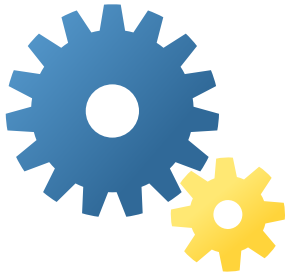


DHCPKit

DHCPKit Documentation

Release 1.0.0

S.J.M. Steffann

Sep 27, 2017

Contents

This package contains contains extensions to DHCPKit for sending detailed information about each processed DHCPv6 transaction to Kafka. This can be used to implement a looking glass and other monitoring tools.

CHAPTER 1

Distribution status

12345

[1] https://pypi.python.org/pypi/dhcpkit_kafka
[2] https://pypi.python.org/pypi/dhcpkit_kafka
[3] https://pypi.python.org/pypi/dhcpkit_kafka
[4] https://pypi.python.org/pypi/dhcpkit_kafka
[5] https://pypi.python.org/pypi/dhcpkit_kafka

CHAPTER 2

Documentation

IPv6 Server extension configuration

This is a server extension that publishes information on what the DHCPv6 server is doing to Kafka. You can then run a collector that collects the information from all the DHCPv6 servers and presents them on a single dashboard.

Overview of section types

Handlers

Configuration sections that specify a handler. Handlers process requests, build the response etc. Some of them add information options to the response, others look up the client in a CSV file and assign addresses and prefixes, and others can abort the processing and tell the server not to answer at all.

You can make the server do whatever you want by configuring the appropriate handlers.

Send-to-kafka

This section specifies the Kafka server cluster that data should be sent to.

Example

```
<send-to-kafka>
    broker host1:9092
    broker host2:9092

    topic dhcpkit.messages
</send-to-kafka>
```

Section parameters

server-name The name of this DHCPv6 server to label Kafka messages with

Default: The FQDN of the server

source-address The source address to use when connecting to Kafka

Example: "dhcp01.example.com"

topic The Kafka topic to publish DHCPKit messages on

Default: "dhcpkit.messages"

broker (multiple allowed) Kafka broker to connect to

Default: "localhost:9092"

dhcpkit_kafka package

Basic information about this package

Subpackages

dhcpkit_kafka.server_extension package

This handler provides a looking glass into DHCP server operations by sending interesting information on Kafka

class `dhcpkit_kafka.server_extension.`**`KafkaHandler`**(*source_address: typing.Tuple[str, int]*, *brokers: typing.Iterable[typing.Tuple[str, int]]*, *topic_name: str*, *server_name: str*)

Bases: `dhcpkit.ipv6.server.handlers.Handler`

Option handler that provides a looking glass into DHCP server operations by logging information about requests and responses into an SQLite database.

The primary key is (duid, interface_id, remote_id)

`analyse_post`(*bundle: dhcpkit.ipv6.server.transaction_bundle.TransactionBundle*)

Finish the Kafka message and send it.

Parameters **`bundle`** – The transaction bundle

`analyse_pre`(*bundle: dhcpkit.ipv6.server.transaction_bundle.TransactionBundle*)

Start building the Kafka message.

Parameters **`bundle`** – The transaction bundle

`connect`()

Connect the producer to the broker.

`kafka` = None

The Kafka client

`kafka_producer` = None

The Kafka producer

`kafka_topic` = None

The Kafka topic we publish to

`last_connect_attempt` = None

Remember when the last connection attempt was for reconnect rate-limiting

`worker_init`()

Initialise the Kafka client in each worker

Submodules

dhcpkit_kafka.server_extension.config module

Configuration elements for the SOL_MAX_RT option handlers

class dhcpkit_kafka.server_extension.config.**KafkaHandlerFactory**(*section: ZConfig.matcher.SectionValue*)

Bases: dhcpkit.ipv6.server.handlers.HandlerFactory

Create the handler for the Kafka producer.

create() → dhcpkit_kafka.server_extension.KafkaHandler

Create a handler of this class based on the configuration in the config section.

Returns A handler object

dhcpkit_kafka.server_extension.config.**topic_name**(*value: str*) → str

Data validation for Kafka topic names.

