intel.

Intel® Edge Insights for Industrial 2.5

API Reference Guide

May 2021

Intel Confidential

Contents

Revision History

Date	Revision	Description
May 2021	1.0	Initial release.

§

1.0 Appendix

1.1 EII Message Bus APIs

This section discusses the EII Message Bus APIs for C, Python* and GO* with its details for application developer.

1.1.1 EII Message Bus - C APIs

Source path: [EII_repo]/common/libs/EIIMessageBus/include/eii/msgbus

Message Bus Return Values (msgbusret.h)

Messaging return codes.

Enums

enum **msgbus_ret_t**

Return type for messaging actions.

Values:

MSG_SUCCESS = 0

MSG_ERR_PUB_FAILED = 1

MSG_ERR_SUB_FAILED = 2

MSG_ERR_RESP_FAILED = 3

MSG_ERR_RECV_FAILED = 4

MSG_ERR_RECV_EMPTY = 5

MSG_ERR_ALREADY_RECEIVED = 6

MSG_ERR_NO_SUCH_SERVICE = 7

MSG_ERR_SERVICE_ALREADY_EXIST = 8

MSG_ERR_BUS_CONTEXT_DESTROYED = 9

MSG_ERR_INIT_FAILED = 10

sMSG_ERR_NO_MEMORY = 11

MSG_ERR_ELEM_NOT_EXIST = 12

MSG_ERR_ELEM_ALREADY_EXISTS = 13

MSG_ERR_ELEM_BLOB_ALREADY_SET = 14

MSG_ERR_ELEM_BLOB_MALFORMED = 15

MSG_RECV_NO_MESSAGE = 16

MSG_ERR_SERVICE_INIT_FAILED = 17

MSG_ERR_REQ_FAILED = 18

MSG_ERR_EINTR = 19

MSG_ERR_MSG_SEND_FAILED = 20

MSG_ERR_DISCONNECTED = 21

MSG_ERR_AUTH_FAILED = 22

MSG_ERR_ELEM_OBJ = 23

MSG_ERR_ELEM_ARR = 24

MSG_ERR_UNKNOWN = 255

Configuration Interface (common/util/c/include/eii/utils/config.h)

EII configuration interface.

Typedefs

*typedefstruct*_config_value **config_value_t**

> Structure representing a configuration value.

Enums

*enum*config_value_type_t

> Valid configuration value types

intel.

Values:

CVT_INTEGER = 0

CVT_FLOATING = 1

CVT_STRING = 2

CVT_BOOLEAN = 3

CVT_OBJECT = 4

CVT_ARRAY = 5

CVT_NONE = 6

Functions

config_t ***config_new(void ****cfg**, void (****free_fn**)(void
*),** config_value_t ***(****get_config_value**)(*const* void *, *const* char *))**

Create a new configuration object.

Return

config_t, or NULL if an error occurs

Parameters

- **cfg** : – Configuration context
- **free_fn** : – Method to free the configuration context
- **get_config_value** : – Method to retrieve a key from the configuration

config_value_t ***config_get(***const* config_t ******config**, *const* char ****key**)

Get value from configuration object.

Note

Returns NULL if the value cannot be found in the configuration object.

Return

config_value_t

Parameters

- **config** : – Configuration object pointer

- **key** : – Key for the configuration value

---

void config_destroy(config_t ***config**)

Destroy the configuration object.

Parameters

- **config** : – Configuration to destroy

---

config_value_t ***config_value_object_get**(*const* config_value_t ***obj**, *const* **char** ***key**)

Retreive a configuration value from a configuration value object.

Note

The obj parameter must be a **CVT_OBJECT** type, NULL will be returned if it is not.

Return

config_value_t

Parameters

- **obj** : – Configuration value object
- **key** : – Key to retrieve from the configuration object

---

config_value_t ***config_value_array_get**(*const* config_value_t ***arr**, **int** *idx*)

Retreive a configuration value from a configuration array.

Note

The arr parameter mustb be of type **CVT_ARRAY** , otherwise NULL will be returned.

Return

config_value_t*

Parameters

- **arr** : – Array configuration value
- **idx** : – Index in the array to retrieve

---

size_t config_value_array_len(*const* config_value_t ***arr**)

Get the length of a configuration array.

intel.

