Workgroup: IPPM Working Group
Internet-Draft:
draft-ietf-ippm-ioam-conf-state-01
Published: 24 October 2021
Intended Status: Standards Track
Expires: 27 April 2022
Authors: X. Min G. Mirsky L. Bo
ZTE Corp. Ericsson China Telecom

Echo Request/Reply for Enabled In-situ OAM Capabilities

Abstract

This document describes an extension to the echo request/reply mechanisms used in IPv6 (including SRv6), MPLS (including SR-MPLS), SFC and BIER environments, which can be used within the IOAM domain, allowing the IOAM encapsulating node to discover the enabled IOAM capabilities of each IOAM transit node and IOAM decapsulating node.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 27 April 2022.

Table of Contents

1. Introduction

In-situ OAM (IOAM) ([I-D.ietf-ippm-ioam-data] [I-D.ietf-ippm-ioam-direct-export]) defines data fields that record OAM information within the packet while the packet traverses a particular network domain, called an IOAM domain. IOAM can be used to complement OAM mechanisms based on, e.g., ICMP or other types of probe packets, and IOAM mechanisms can be leveraged where mechanisms using, e.g., ICMP, do not apply or do not offer the desired results.

As specified in [I-D.ietf-ippm-ioam-data], within the IOAM domain, the IOAM data may be updated by network nodes that the packet traverses. The device which adds an IOAM header to the packet is called an "IOAM encapsulating node". In contrast, the device which removes an IOAM header is referred to as an "IOAM decapsulating node". Nodes within the domain that are aware of IOAM data and read and/or write and/or process IOAM data are called "IOAM transit nodes". IOAM encapsulating or decapsulating nodes can also serve as IOAM transit nodes at the same time. IOAM encapsulating or decapsulating nodes are also referred to as IOAM domain edge devices, which can be hosts or network devices.

To add the correct IOAM header to the data packet, the IOAM encapsulating node needs to know the enabled IOAM capabilities at the IOAM transit and IOAM decapsulating nodes as a whole. For example, how many IOAM transit nodes will add tracing data and what kinds of data fields will be added. A centralized controller which owns the enabled IOAM capabilities of each IOAM device could be used in some IOAM deployments. The IOAM encapsulating node can discover the enabled IOAM capabilities info from the centralized controller, using, for example, NETCONF/YANG, PCEP, or BGP. In the IOAM deployment scenario where there is no centralized controller, NETCONF/YANG or IGP may be used by the IOAM encapsulating node to discover these IOAM capabilities information. However, NETCONF/YANG or IGP has some limitations:

*When NETCONF/YANG is used in this scenario, each IOAM encapsulating node (including the host when it takes the role of an IOAM encapsulating node) needs to implement a NETCONF Client, each IOAM transit node and IOAM decapsulating node (including the host when it takes the role of an IOAM decapsulating node) needs to implement a NETCONF Server, the complexity can be an issue. Furthermore, each IOAM encapsulating node needs to establish NETCONF Connection with each IOAM transit node and IOAM decapsulating node, the scalability can be an issue.

*When IGP is used in this scenario, the IGP and IOAM domains don't always have the same coverage. For example, when the IOAM encapsulating node or the IOAM decapsulating node is a host, the availability can be an issue. Furthermore, it might be too challenging to reflect enabled IOAM capabilities at the IOAM transit node and IOAM decapsulating node if these are controlled by a local policy depending on the identity of the IOAM encapsulating node.

This document describes an extension to the echo request/reply mechanisms used in IPv6 (including SRv6), MPLS (including SR-MPLS), SFC and BIER environments, which can be used within the IOAM domain, allowing the IOAM encapsulating node to discover the enabled IOAM capabilities of each IOAM transit node and IOAM decapsulating node.

The following documents contain references to the echo request/reply mechanisms used in IPv6 (including SRv6), MPLS (including SR-MPLS), SFC and BIER environments:

*[RFC4443] ("Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification"), [RFC4884] ("Extended ICMP to Support Multi-Part Messages") and [RFC8335] ("PROBE: A Utility for Probing Interfaces")

*[RFC8029] ("Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures")

*[I-D.ietf-sfc-multi-layer-oam] ("Active OAM for Service Function Chains in Networks")

*[I-D.ietf-bier-ping] ("BIER Ping and Trace")

The precondition for the feature described in this document to work is that the echo request reaches each IOAM transit node as the data packet traverses, so the feature is assumedly applied to explicit path (strict or loose), or there is only one path between the IOAM encapsulating node and the IOAM decapsulating node, or the echo request can experience the same ECMP processing as the data packet.

