

Open Pluggable Edge Services (OPES) Entities and End Points Communication

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2004).

Abstract

This memo documents tracing and non-blocking (bypass) requirements for Open Pluggable Edge Services (OPES).

Table of Contents

1.	Introduction	2
1.1.	Terminology	2
2.	OPES System	2
3.	Tracing Requirements	3
3.1.	Traceable entities	3
3.2.	System requirements	5
3.3.	Processor requirements	5
3.4.	Callout server requirements	5
4.	Bypass (Non-blocking feature) Requirements	6
4.1.	Bypassable entities	7
4.2.	System requirements	8
4.3.	Processor requirements	8
4.4.	Callout server requirements	9
5.	Protocol Binding	9
6.	Compliance Considerations	9
7.	IANA Considerations	9
8.	Security Considerations	10
8.1.	Tracing security considerations	10
8.2.	Bypass security considerations	11
9.	References	12
9.1.	Normative References	12
9.2.	Informative References	13
10.	Acknowledgements	13

11. Author's Address	13
12. Full Copyright Statement	14

1. 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.

This work specifies OPES tracing and bypass functionality. The architecture document [1] requires that tracing is supported in-band. This design goal limits the type of application protocols that OPES can support. The details of what a trace record can convey are also dependent on the choice of the application level protocol. For these reasons, this work only documents requirements for OPES entities that are needed to support traces and bypass functionality. The task of encoding tracing and bypass features is application protocol specific. Separate documents will address HTTP and other protocols.

The architecture does not prevent implementers from developing out-of-band protocols and techniques to address tracing and bypass. Such protocols are out of scope of the current work.

1.1. Terminology

The keywords "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]. When used with the normative meanings, these keywords will be all uppercase. Occurrences of these words in lowercase comprise normal prose usage, with no normative implications.

2. OPES System

This section provides a definition of OPES System. This is needed in order to define what is traceable (or bypassable) in an OPES Flow.

Definition: An OPES System is a set of all OPES entities authorized by either the data provider or the data consumer application to process a given application message.

The nature of the authorization agreement determines if authority delegation is transitive (meaning an authorized entity is authorized to include other entities).

If specific authority agreements allow for re-delegation, an OPES system can be formed by induction. In this case, an OPES system starts with entities directly authorized by a data provider (or a data consumer) application. The OPES system then includes any OPES entity authorized by an entity that is already in the OPES system. The authority delegation is always viewed in the context of a given application message.

An OPES System is defined on an application message basis. Having an authority to process a message does not imply being involved in message processing. Thus, some OPES system members may not participate in processing of a message. Similarly, some members may process the same message several times.

The above definition implies that there can be no more than two OPES systems [Client-side and server-side OPES systems can process the same message at the same time] processing the same message at a given time. This is based on the assumption that there is a single data provider and a single data consumer as far as a given application message is concerned.

For example, consider a Content Delivery Network (CDN) delivering an image on behalf of a busy web site. OPES processors and services, which the CDN uses to adapt and deliver the image, comprise an OPES System. In a more complex example, an OPES System would contain third party OPES entities that the CDN engages to perform adaptations (e.g., to adjust image quality).

3. Tracing Requirements

The definition of OPES trace and tracing are given next.

OPES trace: application message information about OPES entities that adapted the message.

OPES tracing: the process of creating, manipulating, or interpreting an OPES trace.

Note that the above trace definition assumes in-band tracing. This dependency can be removed if desired. Tracing is performed on per message basis. Trace format is dependent on the application protocol that is being adapted. A traceable entity can appear multiple times in a trace (for example, every time it acts on a message).

3.1. Traceable entities

This section focuses on identifying traceable entities in an OPES Flow.

Tracing information provides an "end" with information about OPES entities that adapted the data. There are two distinct uses of OPES traces. First, a trace enables an "end" to detect the presence of OPES System. Such "end" should be able to see a trace entry, but does not need to be able to interpret it beyond identification of the OPES System and location of certain required OPES-related disclosures (see Section 3.2).

