

RIFT
Internet-Draft
Intended status: Standards Track
Expires: 12 June 2026

J. Head, Ed.
T. Przygienda
Hewlett Packard Enterprise
9 December 2025

Routing in Fat Trees (RIFT) Key/Value Topology Information Elements
Structure and Processing
draft-ietf-rift-kv-tie-structure-and-processing-06

Abstract

The RIFT (Routing in Fat Trees) protocol allows for key/value pairs to be advertised within Key-Value Topology Information Elements (KV TIES). The data contained within these KV TIES can be used for any imaginable purpose.

This document defines the various Key-Types (i.e., Well-Known, OUI, and Experimental) and a method to structure corresponding values. It also defines a Well-Known Key Sub-Type used for testing tie-breaking behavior.

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.

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 12 June 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
2. Key-Value Structure	3
2.1. Key Sub-Type	4
2.2. Experimental Key-Type	5
2.3. Well-Known Key-Type	5
2.4. OUI Key-Type	6
3. Design Considerations	6
3.1. Tie-Breaking Considerations	6
3.1.1. Southbound Key-Value TIE Tie-Break Sub-Type	7
3.2. Key Target	8
3.2.1. Key Target Processing	9
4. IANA Considerations	10
4.1. RIFT Key-Types	10
4.1.1. RIFT Key-Types Requested Entries	11
4.2. RIFT Well-Known Key Sub-Types	11
4.2.1. RIFT Well-Known Key Sub-Types Requested Entries	11
4.3. Expert Review Guidance	12
5. Security Considerations	12
6. Acknowledgements	12
7. Normative References	13
8. Informative References	13
Appendix A. Thrift Models	13
A.1. southbound_kv.thrift	13
Authors' Addresses	14

1. Introduction

The Routing in Fat Trees [RFC9692] protocol allows for key/value pairs to be advertised within Key-Value Topology Information Elements (KV TIEs). There are no restrictions placed on the type of data that is contained in KV TIEs nor what the data is used for.

For example, it might be beneficial to advertise overlay protocol state from leaf nodes to the Top-of-Fabric (ToF) nodes. This would make it possible to view critical state of a fabric-wide service from a single ToF node rather than retrieving and reconciling the same state from multiple leaf nodes.

2. Key-Value Structure

This section describes the generic key structure and semantics, Figure 1 further illustrates these components.

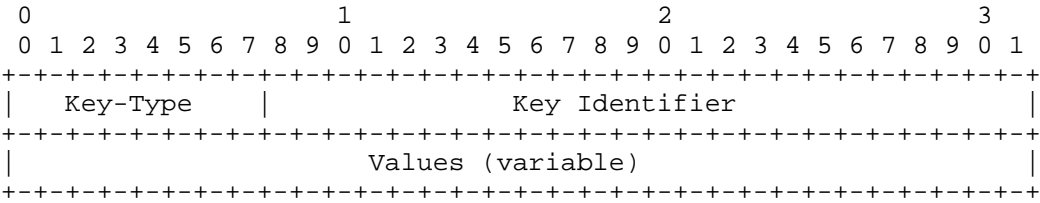


Figure 1: Generic Key-Value Structure

where:

Key-Type:

A 1-byte value that identifies the Key-Type. It MUST be a reserved value from the RIFT Key-Type Registry that is defined later in this document.

The range of valid values is 1 - 255 (2^8-1).

0 is an illegal value and MUST NOT be allocated to or used by any implementation. KV TIEs received with this value and MUST be discarded and logged on receipt.

Key Identifier:

A 3-byte value that identifies the specific key and describes and the semantics of any contained values. It SHOULD be unique within the context of the given Key-Type.

The range of valid values is 1 - 16777215 (2^24-1).

0 is an illegal value and MUST NOT be allocated to or used by any implementation. KV TIEs received with this value and MUST be discarded and logged on receipt.

Values:

A variable length value that contains data associated with the Key Identifier. It SHOULD contain 1 or more elements. The semantics (i.e., existence, order, duplication, etc.) of any contained values is governed by the particular key's specification.

2.1. Key Sub-Type

The Key Sub-Type is a mechanism to further describe the key's semantics. This is illustrated by Figure 2. The Key Sub-Type MUST be used when the Key-Type is either Well-Known or Experimental in order to avoid interoperability issues, but is OPTIONAL for other Key-Types.