Parameters **value** – The value from the configuration file

Returns The validated value

dhcpkit_kafka.tests package

Subpackages

dhcpkit_kafka.tests.messages package

Submodules

dhcpkit_kafka.tests.messages.test_dhcp_kafka_message module

Test the UnknownKafkaMessage implementation

class dhcpkit_kafka.tests.messages.test_dhcp_kafka_message.**DHCPKafkaMessageTestCase**(*methodN*

Bases: *dhcpkit_kafka.tests.messages.test_kafka_message.KafkaMessageTestCase* (page 8)

parse_packet()

setUp()

test_load_wrong_type()

test_validate_message_in()

test_validate_message_out()

test_validate_server_name()

test_validate_timestamp_in()

test_validate_timestamp_out()

class dhcpkit_kafka.tests.messages.test_dhcp_kafka_message.**NoInboundMessageDHCPKafkaMessag**

Bases: *dhcpkit_kafka.tests.messages.test_kafka_message.KafkaMessageTestCase* (page 8)

parse_packet()

setUp()

class dhcpkit_kafka.tests.messages.test_dhcp_kafka_message.**NoOutboundMessageDHCPKafkaMessa**
Bases: *dhcpkit_kafka.tests.messages.test_kafka_message.KafkaMessageTestCase* (page 8)

parse_packet()

setUp()

dhcpkit_kafka.tests.messages.test_kafka_message module

Test the KafkaMessage implementation

class dhcpkit_kafka.tests.messages.test_kafka_message.**KafkaMessageTestCase**(*methodName='runTes*
Bases: unittest.case.TestCase

check_unsigned_integer_property(*property_name: str*, *size: int = None*)
Perform basic verification of validation of an unsigned integer

Parameters

- **property_name** – The property under test
- **size** – The number of bits of this integer field

parse_packet()

setUp()

test_length()

test_parse()

test_save_fixture()

test_save_parsed()

test_validate()

dhcpkit_kafka.tests.messages.test_unknown_kafka_message module

Test the UnknownKafkaMessage implementation

class dhcpkit_kafka.tests.messages.test_unknown_kafka_message.**UnknownKafkaMessageTestCase**(
Bases: *dhcpkit_kafka.tests.messages.test_kafka_message.KafkaMessageTestCase* (page 8)

parse_packet()

setUp()

test_validate_data()

test_validate_message_type()

Submodules

dhcpkit_kafka.message_registry module

The option registry

class dhcpkit_kafka.message_registry.**KafkaMessageRegistry**
Bases: dhcpkit.registry.Registry

Registry for DHCPKit IPv6 Options

entry_point = 'dhcpkit_kafka.messages'

get_name(*item: object*) → str
Get the name for the by_name mapping.

Parameters **item** – The item to determine the name of

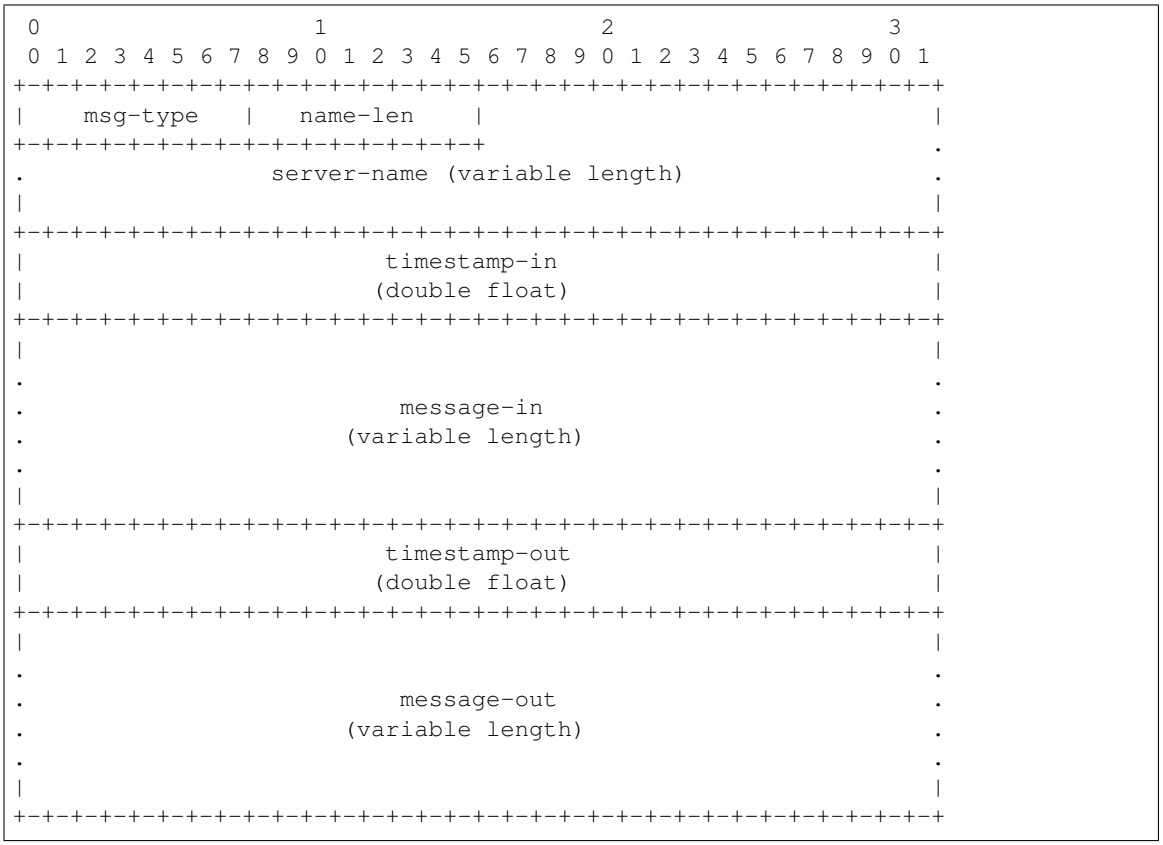
Returns The name to use as key in the mapping