Second, the OPES System administrator is expected to be able to interpret the contents of an OPES trace. The trace can be relayed to the administrator by an "end" without interpretation, as opaque data (e.g., a TCP packet or an HTTP message snapshot). The administrator can use the trace information to identify the participating OPES entities. The administrator can use the trace to identify the applied adaptation services along with other message-specific information.

Since the administrators of various OPES Systems can have various ways of looking into tracing, they require the freedom in what to put in trace records and how to format them.

At the implementation level, for a given trace, an OPES entity involved in handling the corresponding application message is traceable or traced if information about it appears in that trace. This work does not specify any order to that information. The order of information in a trace can be OPES System specific or can be defined by application bindings documents.

OPES entities have different levels of traceability requirements. Specifically,

- o An OPES System MUST add its entry to the trace.
- o An OPES processor SHOULD add its entry to the trace.
- o An OPES service MAY add its entry to the trace.
- o An OPES entity MAY delegate addition of its trace entry to another OPES entity. For example, an OPES System can have a dedicated OPES processor for adding System entries; an OPES processor can use a callout service to manage all OPES trace manipulations (since such manipulations are OPES adaptations).

In an OPES context, a good tracing approach is similar to a trouble ticket ready for submission to a known address. The address is printed on the ticket. The trace in itself is not necessarily a detailed description of what has happened. It is the responsibility of the operator to decode trace details and to resolve the problems.

3.2 System requirements

The following requirements document actions when forming an OPES System trace entry:

- o OPES system MUST include its unique identification in its trace entry. Here, uniqueness scope is all OPES Systems that may adapt the message being traced.
- o An OPES System MUST define its impact on inter- and intra-document reference validity.
- o An OPES System MUST include information about its privacy policy, including identity of the party responsible for setting and enforcing the policy.
- o An OPES System SHOULD include information that identifies, to the technical contact, the OPES processors involved in processing the message.
- o When providing required information, an OPES System MAY use a single URI to identify a resource containing several required items. For example, an OPES System can point to a single web page with a reference to System privacy policy and technical contact information.

This specification does not define the meaning of the terms privacy policy, policy enforcement, or reference validity or technical contact and contains no requirements regarding encoding, language, format, or any other aspects of that information. For example, a URI used for an OPES System trace entry may look like "http://www.examplecompany.com/opes/?client=example.com" where the identified web page is dynamically generated and contains the all OPES System information required above.

3.3. Processor requirements

The following requirements document actions when forming an OPES System trace entry:

- o OPES processor SHOULD add its unique identification to the trace. Here, uniqueness scope is the OPES System containing the processor.

3.4. Callout server requirements

In an OPES system, it is the task of an OPES processor to add trace records to application messages. The OPES System administrator decides if and under what conditions callout servers may add trace information to application messages.

4. Bypass (Non-blocking feature) Requirements

IAB recommendation (3.3) [6] requires that the OPES architecture does not prevent a data consumer application from retrieving non-OPES version of content from a data provider application, provided that the non-OPES content exists. IAB recommendation (3.3) suggests that the Non-blocking feature (bypass) be used to bypass faulty OPES intermediaries (once they have been identified, by some method).

In addressing IAB consideration (3.3), one need to specify what constitutes non-OPES content. In this work the definition of "non-OPES" content is provider dependent. In some cases, the availability of "non-OPES" content can be a function of the internal policy of a given organization that has contracted the services of an OPES provider. For example, Company A has as contract with an OPES provider to perform virus checking on all e-mail attachments. An employee X of Company A can issue a non-blocking request for the virus scanning service. The request could be ignored by the OPES provider since it contradicts its agreement with Company A.

The availability of non-OPES content can be a function of content providers (or consumers or both) policy and deployment scenarios [5]. For this reason, this work does not attempt to define what is an OPES content as opposed to non-OPES content. The meaning of OPES versus non-OPES content is assumed to be determined through various agreements between the OPES provider, data provider and/or data consumer. The agreement determines what OPES services can be bypassed and in what order (if applicable).