2. Conventions

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Abbreviations

BIER: Bit Index Explicit Replication

BGP: Border Gateway Protocol

ECMP: Equal-Cost Multipath

E2E: Edge to Edge

ICMP: Internet Control Message Protocol

IGP: Interior Gateway Protocol

IOAM: In-situ Operations, Administration, and Maintenance

LSP: Label Switched Path

MPLS: Multi-Protocol Label Switching

MBZ: Must Be Zero

MTU: Maximum Transmission Unit

NTP: Network Time Protocol

OAM: Operations, Administration, and Maintenance

PCEP: Path Computation Element (PCE) Communication Protocol

POSIX: Portable Operating System Interface

POT: Proof of Transit

PTP: Precision Time Protocol

SR-MPLS: Segment Routing with MPLS data plane

SRv6: Segment Routing with IPv6 data plane

SFC: Service Function Chain

TTL: Time to Live

3. IOAM Capabilities Formats

3.1. IOAM Capabilities Query Container

For echo request, IOAM Capabilities Query uses container which has the following format:

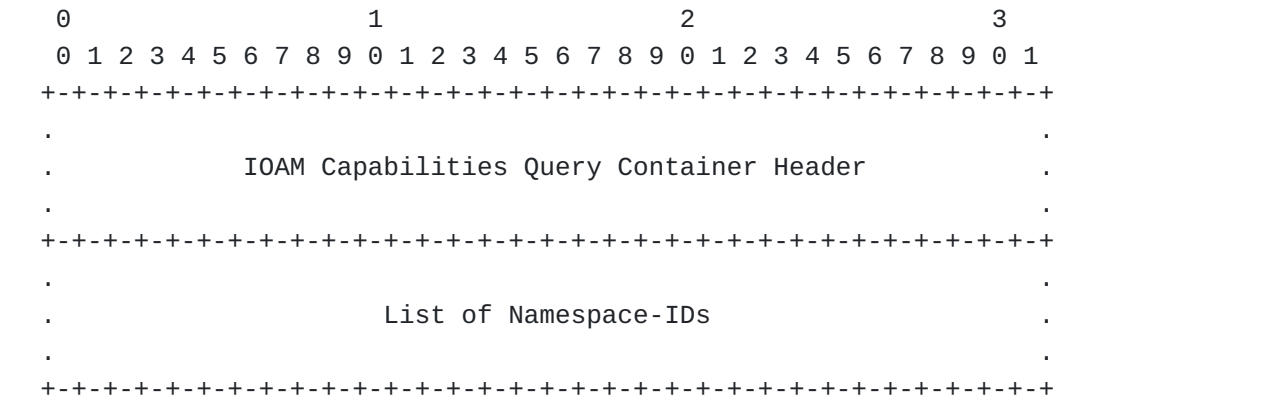

Figure 1: IOAM Capabilities Query Container of Echo Request

When this container is present in or equal to the echo request sent by an IOAM encapsulating node, that means the IOAM encapsulating node requests the receiving node to reply with its enabled IOAM capabilities. If there is no IOAM capability to be reported by the receiving node, then this container SHOULD be ignored by the receiving node, which means the receiving node SHOULD send an echo reply without IOAM capabilities or no echo reply, in the light of whether the echo request includes other containers than the IOAM Capabilities Query Container. A list of Namespace-IDs (one or more Namespace-IDs) MUST be included in this container in the echo

request. The IOAM encapsulating node requests only the enabled IOAM capabilities that match one of the Namespace-IDs. The Namespace-ID has the same definition as what's specified in Section 5.3 of [I-D.ietf-ippm-ioam-data].

The IOAM Capabilities Query Container has a container header that is used to identify the type and optionally length of the container payload, and the container payload (List of Namespace-IDs) is zero-padded to align to a 4-octet boundary.

The length, structure, and definition of the IOAM Capabilities Query Container Header depends on the specific environment it is applied at.