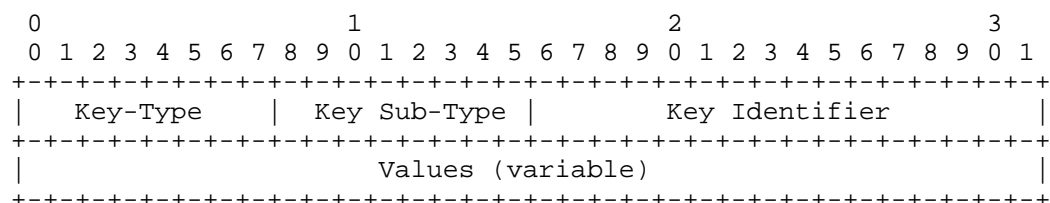


Figure 2: Generic Key-Value Structure with Key Sub-Type

where:

Key Sub-Type:

A 1-byte value that identifies the Key Sub-Type which describes the key and its semantics.

The range of valid values is 1 - 255 (2^8-1).

0 is an illegal value and MUST NOT be allocated to or used by any implementation. KV TIES received with this value and MUST be discarded and logged on receipt.

Key Identifier:

If the Key Sub-Type is used, it inherently reduces the Key Identifier space from 3-bytes to 2-bytes. The Key Identifier SHOULD be unique within the context of the given Key Sub-Type. The Key Identifier is otherwise unchanged.

The range of valid values is now 1 - 65535 ($2^{16}-1$).

0 is an illegal value and MUST NOT be allocated to or used by any implementation. KV TIES received with this value and MUST be discarded and logged on receipt.

2.2. Experimental Key-Type

This section reserves a value in the RIFT Key-Type Registry to indicate an Experimental Key-Type.

As shown in Figure 3, the Key-Type will be used to identify the Key-Type as Experimental. The Experimental Key-Type MUST support the use of a Key Sub-Type. The Key Identifier will be used to identify the specific key and the semantics of any contained values.

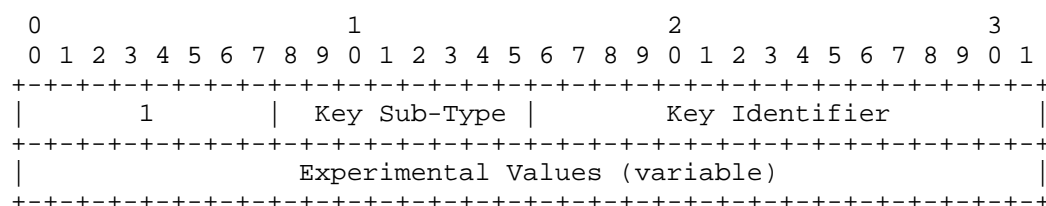


Figure 3: Experimental Key-Type

2.3. Well-Known Key-Type

This section reserves a value in the RIFT Key-Type Registry to indicate a Well-Known Key-Type.

As shown in Figure 4, the Key-Type will be used to identify the Key-Type as Well-Known. The Well-Known Key-Type MUST support the use of a Key Sub-Type. The Key Identifier will be used to identify the specific key and the semantics of any contained values.

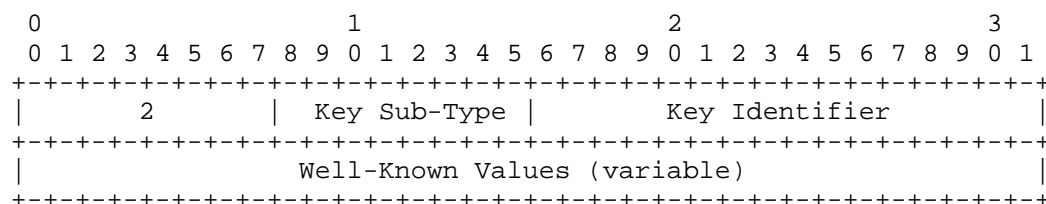


Figure 4: Well-Known Key-Type

2.4. OUI Key-Type

This section reserves a value in the RIFT Key-Type Registry to indicate an OUI (vendor-specific) Key-Type that any implementation MAY support.