This specification documents bypassing of an OPES service or a group of services identified by a URI. In this context, to "bypass the service" for a given application message in an OPES Flow means to "not invoke the service" for that application message. A bypass URI that identifies an OPES system (processor) matches all services attached to that OPES system (processor). However, bypassing of OPES processors and OPES Systems themselves requires non-OPES mechanisms and is out of this specification scope. A bypass request an instruction to bypass, usually embedded in an application message.

The current specification does not provide for a good mechanism that allow and "end" to specify to "bypass this service but only if it is a part of that OPES system" or "bypass all services of that OPES system but not of this OPES system". Furthermore, if an OPES processor does not know for sure that a bypass URI does not match its service, it must bypass that service.

If no non-OPES content is available without the specified service, the bypass request for that service must be ignored. This design implies that it may not be possible to detect non-OPES content existence or to detect violations of bypass rules in the environments where the tester does not know whether non-OPES content exists. This design assumes that most bypass requests are intended for situations where serving undesirable OPES content is better than serving an error message that no preferred non-OPES content exists.

Bypass feature is to malfunctioning OPES services as HTTP "reload" request is to malfunctioning HTTP caches. The primary purpose of the bypass is to get usable content in the presence of service failures and not to provide the content consumer with more information on what is going on. OPES trace should be used for the latter instead.

While this work defines a "bypass service if possible" feature, there are other related bypass features that can be implemented in OPES and/or in application protocols being adapted. For example, a "bypass service or generate an error" or "bypass OPES entity or generate an error". Such services would be useful for debugging broken OPES systems and may be defined in other OPES specifications. This work concentrates on documenting a user-level bypass feature addressing direct IAB concerns.

4.1. Bypassable entities

In this work, the focus is on developing a bypass feature that allows a user to instruct the OPES System to bypass some or all of its services. The collection of OPES services that can be bypassed is a function of the agreement of the OPES provider with either (or both) the content provider or the content consumer applications. In the general case, a bypass request is viewed as a bypass instruction that contains a URI that identifies an OPES entity or a group of OPES entities that perform a service (or services) to be bypassed. An instruction may contain more than one such URI. A special wildcard identifier can be used to represent all possible URIs.

In an OPES Flow, a bypass request is processed by each involved OPES processor. This means that an OPES processor examines the bypass instruction and if non-OPES content is available, the processor then bypasses the indicated services. The request is then forwarded to the next OPES processor in the OPES Flow. The next OPES processor would then handle all bypass requests, regardless of the previous processor actions. The processing chain continues throughout the whole processors that are involved in the OPES Flow.

4.2. System requirements

In an OPES System, bypass requests are generally client centric (originated by the data consumer application) and go in the opposite direction of tracing requests. This work requires that the bypass feature be performed in-band as an extension to an application specific protocol. Non-OPES entities should be able to safely ignore these extensions. The work does not prevent OPES Systems from developing their own out of band protocols.

The following requirements apply for bypass feature as related to an OPES System (the availability of a non-OPES content is a precondition):

- o An OPES System MUST support a bypass feature. This means that the OPES System bypasses services whose URIs are identified by an OPES "end".
- o An OPES System MUST provide OPES version of the content if non-OPES version is not available.

In order to facilitate the debugging (or data consumer user experience) of the bypass feature in an OPES System, it would be beneficial if non-bypassed entities included information related to why they ignored the bypass instruction. It is important to note that in some cases the tracing facility itself may be broken and the whole OPES System (or part) may need to be bypassed through the issue of a bypass instruction.

4.3. Processor requirements

Bypass requirements for OPES processors are (the availability of a non-OPES content is a precondition):

- o OPES processor SHOULD be able to interpret and process a bypass instruction. This requirement applies to all bypass instructions, including those that identify unknown-to-recipient services.
- o OPES processors MUST forward bypass request to the next application hop provided that the next hop speaks application protocol with OPES bypass support.
- o OPES processor SHOULD be able to bypass it's service(s) execution.