3.2. IOAM Capabilities Response Container

For echo reply, IOAM Capabilities Response uses container which has the following format:

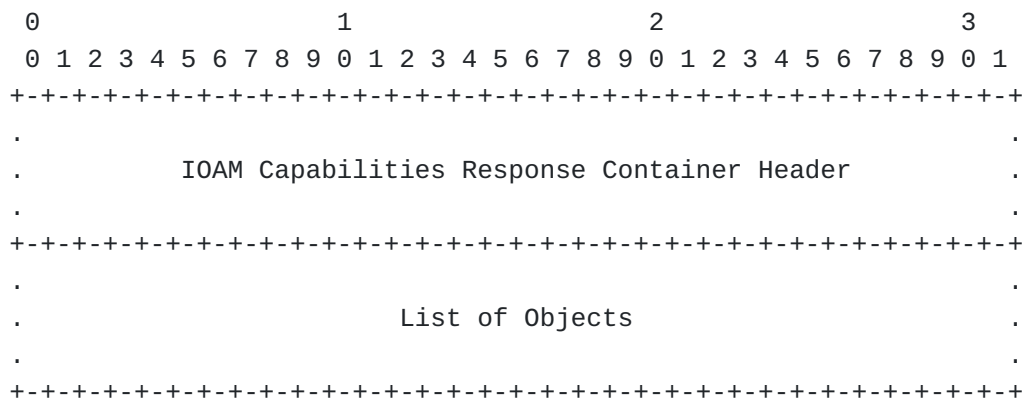

Figure 2: IOAM Capabilities Response Container of Echo Reply

When this container is present in or equal to the echo reply sent by an IOAM transit node or IOAM decapsulating node, that means the IOAM function is enabled at this node, and this container contains the enabled IOAM capabilities of the sender. A list of objects (one or more objects) which contains the enabled IOAM capabilities SHOULD be included in this container of echo reply.

The IOAM Capabilities Response Container has a container header that is used to identify the type and optionally length of the container payload, and the container payload (List of Objects) is zero-padded to align to a 4-octet boundary.

The length, structure, and definition of the IOAM Capabilities Response Container Header depends on the specific environment it is applied at.

Based on the IOAM data fields defined in [I-D.ietf-ippm-ioam-data] and [I-D.ietf-ippm-ioam-direct-export], six types of objects are defined in this document. The same type of object MAY be present in the IOAM Capabilities Response Container more than once, only if with a different Namespace-ID.

Similar to the container, each object has an object header that is used to identify the type and length of the object payload, and the object payload is zero-padded to align to a 4-octet boundary.

The length, structure, and definition of Object Header depends on the specific environment it is applied at.

3.2.1. IOAM Pre-allocated Tracing Capabilities Object

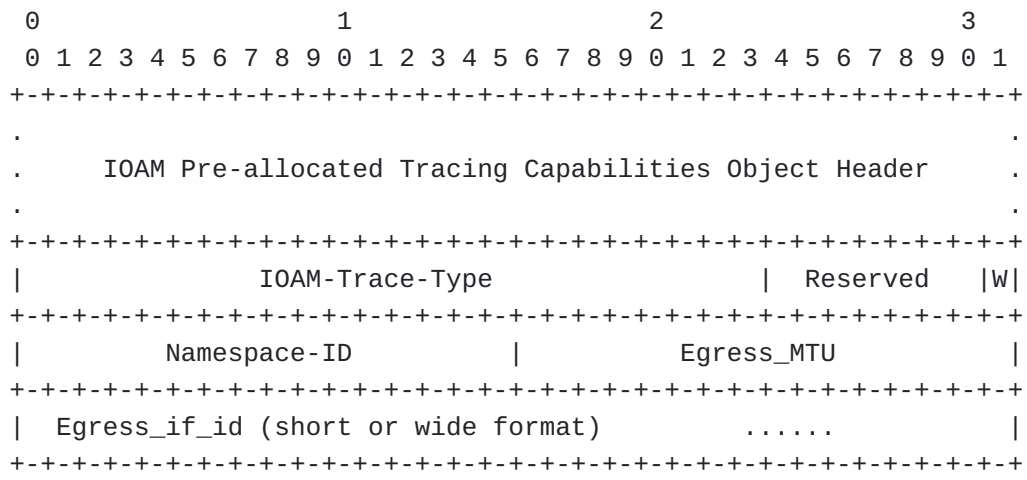

Figure 3: IOAM Pre-allocated Tracing Capabilities Object

When this Object is present in the IOAM Capabilities Response Container, that means the sending node is an IOAM transit node and IOAM pre-allocated tracing function is enabled at this IOAM transit node.