> **Return**
>
> int
>
> > **Parameters**
> >
> > - **arr** : – Array configuration value

config_value_t *config_value_new_integer(int64_t value)

> Helper function to create a new config_value_t pointer to the given integer value.
>
> **Return**
>
> config_value_t*
>
> > **Parameters**
> >
> > - **value** : – Integer value

config_value_t *config_value_new_floating(double value)

> Helper to create a new config_value_t pointer to the given double value.
>
> **Return**
>
> config_value_t*
>
> > **Parameters**
> >
> > - **value** : – Floating point value

config_value_t *config_value_new_string(const char *value)

> Helper function to create a new config_value_t pointer to the given string value.
>
> **Return**
>
> config_value_t*
>
> > **Parameters**
> >
> > - **value** : – String value

config_value_t *config_value_new_boolean(bool value)

> Helper function to create a new config_value_t pointer to the given Boolean value.

Return

config_value_t*

Parameters

• **value** : – Boolean value

config_value_t *__config_value_new_object__(**void** **value***,** config_value_t **(*get)(__const__ **void** *,* __const__ **char** *),* **void** (**free_fn*)(**void** *))

Helper function to create a new config_value_t pointer to the given configuration object value.

Return

config_value_t*

Parameters

• **value** : – Object value
• **free_fn** : – Free method for the object value

config_value_t *__config_value_new_array__(**void** **array*,
size_t *length*, config_value_t **(*get)(__const__ **void** *, **int**),* **void** (**free_fn*)(**void** *))

Helper function to create a new config_value_t pointer to the given array value.

Return

config_value_t*

Parameters

• **array** : – Pointer to array context
• **length** : – Array length
• **get** : – Get method for getting an element in the array
• **free_fn** : – Function to free the array object

config_value_t *__config_value_new_none__()

Helper function to create a new config_value_t pointer to an empty configuration value.

Return

config_value_t*

---

void **config_value_destroy(**config_value_t ***value)**

Destroy a configuration value.

> **Parameters**

- **value** : – Configuration value to destroy

---

*struct***config_value_object_t**

#include <config.h>

Config value object representation. Includes method for freeing the object when the caller that obtained the object is finished with it.

Public Members

void *object

config_value_t ***(*get)(***const* **void *obj,** *const* **char *key)**

void (*free)(void *object)

---

*struct***config_value_array_t**

#include <config.h>

Config value array representation. Includes methods for getting elements at a given index and for freeing the array.

Public Members

void *array

size_t length

config_value_t ***(*get)(***const* **void *array, int idx)**

void (*free)(void *array)

---

*struct***_config_value**

#include <config.h>

Structure representing a configuration value.

Public Members

config_value_type_t**type**

int64_t integer

double floating

char *string

bool boolean

config_value_object_t ***object**

config_value_array_t ***array**

*union*_config_value**::[anonymous]body**

*struct***config_t**

#include <config.h>

Configuration object

Public Members

void *cfg

void (*free)(void *)

config_value_t ***(*get_config_value)(***const* void *, *const* char *)

Message Envelope API (**msg_envelope.h** **)**

Messaging envelope abstraction.

Enums

*enum***content_type_t**

Content types

Values:

CT_JSON = 0

CT_BLOB = 1

***enum*msg_envelope_data_type_t**

Message envelope value data types.

Values:

MSG_ENV_DT_INT = 0

MSG_ENV_DT_FLOATING = 1

MSG_ENV_DT_STRING = 2

MSG_ENV_DT_BOOLEAN = 3

MSG_ENV_DT_BLOB = 4

MSG_ENV_DT_OBJECT = 5

MSG_ENV_DT_ARRAY = 6

MSG_ENV_DT_NONE = 7

Functions

---

msg_envelope_t ***msgbus_msg_envelope_new(**content_type_t***ct*)**

Create a new msg_envelope_t to be sent over the message bus.

Return

msg_envelope_t, or NULL if an error occurs

Parameters

• **ct** : - Content type

---

msg_envelope_elem_body_t ***msgbus_msg_envelope_new_none()**

Create a new empty msg envelope element.