OPES processors that know how to process and interpret a bypass instruction have the following requirements:

- o The recipient of a bypass instruction with a URI that does not identify any known-to-recipient OPES entity MUST treat that URI as a wildcard identifier (meaning bypass all applicable services).

4.4. Callout server requirements

In an OPES system, it is the task of an OPES processor to process bypass requests. The OPES System administrator decides if and under what conditions callout servers process bypass requests.

5. Protocol Binding

The task of encoding tracing and bypass features is application protocol specific. Separate documents will address HTTP and other protocols. These documents must address the ordering of trace information as needed.

6. Compliance Considerations

This specification defines compliance for the following compliance subjects: OPES System, processors, entities and callout servers.

A compliance subject is compliant if it satisfies all applicable "MUST" and "SHOULD" level requirements. By definition, to satisfy a "MUST" level requirement means to act as prescribed by the requirement; to satisfy a "SHOULD" level requirement means to either act as prescribed by the requirement or have a reason to act differently. A requirement is applicable to the subject if it instructs (addresses) the subject.

Informally, compliance with this document means that there are no known "MUST" violations, and all "SHOULD" violations are conscious. In other words, a "SHOULD" means "MUST satisfy or MUST have a reason to violate". It is expected that compliance claims are accompanied by a list of unsupported SHOULDs (if any), in an appropriate format, explaining why preferred behavior was not chosen.

Only normative parts of this specification affect compliance. Normative parts are: parts explicitly marked using the word "normative", definitions, and phrases containing unquoted capitalized keywords from RFC 2119 [2]. Consequently, examples and illustrations are not normative.

7. IANA Considerations

This specification contains no IANA considerations. Application bindings MAY contain application-specific IANA considerations.

8. Security Considerations

Security considerations for OPES are documented in [4]. Policy and authorization issues are documented in [3]. It is recommended that designers consult these documents before reading this section.

This document is a requirement document for tracing and bypass feature. The requirements that are stated in this document can be used to extend an application level protocol to support these features. As such, the work has security precautions.

8.1. Tracing security considerations

The tracing facility for OPES architecture is implemented as a protocol extension. Inadequate implementations of the tracing facility may defeat safeguards built into the OPES architecture. The tracing facility by itself can become a target of malicious attacks or used to launch attacks on an OPES System.

Threats caused by or against the tracing facility can be viewed as threats at the application level in an OPES Flow. In this case, the threats can affect the data consumer and the data provider application.

Since tracing information is a protocol extension, these traces can be injected in the data flow by non-OPES entities. In this case, there are risks that non-OPES entities can be compromised in a fashion that threat the overall integrity and effectiveness of an OPES System. For example, a non-OPES proxy can add fake tracing information into a trace. This can be done in the form of wrong, or unwanted, or non-existent services. A non-OPES entity can inject large size traces that may cause buffer overflow in a data consumer application. The same threats can arise from compromised OPES entities. An attacker can control an OPES entity and inject wrong, or very large trace information that can overwhelm an end or the next OPES entity in an OPES flow. Similar threats can result from bad implementations of the tracing facility in trusted OPES entities.

Compromised tracing information can be used to launch attacks on an OPES System that give the impression that unwanted content transformation was performed on the data. This can be achieved by inserting wrong entity (such OPES processor) identifiers. A compromised trace can affect the overall message integrity structure. This can affect entities that use message header information to perform services such as accounting, load balancing, or reference-based services.

Compromised trace information can be used to launch DoS attacks that can overwhelm a data consumer application or an OPES entity in an OPES Flow. Inserting wrong tracing information can complicate the debugging tasks performed by system administrator during trouble shooting of OPES System behavior.

As a precaution, OPES entities ought to be capable of verifying that the inserted traces are performed by legal OPES entities. This can be done as part of the authorization and authentication face. Policy can be used to indicate what trace information can be expected from a peer entity. Other application level related security concerns can be found in [4].

8.2. Bypass security considerations

The bypass facility for OPES architecture is implemented as a protocol extension. Inadequate implementations of the bypass facility may defeat safeguards built into the OPES architecture. The bypass facility by itself can become a target of malicious attacks or used to lunch attacks on an OPES System.