IOAM-Trace-Type field has the same definition as what's specified in Section 5.4 of [I-D.ietf-ippm-ioam-data].

Reserved field is reserved for future use and MUST be set to zero.

W flag indicates whether Egress_if_id is in short or wide format. The W-bit is set if the Egress_if_id is in wide format. The W-bit is clear if the Egress_if_id is in short format.

Namespace-ID field has the same definition as what's specified in Section 5.3 of [I-D.ietf-ippm-ioam-data], it should be one of the Namespace-IDs listed in the IOAM Capabilities Query Object of the echo request.

Egress_MTU field has 16 bits and specifies the MTU (in octets) of the egress direction out of which the sending node would forward the received echo request, it should be the MTU of the egress interface or the MTU between the sending node and the downstream IOAM transit node.

Egress_if_id field has 16 bits (in short format) or 32 bits (in wide format) and specifies the identifier of the egress interface out of which the sending node would forward the received echo request.

3.2.2. IOAM Incremental Tracing Capabilities Object

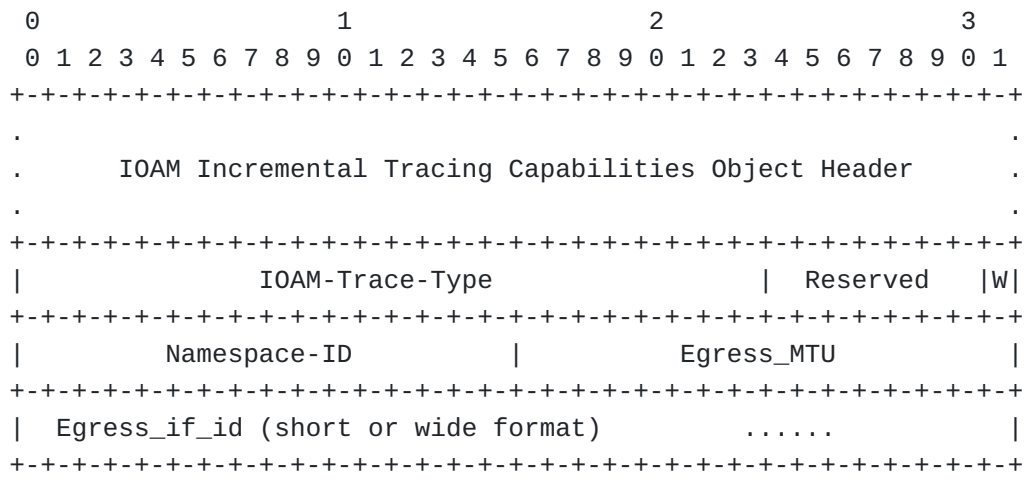

Figure 4: IOAM Incremental Tracing Capabilities Object

When this Object is present in the IOAM Capabilities Response Container, that means the sending node is an IOAM transit node and IOAM incremental tracing function is enabled at this IOAM transit node.

IOAM-Trace-Type field has the same definition as what's specified in Section 5.4 of [I-D.ietf-ippm-ioam-data].

Reserved field is reserved for future use and MUST be set to zero.

W flag indicates whether Egress_if_id is in short or wide format. The W-bit is set if the Egress_if_id is in wide format. The W-bit is clear if the Egress_if_id is in short format.

Namespace-ID field has the same definition as what's specified in Section 5.3 of [I-D.ietf-ippm-ioam-data], it should be one of the Namespace-IDs listed in the IOAM Capabilities Query Object of the echo request.

Egress_MTU field has 16 bits and specifies the MTU (in octets) of the egress direction out of which the sending node would forward the

received echo request, it should be the MTU of the egress interface or the MTU between the sending node and the downstream IOAM transit node.

Egress_if_id field has 16 bits (in short format) or 32 bits (in wide format) and specifies the identifier of the egress interface out of which the sending node would forward the received echo request.