As shown in Figure 5, the Key-Type will be used to identify the Key-Type as OUI. The Key Identifier MUST use the implementing organization's reserved OUI space to indicate the key and the semantics of any contained values.

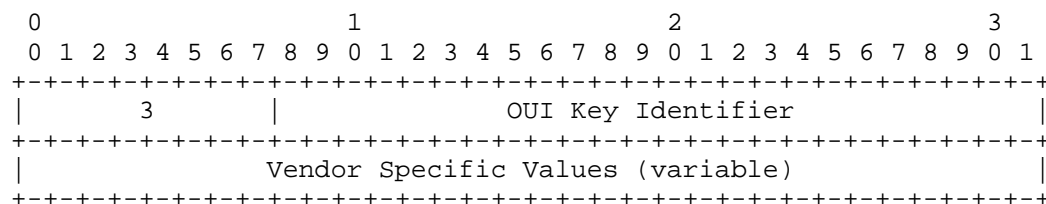


Figure 5: OUI Key-Type

3. Design Considerations

NOTE:This document uses the terms to denote directionality, specifically, "northbound" meaning "toward the top of the fabric" and "southbound" meaning "toward the bottom of the fabric".

While no restrictions are placed on Key-Value data or what it is used for, it is RECOMMENDED that a serialized Thrift [THRIFT] model be used for simpler interoperability. [RIFT-AUTO-EVPN] is an example of this type of implementation.

Key-Value elements SHOULD NOT be used to carry topology information used by RIFT itself to perform distributed computations.

It is possible that deployments may have nodes with differing software implementations where some nodes support a given KV TIE and others do not. In this scenario, nodes that receive KV TIEs that they don't recognize (e.g., an unknown Key-Type) will continue to flood them as specified in RIFT [RFC9692].

3.1. Tie-Breaking Considerations

In cases where KV TIEs are flooded southbound, policies SHOULD be implemented in order to avoid network-wide flooding.

For networks with more than one ToF node, it is RECOMMENDED that those ToF nodes contain identical KV TIE information when being distributed southbound. RIFT [RFC9692] requires that only one KV TIE is selected when identical keys are received from multiple northbound neighbors. If this is not considered, then the tie-breaking rules may cause a node to select a suboptimal KV TIE. Consider a case where failure conditions cause the ToF nodes to become split-brained. While the key will be identical, the value(s) contained within may differ. The node(s) receiving these differing KV TIEs will select the one from the ToF node with the highest System ID, potentially leading to unintended effects.

3.1.1.1. Southbound Key-Value TIE Tie-Break Sub-Type

This section reserves a Key Sub-Type from the RIFT Well-Known Key Sub-Types registry.

This Key-Value pair contains information that allows implementations to test and verify proper tie-breaking behavior for the Southbound Key store. All implementations SHOULD support this Sub-Type.

All implementations SHOULD use the Appendix A.1 Thrift model

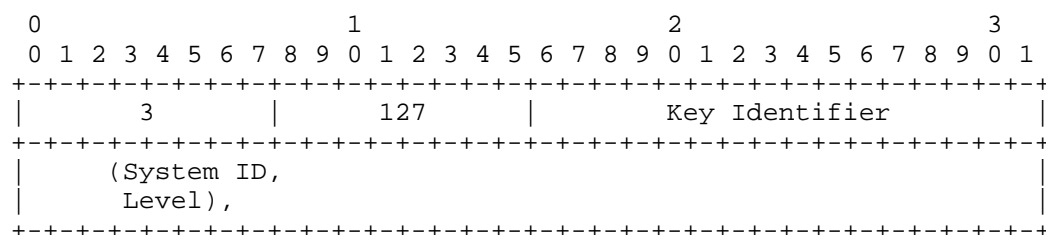


Figure 6: Southbound Tie-Break Sub-Type

where:

System ID:

A REQUIRED value indicating the node's unique System ID.

Level:

A RECOMMENDED value indicating the node's level.

3.2. Key Target

The Key Target is an OPTIONAL value that identifies group(s) of node(s) that are intended to receive a given Key-Value TIE. Key Targets are 64-bits in length with a valid range of 0 - 18446744073709551615 ($2^{64}-1$), this will reduce the chances that Key Target values collide.

Figure 7 illustrates the format.

