Network Working Group
Request for Comments: 3836
Category: Informational

A. Beck
M. Hofmann
Lucent Technologies
H. Orman
Purple Streak Development
R. Penno
Nortel Networks
A. Terzis
Johns Hopkins University

August 2004

Requirements for Open Pluggable Edge Services (OPES) Callout Protocols

Status of this Memo

Copyright Notice

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. The requirements are intended to help evaluate possible protocol candidates, as well as to guide the development of such protocols.

Table of Contents

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 BCP 14, 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 OPES 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 rules installed on the OPES processor. The enforcement of rules can trigger the execution of service applications local to the OPES processor. Alternatively, the OPES processor can distribute the responsibility of service execution by communicating and collaborating with one or more remote callout servers. As described in [1], an OPES 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. Reliability

The OPES callout protocol MUST be able to provide ordered reliability for the communication between an OPES 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 SHOULD specify that it must be used with a transport protocol that provides ordered/unordered reliability at the transport-layer, for example TCP [6] or SCTP [7].

3.2. Congestion Avoidance

The OPES callout protocol MUST ensure that congestion avoidance matching the standard of RFC 2914 [4] is applied on all communication between the OPES processor and callout server. For this purpose, the callout protocol SHOULD use a congestion-controlled transport-layer protocol, presumably either TCP [6] or SCTP [7].

3.3. Callout Transactions

The OPES callout protocol MUST enable an OPES 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 processor to forward a partial or complete 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 processor. Additionally, the callout protocol MUST enable a callout server to report the result of a callout transaction (e.g., in the form of a status code) back to the OPES processor.

A callout transaction is defined as a message exchange between an OPES processor and a callout server consisting of a callout request and a callout response. Both, the callout request and the callout response MAY each consist of one or more callout protocol messages, i.e. a series of protocol messages. A callout request MUST always contain a partial or complete application message. A callout response MUST always indicate the result of the callout transaction. A callout response MAY contain a modified application message.

Callout transactions are always initiated by a callout request from an OPES processor and are typically terminated by a callout response from a callout server. The OPES callout protocol MUST, however, also provide a mechanism that allows either endpoint of a callout transaction to terminate a callout transaction before a callout request or response has been completely received by the corresponding callout endpoint. Such a mechanism MUST ensure 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.

3.4. Callout Connections

The OPES callout protocol MUST enable an OPES processor and a callout server to perform multiple callout transactions over a callout connection. Additionally, the callout protocol MUST provide a method of associating callout transactions with callout connections. A callout connection is defined as a logical connection at the application-layer between an OPES processor and a callout server. A callout connection MAY have certain parameters associated with it, for example parameters that control the fail-over behavior of connection endpoints. Callout connection-specific parameters MAY be negotiated between OPES processors and callout servers (see Section 3.12).

The OPES callout protocol MAY choose to multiplex multiple callout connections over a single transport-layer connection if a flow control mechanism that guarantees fairness among multiplexed callout connections is applied.

Callout connections MUST always be initiated by an OPES processor. A callout connection MAY be closed by either endpoint of the connection, provided that doing so does not affect the normal operation of on-going callout transactions associated with the callout connection.

3.5. Asynchronous Message Exchange

The OPES callout protocol MUST support an asynchronous message exchange over callout connections.

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

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

3.6. Message Segmentation

The OPES callout protocol MUST allow an OPES 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 re-assemble the fragmented application message.

Likewise, the callout protocol MUST enable a callout server to return an application message to an OPES processor in a series of smaller message fragments. The callout protocol MUST enable the receiving OPES processor to re-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 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 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 OPES processor.

Application message segmentation is also required if the OPES callout protocol provides a flow control mechanism in order to multiplex multiple callout connections over a single transport-layer connection (see Section 3.4).

3.7. Support for Keep-Alive Mechanism

The OPES callout protocol MUST provide a keep-alive mechanism which, if used, would allow both endpoints of a callout connection 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 connections or a separate connection between OPES processor and callout server. The callout protocol MAY also specify that the use of the keep-alive mechanism is optional.

The detection of a callout server failure may enable an OPES processor to establish a callout 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 processor may enable a callout server to release resources which would otherwise not be available for callout transactions with other OPES processors.

3.8. 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 processor and a callout server.

3.9. Multiple Callout Servers

The OPES callout protocol MUST allow an OPES 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 processor will likely have to communicate with multiple callout servers. The protocol design MUST enable an OPES processor to do so.

3.10. Multiple OPES Processors

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

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

3.11. Support for Different Application Protocols

The OPES callout protocol SHOULD be application protocol-agnostic, i.e., it SHOULD not make any assumptions about the characteristics of the application-layer protocol used on the data path between the data provider and data consumer. At a minimum, the callout protocol MUST be compatible with HTTP [5].