3.2.3. IOAM Proof-of-Transit Capabilities Object

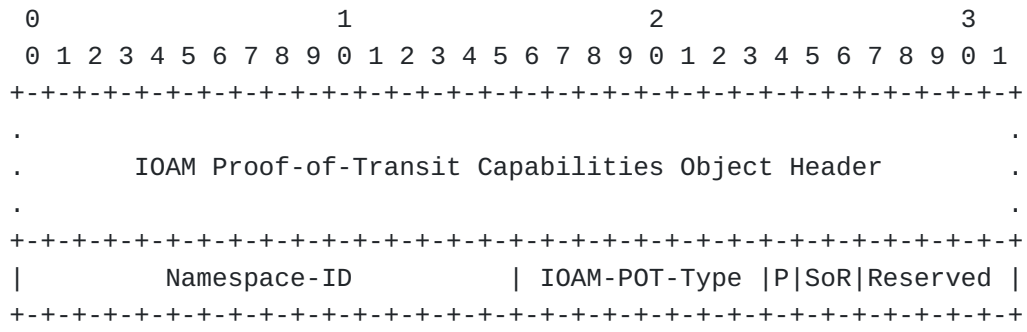

Figure 5: IOAM Proof-of-Transit Capabilities Object

When this Object is present in the IOAM Capabilities Response Container, that means the sending node is an IOAM transit node and IOAM Proof of Transit function is enabled at this IOAM transit node.

Namespace-ID field has the same definition as what's specified in Section 5.3 of [I-D.ietf-ippm-ioam-data], it should be one of the Namespace-IDs listed in the IOAM Capabilities Query Object of the echo request.

IOAM-POT-Type field and P bit have the same definition as what's specified in Section 5.5 of [I-D.ietf-ippm-ioam-data]. If the IOAM encapsulating node receives IOAM-POT-Type and/or P bit values from an IOAM transit node that are different from its own, then the IOAM encapsulating node MAY choose to abandon the proof of transit function or to select one kind of IOAM-POT-Type and P bit, it's based on the policy applied to the IOAM encapsulating node.

SoR field has two bits, which means the size of "Random" and "Cumulative" data that are specified in Section 5.5 of [I-D.ietf-ippm-ioam-data]. This document defines SoR as follow:

0b00 means 64-bit "Random" and 64-bit "Cumulative" data.

0b01~0b11: Reserved for future standardization

Reserved field is reserved for future use and MUST be set to zero.

3.2.4. IOAM Edge-to-Edge Capabilities Object

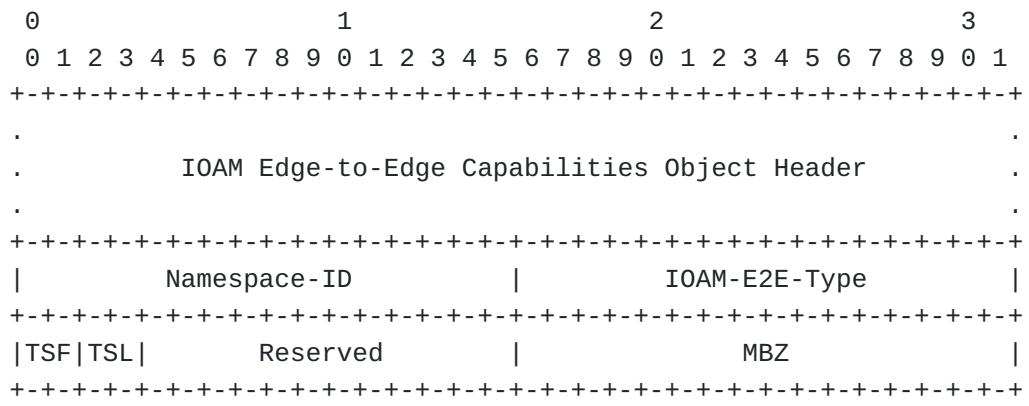

Figure 6: IOAM Edge-to-Edge Capabilities Object

When this Object is present in the IOAM Capabilities Response Container, that means the sending node is an IOAM decapsulating node and IOAM edge-to-edge function is enabled at this IOAM decapsulating node.

Namespace-ID field has the same definition as what's specified in Section 5.3 of [I-D.ietf-ippm-ioam-data], it should be one of the Namespace-IDs listed in the IOAM Capabilities Query Object of the echo request.

IOAM-E2E-Type field has the same definition as what's specified in Section 5.6 of [I-D.ietf-ippm-ioam-data].