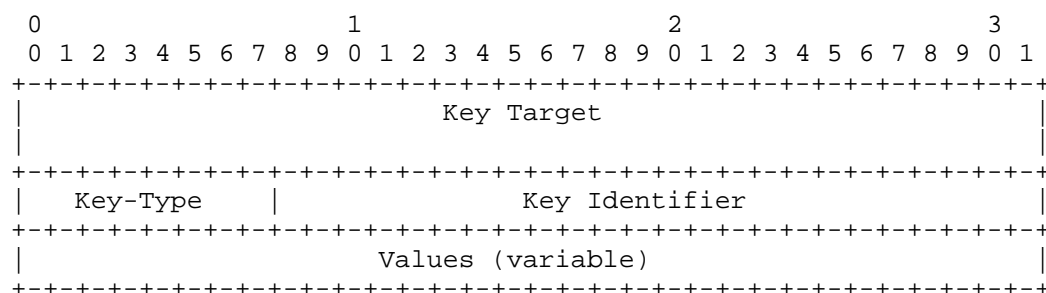


Figure 7: Key Target Format

A value of all 0s indicates that every node is intended to receive this Key-Value TIE and MUST NOT be used for any other reason.

A value of all 1s indicates that all leaf nodes are intended to receive this Key-Value TIE and MUST NOT be used for any other reason.

Any other value MUST be derived from the following normative algorithm. Note that while the algorithm is shown using example code written in [Rust], this document does not mandate the use of any particular language for implementation.


```

<CODE BEGINS>
/// random seeds used in algorithms to increase entropy
pub const RANDOMSEEDS: [UnsignedSystemID; 3] = [
    67438371571u64,
    37087353685,
    88675895388,
];

/// given a system ID delivers the bits set by the according Bloom Filter in the southbound
/// key value target.
///
/// @note: This is standardized and cannot be changed between releases!
pub (crate) fn target2bits(target: UnsignedSystemID) -> KeyValueTargetType {
    (0 as usize .. 3)
        .map(|s| {
            let rot = (target ^ RANDOMSEEDS[s]).rotate_left(s as _);
            rot.to_ne_bytes().iter().fold(0, |v: u8, nv| v.rotate_right(4) ^ *nv) % 64
        })
        .fold(0, |v, nv| v | (1 << nv))
}
<CODE ENDS>

```

Figure 8: Key Target Standard Algorithm

3.2.1. Key Target Processing

Nodes that support the processing of Key Targets MUST only do so on KV TIEs in the southbound direction. Key Targets MUST NOT be present on KV TIEs in the northbound direction and are otherwise ignored and logged.

Nodes that do not support the processing of Key Targets MUST continue to send KV TIEs to all nodes in the appropriate direction. Additionally, Key Targets MUST be preserved when KV TIEs are re-originated in the southbound direction.

3.2.1.1. Purging/Rollover

There are several reasons a node may select a different KV TIE. For example, the KV TIE is considered newer due to the sequence number incrementing, there was a change in the original tie-breaking result between multiple KV TIEs, or a loss of northbound connectivity to the node that advertised the previously selected KV TIE.

Consider a case where Leaf-1, Leaf-2, and Leaf-3 are members of a group of nodes represented by Key Target KT1. If Leaf-2 is removed from that group and a newer instance of the KV TIE needs to be flooded Leaf-2 will have to maintain the older KV TIE in the LSDB until the lifetime expires. This could lead to suboptimal behavior in the fabric.

If the new KV TIE being flooded does not include the previous Key Target value, then implementations SHOULD flood the newer instance of the KV TIE with a very short lifetime to nodes that belonged to the previous Key Target but not the new Key Target. This logic only applies to KV TIEs being flooded in the southbound direction.

4. IANA Considerations

Per [RFC8126], IANA is requested to create two new registries in the "Routing in Fat Trees (RIFT)" registry group at <https://www.iana.org/assignments/rift>

- * RIFT Key-Types

- * RIFT Well-Known Key Sub-Types

The following sections detail each registry's individual requirements and suggested values.

Experts reviewing requests for new values to either registry MUST consider the items in the Expert Review Guidance (Section 4.3) section.

4.1. RIFT Key-Types

This section requests that IANA create and help govern the following registry:

- *Registry Name:*
RIFT Key-Types

- *Registration Procedures:*
Expert Review

- *Description:*
Key-Type registry for the RIFT protocol.