Threats caused by or against the bypass facility can be viewed as threats at the application level in an OPES Flow. In this case, the threats can affect the data consumer and the data provider application.

There are risks for the OPES System by non-OPES entities, whereby, these entities can insert bypass instructions into the OPES Flow. The threat can come from compromised non-OPES entities. The threat might affect the overall integrity and effectiveness of an OPES System. For example, a non-OPES proxy can add bypass instruction to bypass legitimate OPES entities. The attack might result in overwhelming the original content provider servers, since the attack essentially bypass any load balancing techniques. In addition, such an attack is also equivalent to a DoS attack, whereby, a legitimate data consumer application may not be able to access some content from a content provider or its OPES version.

Since an OPES Flow may include non-OPES entities, it is susceptible to man-in-the-middle attacks, whereby an intruder may inject bypass instructions into the data path. These attacks may affect content availability or disturb load balancing techniques in the network.

The above threats can also arise by compromised OPES entities. An intruder can compromise an OPES entities and then use man-in-the-middle techniques to disturb content availability to a data consumer application or overload a content provider server (essentially, some form of a DoS attack).

Attackers can use the bypass instruction to affect the overall integrity of the OPES System. The ability to introduce bypass instructions into a data flow may effect the accounting of the OPES System. It may also affect the quality of content that is delivered to the data consumer applications. Similar threats can arise from bad implementations of the bypass facility.

Inconsistent or selective bypass is also a threat. Here, one end can try to bypass a subset of OPES entities so that the resulting content is malformed and crashes or compromises entities that process that content (and expect that content to be complete and valid). Such exceptions are often not tested because implementers do not expect a vital service to disappear from the processing loop.

Other threats can arise from configuring access control policies for OPES entities. It is possible that systems implementing access controls via OPES entities may be incorrectly configured to honor bypass and, hence, give unauthorized access to intruders.

Tap bypass can also be a threat. This is because systems implementing wiretaps via OPES entities may be incorrectly configured to honor bypass and, hence, ignore (leave undetected) traffic with bypass instructions that should have been tapped or logged. It is also possible for one end to bypass services such as virus scanning at the receiving end. This threat can be used by hackers to inject viruses throughout the network. Following an IETF policy on Wiretapping [7], OPES communication model does not consider wiretapping requirements. Nevertheless, the documented threat is real, not obvious, and OPES technology users operating in wiretapping or similar logging environments should be aware of it.

Other application level related security concerns can be found in [4].

9. References

9.1. 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] Barbir, A., Batuner, O., Beck, A., Chan, T., and H. Orman, "Policy, Authorization, and Enforcement Requirements of Open Pluggable Edge Services (OPES)", RFC 3838, August 2004.
- [4] Barbir, A., Batuner, O., Srinivas, B., Hofmann, M., and H. Orman, "Security Threats and Risks for Open Pluggable Edge Services (OPES)", RFC 3837, August 2004.

9.2 Informative References

- [5] Barbir A., Burger, E., Chen, R., McHenry, S., Orman, H., and R. Penno, "Open Pluggable Edge Services (OPES) Use Cases and Deployment Scenarios", RFC 3752, April 2004.
- [6] Floyd, S. and L. Daigle, "IAB Architectural and Policy Considerations for Open Pluggable Edge Services", RFC 3238, January 2002.
- [7] IAB and IESG, "IETF Policy on Wiretapping", RFC 2804, May 2000.

10. Acknowledgements

Several people has contributed to this work. Many thanks to: Alex Rousskov, Hilarie Orman, Oscar Batuner, Markus Huffman, Martin Stecher, Marshall Rose and Reinaldo Penno.

11. Author's Address

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

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

12. Full Copyright Statement

Copyright (C) The Internet Society (2004).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/S HE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights 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; nor does it represent that it has made any independent effort to identify any such rights. Information on the IETF's procedures with respect to rights in IETF Documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat 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 on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

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