TSF field specifies the timestamp format used by the sending node. This document defines TSF as follows:

0b00: PTP timestamp format

0b01: NTP timestamp format

0b10: POSIX timestamp format

0b11: Reserved for future standardization

TSL field specifies the timestamp length used by the sending node. This document defines TSL as follow.

When the TSF field is set to 0b00, which indicates the PTP timestamp format, the values of the TSL field are interpreted as follows:

0b00: 64-bit PTPv1 timestamp as defined in IEEE1588-2008 [IEEE1588v2]

0b01: 80-bit PTPv2 timestamp as defined in IEEE1588-2008 [IEEE1588v2]

0b10~0b11: Reserved for future standardization

When the TSF field is set to 0b01, which indicates the NTP timestamp format, the values of the TSL field are interpreted as follows:

0b00: 32-bit NTP timestamp as defined in NTPv4 [RFC5905]

0b01: 64-bit NTP timestamp as defined in NTPv4 [RFC5905]

0b10: 128-bit NTP timestamp as defined in NTPv4 [RFC5905]

0b11: Reserved for future standardization

When the TSF field is set to 0b10 or 0b11, the TSL field would be ignored.

Reserved field is reserved for future use and MUST be set to zero.

3.2.5. IOAM DEX Capabilities Object

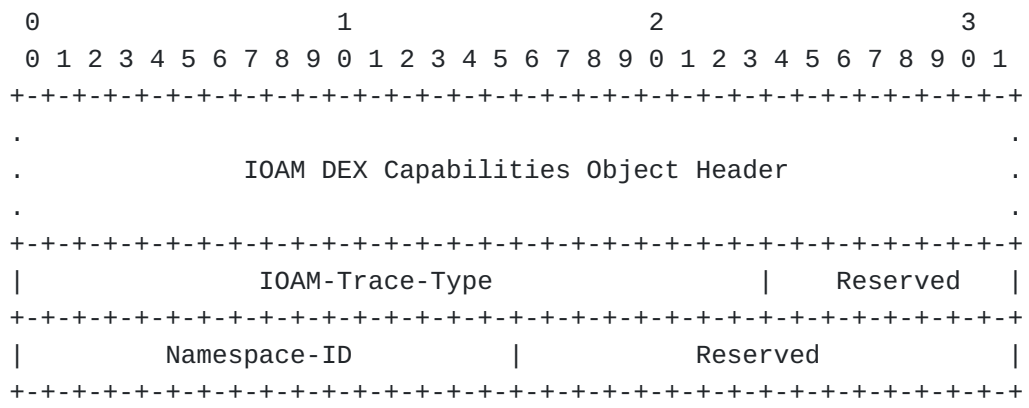

Figure 7: IOAM DEX Capabilities Object

When this Object is present in the IOAM Capabilities Response Container, that means the sending node is an IOAM transit node and the IOAM direct exporting function is enabled at this IOAM transit node.

IOAM-Trace-Type field has the same definition as what's specified in Section 3.2 of [I-D.ietf-ippm-ioam-direct-export].

Namespace-ID field has the same definition as what's specified in Section 5.3 of [I-D.ietf-ippm-ioam-data], it should be one of the

Namespace-IDs listed in the IOAM Capabilities Query Object of the echo request.

Reserved field is reserved for future use and MUST be set to zero.

