OPES data processor the cacheability of callout responses. This implies that callout responses may have to carry cache-control instructions for the OPES data processor.

The OPES callout protocol MUST further enable the OPES data processor to indicate to the callout server if it has kept a local copy of the forwarded application message (or parts thereof). This information enables the callout server to determine whether the forwarded application message must be returned to the OPES data processor even it has not been modified by an OPES service.

The OPES callout protocol MUST also allow OPES data processors to comply with the tracing requirements of the OPES architecture as laid out in [1] and [5]. This implies that the callout protocol MUST enable a callout server to convey to the OPES data processor information about the OPES service operations performed on the forwarded application message.

[3.12](#) Asynchronous Message Exchange

The OPES callout protocol MUST support an asynchronous message exchange between an OPES data processor and a callout server.

In order to allow asynchronous processing on the OPES data processor and callout server, it MUST be possible to separate request issuance from response processing. The protocol MUST therefore allow multiple outstanding requests and provide a method to correlate responses to requests.

Additionally, the callout protocol MUST enable a callout server to respond to a callout request before it has received the entire request.

[3.13](#) Message Segmentation

The OPES callout protocol MUST allow an OPES data processor to forward an application message to a callout server in a series of smaller message fragments. The callout protocol MUST further enable the receiving callout server to assemble the fragmented application message.

Likewise, the callout protocol MUST enable a callout server to return an application message to a data processor in a series of smaller

message fragments. The callout protocol MUST enable the receiving data processor to assemble the fragmented application message.

Depending on the application-layer protocol used on the data path, application messages may be very large in size (for example in the

case of audio/video streams) or of unknown size. In both cases, the OPES data processor has to initiate a callout transaction before it has received the entire application message to avoid long delays for the data consumer. The OPES data processor MUST therefore be able to forward fragments or chunks of an application message to a callout server as it receives them from the data provider or consumer. Likewise, the callout server MUST be able to process and return application message fragments as it receives them from the data processor.

[4.](#) Performance Requirements

[4.1](#) Protocol Efficiency

The OPES callout protocol SHOULD be efficient in that it minimizes the additionally introduced latency, for example by minimizing the protocol overhead. At a minimum, the callout protocol SHOULD satisfy the performance requirements of the application-layer protocol whose messages are forwarded from the OPES data processor to the callout server.

As OPES callout transactions introduce additional latency to application protocol transactions on the data path, callout protocol efficiency is crucial.

[5. Security Requirements](#)

In the absence of any security mechanisms, sensitive information might be communicated between the OPES data processor and the callout server in violation of either endpoint's security and privacy policy through misconfiguration or a deliberate insider attack. By using strong authentication, message encryption, and integrity checks, this threat can be minimized to a smaller set of insiders and/or operator configuration errors.

The OPES data processor and the callout servers SHOULD have enforceable policies that limit the parties they communicate with, that determine the protections to use based on identities of the endpoints and other data (such as enduser policies). In order to enforce the policies, they MUST be able to authenticate the callout protocol endpoints using cryptographic methods.

[5.1 Authentication, Confidentiality, and Integrity](#)

The parties to the callout protocol MUST have a sound basis for binding authenticated identities to the protocol endpoints, and they MUST verify that these identities are consistent with their security

policies.

The OPES callout protocol MUST provide message authentication, confidentiality, and integrity between the OPES data processor and the callout server. It MUST provide mutual authentication. The callout protocol SHOULD use existing security mechanisms for this purpose. The callout protocol specification is not required to specify the security mechanisms, but it MAY instead refer to a lower-level security protocol and discuss how its mechanisms are to be used with the callout protocol.

[5.2](#) Hop-by-Hop Confidentiality

If encryption is a requirement for the content path, then this confidentiality MUST be extended to the communication between the callout servers and the OPES data processor. In order to minimize data exposure, the callout protocol MUST use a different encryption key for each encrypted content stream.

[5.3](#) Operation Across Un-trusted Domains

The OPES callout protocol MUST operate securely across un-trusted domains between the OPES data processor and the callout server.

If the communication channels between the OPES data processor and callout server cross outside of the organization taking

responsibility for the OPES services, then endpoint authentication and message protection (confidentiality and integrity) MUST be used.

[5.4](#) Privacy

Any communication carrying information relevant to privacy policies MUST protect the data using encryption.

[6](#). Security Considerations

The security requirements for the OPES callout protocol are discussed in [Section 5](#).

This document is based in parts on previous work by Anca Dracinschi Sailer, Volker Hilt, and Rama R. Menon.

References

- [1] Barbir, A., "An Architecture for Open Pluggable Edge Services (OPES)", [draft-ietf-opes-architecture-00](#) (work in progress), May 2002.
- [2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997.
- [3] 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.
- [4] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", [RFC 1889](#), January 1996.
- [5] Floyd, S. and L. Daigle, "IAB Architectural and Policy Considerations for Open Pluggable Edge Services", [RFC 3238](#), January 2002.

Authors' Addresses

Andre Beck
Lucent Technologies
101 Crawfords Corner Road
Holmdel, NJ 07733
US

E-Mail: abeck@bell-labs.com

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

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

Internet-Draft

Requirements for OPES Callout Protocols

May 2002

Hilarie Orman
Purple Streak Development

E-Mail: ho@alum.mit.edu

URI: <http://www.purplestreak.com>

Reinaldo Penno
Nortel Networks
2305 Mission College Boulevard
San Jose, CA 95134
US

E-Mail: rpenno@nortelnetworks.com

Andreas Terzis
Individual Consultant
150 Golf Course Dr.
Rohnert Park, CA 94928
US

Phone: +1 707 586 8864

E-Mail: aterzis@sbcglobal.net

Full Copyright Statement

Copyright (C) The Internet Society (2002). 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.

Acknowledgement

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