The OPES entities on the data path may use different application-layer protocols, including, but not limited to, HTTP [5] and RTP [8]. It would be desirable to be able to use the same OPES callout protocol for any such application-layer protocol.

3.12. Capability and Parameter Negotiations

The OPES callout protocol MUST support the negotiation of capabilities and callout connection parameters between an OPES processor and a callout server. This implies that the OPES processor and the callout server MUST be able to exchange their capabilities and preferences. Then they MUST be able to engage in a deterministic negotiation process that terminates either with the two endpoints agreeing on the capabilities and parameters to be used for future callout connections/transactions or with a determination that their capabilities are incompatible.

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

The callout protocol MUST NOT use negotiation to determine the transport protocol to be used for callout connections. The callout protocol MAY, however, specify that a certain application message protocol (e.g., HTTP [5], RTP [8]) requires the use of a certain transport protocol (e.g., TCP [6], SCTP [7]).

Callout connection parameters may also pertain to the characteristics of OPES callout services if, for example, callout connections 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.

Callout connection parameters MUST be negotiated on a per-callout connection basis and before any callout transactions are performed over the corresponding callout connection. Other parameters and

capabilities, such as the fail-over behavior, MAY be negotiated between the two endpoints independently of callout connections.

The parties to a callout protocol MAY use callout connections to negotiate all or some of their capabilities and parameters. Alternatively, a separate control connection MAY be used for this purpose.

3.13. Meta Data and Instructions

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

Specifically, the callout protocol MUST enable an OPES processor to include information about the forwarded application message in a callout request, e.g. in order to specify the type of 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 processor MUST further be able to include an ordered list of one or more uniquely specified OPES services which are to be performed on the forwarded application message in the specified order. However, as the callout protocol MAY also choose to associate callout connections with specific OPES services, there may not be a need to identify OPES services on a per-callout transaction basis.

Additionally, the OPES callout protocol MUST allow the callout server to indicate to the OPES processor the cacheability of callout responses. This implies that callout responses may have to carry cache-control instructions for the OPES processor.

The OPES callout protocol MUST further enable the OPES 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 processor, even if it has not been modified by an OPES service.

The OPES callout protocol MUST also allow OPES processors to comply with the tracing requirements of the OPES architecture as laid out in [1] and [3]. This implies that the callout protocol MUST enable a callout server to convey to the OPES processor information about the OPES service operations performed on the forwarded application message.

4. Performance Requirements

4.1. Protocol Efficiency

The OPES callout protocol SHOULD have minimal latency. For example, the size and complexity of its headers could be minimized.

Because OPES callout transactions add latency to application protocol transactions on the data path, callout protocol efficiency is crucial to overall performance.

5. Security Requirements

In the absence of any security mechanisms, sensitive information might be communicated between the OPES processor and the callout server in violation of either endpoint's security and privacy policy, through misconfiguration or 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 processor and the callout servers SHOULD have enforceable policies that limit the parties they communicate with and 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 for message authentication, confidentiality, and integrity between the OPES processor and the callout server. It MUST provide mutual authentication. For this purpose, the callout protocol SHOULD use existing security mechanisms. 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 hop-by-hop encryption is a requirement for the content path, then this confidentiality MUST be extended to the communication between the OPES processor and the callout server. While it is recommended that the communication between the OPES processor and callout server always be encrypted, encryption MAY be optional if both the OPES processor and the callout server are co-located together in a single administrative domain with strong confidentiality guarantees.

In order to minimize data exposure, the callout protocol MUST use a different encryption key for each encrypted content stream.

5.3. Operation Across Untrusted Domains

The OPES callout protocol MUST operate securely across untrusted domains between the OPES processor and the callout server.

If the communication channels between the OPES processor and callout server cross outside of the organization which is responsible 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.

7. References

7.1. Normative References

[1] Barbir, A., Penno, R., Chen, R., Hofmann, M., and H. Orman, "An Architecture for Open Pluggable Edge Services (OPES)", RFC 3835, August 2004.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

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

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

7.2. Informative References

[6] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[7] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[8] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 3550, July 2003.

8. Acknowledgments

Parts of this document are based on previous work by Anca Dracinschi Sailer, Volker Hilt, and Rama R. Menon.

The authors would like to thank the participants of the OPES WG for their comments on this document.

9. Authors' Addresses

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

EMail: abeck@bell-labs.com

Markus Hofmann
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
URI: http://www.purplestreak.com

Reinaldo Penno
Nortel Networks
600 Technology Park Drive
Billerica, MA 01821
US

EMail: rpenno@nortelnetworks.com

Andreas Terzis
Computer Science Department
Johns Hopkins University
3400 North Charles Street, 224 NEB
Baltimore, MD 21218

Phone: +1 410 516 5847
EMail: terzis@cs.jhu.edu

10. Full Copyright Statement

Intellectual Property

Acknowledgement

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