3.2.6. IOAM End-of-Domain Object

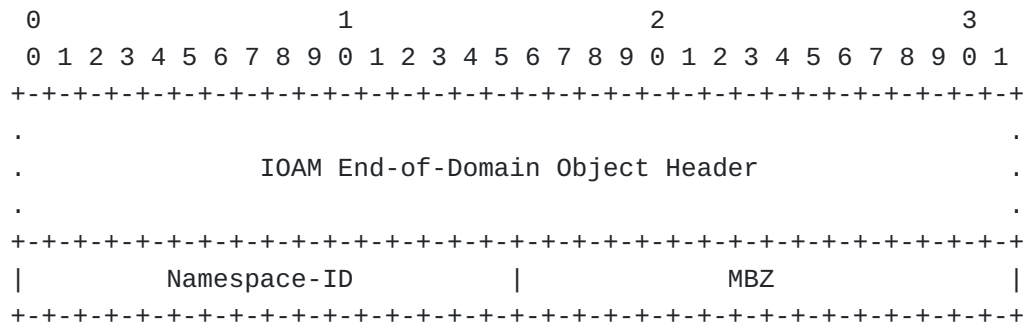

Figure 8: IOAM End-of-Domain Object

When this Object is present in the IOAM Capabilities Response Container, that means the sending node is an IOAM decapsulating node. Unless the IOAM Edge-to-Edge Capabilities Object is present, which also indicates that the sending node is an IOAM decapsulating node, the End-of-Domain Object MUST be present in the IOAM Capabilities Response Container sent by an IOAM decapsulating node. When the IOAM edge-to-edge function is enabled at the IOAM decapsulating node, it's RECOMMENDED to include only the IOAM Edge-to-Edge Capabilities Object but not the IOAM End-of-Domain Object.

Namespace-ID field has the same definition as what's specified in Section 5.3 of [I-D.ietf-ippm-ioam-data], it SHOULD be one of the Namespace-IDs listed in the IOAM Capabilities Query Container.

4. Operational Guide

Once the IOAM encapsulating node is triggered to discover the enabled IOAM capabilities of each IOAM transit node and IOAM decapsulating node, the IOAM encapsulating node will send echo requests that include the IOAM Capabilities Query Container. First, with TTL equal to 1 to reach the closest node, which may be an IOAM transit node or not. Then with TTL equal to 2 to reach the second nearest node, which also may be an IOAM transit node or not. And further, increasing by 1 the TTL every time the IOAM encapsulating node sends a new echo request, until the IOAM encapsulating node receives an echo reply sent by the IOAM decapsulating node, which should contain the IOAM Capabilities Response Container including the IOAM Edge-to-Edge Capabilities Object or the IOAM End-of-Domain Object. Alternatively, if the IOAM encapsulating node knows precisely all the IOAM transit nodes and IOAM decapsulating node

beforehand, once the IOAM encapsulating node is triggered to discover the enabled IOAM capabilities, it can send an echo request to each IOAM transit node and IOAM decapsulating node directly, without TTL expiration.

The IOAM encapsulating node may be triggered by the device administrator, the network management system, the network controller, or data traffic. The specific triggering mechanisms are outside the scope of this document.

Each IOAM transit node and IOAM decapsulating node that receives an echo request containing the IOAM Capabilities Query Container will send an echo reply to the IOAM encapsulating node. For the echo reply, there should be an IOAM Capabilities Response Container containing one or more Objects. The IOAM Capabilities Query Container of the echo request would be ignored by the receiving node unaware of IOAM.

5. IANA Considerations

This document requests the following IANA Actions.

IANA is requested to create a registry group named "In-Situ OAM (IOAM) Capabilities Parameters".

This group will include the following registries:

*IOAM SoR Capability

*IOAM TSF+TSL Capability

New registries in this group can be created via RFC Required process as per [RFC8126].

The subsequent sub-sections detail the registries herein contained.

Considering the Containers/Objects defined in this document would be carried in different types of Echo Request/Reply messages, such as ICMPv6 or LSP Ping, it is intended that the registries for Container/Object Type would be requested in subsequent documents.

5.1. IOAM SoR Capability Registry

This registry defines 4 code points for the IOAM SoR Capability field for identifying the size of "Random" and "Cumulative" data as explained in Section 5.5 of [I-D.ietf-ippm-ioam-data]. The following code points are defined in this document:

SoR	Description
0b00	64-bit "Random" and 64-bit "Cumulative" data

0b01 - 0b11 are available for assignment via RFC Required process as per [RFC8126].

5.2. IOAM TSF+TSL Capability Registry

This registry defines 3 code points for the IOAM TSF Capability field for identifying the timestamp format as explained in Section 6 of [I-D.ietf-ippm-ioam-data].

- *When the code point for the IOAM TSF Capability field equals 0b00 which means PTP timestamp format, this registry defines 2 code points for the IOAM TSL Capability field for identifying the timestamp length.

- *When the code point for the IOAM TSF Capability field equals 0b01 which means NTP timestamp format, this registry defines 3 code points for the IOAM TSL Capability field for identifying the timestamp length.