- *Reference:*
This document.

4.1.1. RIFT Key-Types Requested Entries

This section requests that IANA register the following suggested values to the "RIFT Key-Types" registry.

Value	Key-Type	Description	Reference
0	Illegal	Not allowed.	This document
1	Experimental	Indicates that the Key-Type is Experimental.	This document.
2	Well-Known	Indicates that the Key-Type is Well-Known.	This document.
3	OUI	Indicates that the Key-Type is OUI (vendor specific).	This document.

Table 1: RIFT Key-Types Requested Entries

4.2. RIFT Well-Known Key Sub-Types

This section requests that IANA create and help govern the following registry:

```
*Registry Name:*
  RIFT Well-Known Key Sub-Types

*Registration Procedures:*
  Expert Review

*Description:*
  Well-Known Key Sub-Types registry for the RIFT protocol.

*Reference:*
  This document.
```

4.2.1. RIFT Well-Known Key Sub-Types Requested Entries

This section requests that IANA register the following suggested values to the "RIFT Well-Known Key Sub-Types" Registry.

Value	Key-Identifier	Description	Reference
0	Illegal	Not allowed.	This document.
127	Southbound Tie-Break Key Sub-Type	Used for Southbound Keystore tie-breaking testing and verification.	This document.

Table 2: RIFT Well-Known Key Sub-Types Requested Entries

4.3. Expert Review Guidance

Experts reviewing requests for values from the "RIFT Key-Types" registry or the "RIFT Well-Known Key-Types" registry are responsible for the following:

1. Determining the existence of a specification that clearly defines the purpose supporting the request and MUST contain all required fields for given registry.

The document MUST also be permanent and publicly available.

2. Ensuring that the specification properly defines if Key Sub-Types are used and how the Key Identifier value is derived.
3. Ensuring that any requests are made available to the RIFT working group for review should the work originate from outside the RIFT Working Group.
4. Ensuring that any work produce outside of the IETF does not conflict with any work that is already published or actively pursuing being published.

5. Security Considerations

This document introduces no new security concerns to RIFT or other specifications referenced in this document given that the Key-Value TIEs are already extensively secured by the RIFT [RFC9692] protocol specification itself.

6. Acknowledgements

Thanks to Italo Busi for his very thoughtful review which yielded and improved spec.

7. Normative References

- [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>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", 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>>.
- [RFC9692] Przygienda, T., Ed., Head, J., Ed., Sharma, A., Thubert, P., Rijnsman, B., and D. Afanasiev, "RIFT: Routing in Fat Trees", RFC 9692, DOI 10.17487/RFC9692, April 2025, <<https://www.rfc-editor.org/info/rfc9692>>.

8. Informative References

- [RIFT-AUTO-EVPN] Head, J., Przygienda, T., and W. Lin, "RIFT Auto-EVPN", Work in Progress, draft-ietf-rift-auto-evpn-06, January 2025, <<https://www.ietf.org/archive/id/draft-ietf-rift-auto-evpn-06.html>>.
- [Rust] Rust Foundation, "The Rust Reference", <<https://doc.rust-lang.org/reference/>>.
- [THRIFT] Apache Software Foundation, "Thrift Language Implementation and Documentation", <<https://github.com/apache/thrift/tree/0.15.0/doc>>.

Appendix A. Thrift Models

This section contains the Thrift model that MAY be used to test southbound Key-Value tie-breaking based on System ID. Per the main RIFT [RFC9692] specification, all signed values MUST be interpreted as unsigned values.

A.1. southbound_kv.thrift

```
include "common.thrift"

namespace py southbound_kv
namespace rs models

const i8 GlobalSystemIdentifierKV = 127

/** simple type to test correct tie-breaking based on system ID */
struct SystemIdentifierKV {
    1: required common.SystemIDType system_id,
    2: optional common.LevelType level,
}
```

Figure 9: RIFT Common Schema for Southbound Key-Value Tie-Break
Key Sub-Type

Authors' Addresses

Jordan Head (editor)
Hewlett Packard Enterprise
1137 Innovation Way
Sunnyvale, CA
United States of America
Email: jhead@juniper.net

Tony Przygienda
Hewlett Packard Enterprise
1137 Innovation Way
Sunnyvale, CA
United States of America
Email: prz@juniper.net