dhcpkit_kafka.messages module

Messages that are sent over Kafka

class dhcpkit_kafka.messages.**DHCPKafkaMessage**(*server_name: str = ''*, *timestamp_in: typing.Union[int, float] = 0*, *message_in: dhcpkit.ipv6.messages.Message = None*, *timestamp_out: typing.Union[int, float] = 0*, *message_out: dhcpkit.ipv6.messages.Message = None*)
Bases: dhcpkit_kafka.messages.KafkaMessage (page 10)

A message for publishing DHCPv6 messages over Kafka for analysis.



load_from(*buffer: bytes*, *offset: int = 0*, *length: int = None*) → int
Load the internal state of this object from the given buffer.

Parameters

- **buffer** – The buffer to read data from
- **offset** – The offset in the buffer where to start reading
- **length** – The amount of data we are allowed to read from the buffer

Returns The number of bytes used from the buffer

message_type = 1

save() → typing.Union[bytes, bytearray]
Save the internal state of this object as a buffer.

Returns The buffer with the data from this element

validate()
Validate that the contents of this object

class dhcpkit_kafka.messages.**KafkaMessage**
Bases: dhcpkit.protocol_element.ProtocolElement

The base class for Kafka messages.

classmethod determine_class(*buffer: bytes*, *offset: int = 0*) → type
Return the appropriate subclass from the registry, or UnknownClientServerMessage if no subclass is registered.

Parameters

- **buffer** – The buffer to read data from
- **offset** – The offset in the buffer where to start reading

Returns The best known class for this message data

message_type = 0

class dhcpkit_kafka.messages.**UnknownKafkaMessage**(*message_type: int = 0*, *message_data: bytes = b''*)
Bases: *dhcpkit_kafka.messages.KafkaMessage* (page 10)

Container for raw message content for cases where we don't know how to decode the message.

load_from(*buffer: bytes*, *offset: int = 0*, *length: int = None*) → int
Load the internal state of this object from the given buffer. The buffer may contain more data after the structured element is parsed. This data is ignored.

Parameters

- **buffer** – The buffer to read data from
- **offset** – The offset in the buffer where to start reading
- **length** – The amount of data we are allowed to read from the buffer

Returns The number of bytes used from the buffer

save() → typing.Union[bytes, bytearray]
Save the internal state of this object as a buffer.

Returns The buffer with the data from this element

validate()
Validate that the contents of this object conform to protocol specs.

Changes per version

1.0.0 - 2016-12-08

Fixes

- Limit the queue sizes of pykafka to prevent memory problems when the Kafka cluster is unreachable

Changes for users

- Include documentation

Applicable copyright licences

DHCPKit License

Python Module Index

d

Index