The following code points are defined in this document:

TSF	TSL	Description
0b00		PTP Timestamp Format
	0b00	64-bit PTPv1 timestamp
	0b01	80-bit PTPv2 timestamp
0b01		NTP Timestamp Format
	0b00	32-bit NTP timestamp
	0b01	64-bit NTP timestamp
	0b10	128-bit NTP timestamp
0b10		POSIX Timestamp Format

Unassigned code points of TSF+TSL are available for assignment via RFC Required process as per [RFC8126].

6. Security Considerations

Queries and responses about the state of an IOAM domain should be processed only from a trusted source. An unauthorized query MUST be discarded by an implementation that supports this specification. Similarly, an unsolicited echo response with the IOAM Capabilities Container MUST be discarded. Authentication of echo request/reply that includes the IOAM Capabilities Container is one of the integrity protection methods. Implementations could also provide a means of filtering based on the source address of the received echo request/reply. The integrity protection for enabled IOAM

capabilities information collection can also be achieved using mechanisms in the underlay data plane. For example, if the underlay is an IPv6 network, IP Authentication Header [RFC4302] or IP Encapsulating Security Payload Header [RFC4303] can be used to provide integrity protection.

Information about the state of the IOAM domain collected in the IAOM Capabilities Container is confidential. An implementation can use secure transport to provide privacy protection. For example, if the underlay is an IPv6 network, confidentiality can be achieved using the IP Encapsulating Security Payload Header [RFC4303].

7. Acknowledgements

The authors would like to acknowledge Tianran Zhou, Dhruv Dhody, Frank Brockners, Cheng Li and Gyan Mishra for their careful review and helpful comments.

The authors appreciate the f2f discussion with Frank Brockners on this document.

The authors would like to acknowledge Tommy Pauly and Ian Swett for their good suggestion and guidance.

8. References

8.1. Normative References

[I-D.ietf-ippm-ioam-data] Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields for In-situ OAM", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-data-15, 3 October 2021, <https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-data-15.txt>.

[I-D.ietf-ippm-ioam-direct-export] Song, H., Gafni, B., Zhou, T., Li, Z., Brockners, F., Bhandari, S., Sivakolundu, R., and T. Mizrahi, "In-situ OAM Direct Exporting", Work in Progress, Internet-Draft, draft-ietf-ippm-ioam-direct-export-07, 13 October 2021, <https://www.ietf.org/archive/id/draft-ietf-ippm-ioam-direct-export-07.txt>.

[IEEE1588v2] IEEE, "IEEE Std 1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588-2008, 2008, <http://standards.ieee.org/findstds/standard/1588-2008.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/

RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[I-D.ietf-bier-ping] Kumar, N., Pignataro, C., Akiya, N., Zheng, L., Chen, M., and G. Mirsky, "BIER Ping and Trace", Work in Progress, Internet-Draft, draft-ietf-bier-ping-07, 11 May 2020, <https://www.ietf.org/archive/id/draft-ietf-bier-ping-07.txt>.

[I-D.ietf-sfc-multi-layer-oam] Mirsky, G., Meng, W., Ao, T., Leung, K., and G. Mishra, "Active OAM for Service Function Chaining", Work in Progress, Internet-Draft, draft-ietf-sfc-multi-layer-oam-16, 19 October 2021, <https://www.ietf.org/archive/id/draft-ietf-sfc-multi-layer-oam-16.txt>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <https://www.rfc-editor.org/info/rfc4303>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4884] Bonica, R., Gan, D., Tappan, D., and C. Pignataro, "Extended ICMP to Support Multi-Part Messages", RFC 4884, DOI 10.17487/RFC4884, April 2007, <https://www.rfc-editor.org/info/rfc4884>.

[RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <https://www.rfc-editor.org/info/rfc8029>.

[RFC8335] Bonica, R., Thomas, R., Linkova, J., Lenart, C., and M. Boucadair, "PROBE: A Utility for Probing Interfaces", RFC 8335, DOI 10.17487/RFC8335, February 2018, <https://www.rfc-editor.org/info/rfc8335>.

Authors' Addresses

Xiao Min
ZTE Corp.
Nanjing
China

Phone: +86 25 88013062
Email: xiao.min2@zte.com.cn

Greg Mirsky
Ericsson
United States of America

Email: gregimirsky@gmail.com

Lei Bo
China Telecom
Beijing
China

Phone: +86 10 50902903
Email: leibo@chinatelecom.cn