Return

msg_envelope_elem_body_t , or NULL if errors occur

msg_envelope_elem_body_t ***msgbus_msg_envelope_new_array()**

Create a new array element to be added to a message envelope.

> **Return**
>
> **msg_envelope_elem_body_t** , or NULL if errors occur

msg_envelope_elem_body_t ***msgbus_msg_envelope_new_object()**

Create a new nested object to be added to a message envelope, array, or other object.

> **Return**
>
> **msg_envelope_elem_body_t** , or NULL if errors occur

msg_envelope_elem_body_t ***msgbus_msg_envelope_new_string(*const* char *string)**

Helper function for creating a new message envelope element containing a string value.

> **Return**
>
> msg_envelope_body_t, or NULL if errors occur
>
> > **Parameters**
> >
> > - **string** : – String value to be placed in the envelope element

msg_envelope_elem_body_t ***msgbus_msg_envelope_new_integer(int64_t *integer*)**

Helper function for creating a new message envelope element containing an integer value.

> **Return**
>
> msg_envelope_body_t, or NULL if errors occur
>
> > **Parameters**
> >
> > - **integer** : – Integer value to be placed in the envelope element

msg_envelope_elem_body_t ***msgbus_msg_envelope_new_floating(double *floating*)**

Helper function for creating a new message envelope element containing a floating-point value.

Return

msg_envelope_body_t, or NULL if errors occur

Parameters

- **floating** : – Floating-point value to be placed in the envelope element

msg_envelope_elem_body_t ***msgbus_msg_envelope_new_bool(bool** *boolean***)**

Helper function for creating a new message envelope element containing a boolean value.

Return

msg_envelope_body_t, or NULL if errors occur

Parameters

- **boolean** : – Boolean value to be placed in the envelope element

msg_envelope_elem_body_t ***msgbus_msg_envelope_new_blob(char ****data***, size_t** *len***)**

Helper function for creating a new message envelope element containing a data blob.

Note

The enevelope element takes ownership of releasing the data.

Return

msg_envelope_body_t, or NULL if errors occur

Parameters

- **blob** : – Blob data to be placed in the envelope element
- **len** : – Size of the data blob

msgbus_ret_t**msgbus_msg_envelope_elem_object_put(**msg_envelope_elem_body_t **obj, co nst* **char ****key***,** msg_envelope_elem_body_t **value***)**

Put a new (key, value) pair into a message envelope nested object.

Return

msgbus_ret_t

Parameters

- **obj** : – Message envelope object element to add the (key, value) pair to
- **key** : – Key for the value
- **value** : – Value associated to the key

msg_envelope_elem_body_t ***msgbus_msg_envelope_elem_object_get(**msg_envelope_elem _body_t ***obj**, *const* char ***key)**

Get value from a message envelope nested object.

Return

msg_envelope_body_t , or NULL if errors occur

Parameters

- **obj** : – Message envelope object element to retrieve value from
- **key** : – Key of the value to retrieve

msgbus_ret_t**msgbus_msg_envelope_elem_object_remove(**msg_envelope_elem_body_t ***obj**, *const* char ***key)**

Remove (key, value) pair from the message envelope nested object.

Return

msgbus_ret_t

Parameters

- **obj** : – Message envelope object element to remove element from
- **key** : – Key to remove

msgbus_ret_t**msgbus_msg_envelope_elem_array_add(**msg_envelope_elem_body_t ***arr**, ms g_envelope_elem_body_t ***value)**

Add item to the msg envelope array element.

Return

msgbus_ret_t

Parameters

- **arr** : – Array to add the element to

- **value** : – Element to add

msg_envelope_elem_body_t ***msgbus_msg_envelope_elem_array_get_at(**msg_envelope_elem_body_t ***arr, int idx)**

Get item in the msg envelope array element.

Return

msg_envelope_elem_body_t, or NULL if an error occurs

Parameters

- **arr** : – Array to remove the element from
- **idx** : – Index of the item to get

msgbus_ret_t**msgbus_msg_envelope_elem_array_remove_at(**msg_envelope_elem_body_t ***arr, int idx)**

Remove item to the msg envelope array element.

Return

msgbus_ret_t

Parameters

- **arr** : – Array to remove the element from
- **value** : – Element to remove

void msgbus_msg_envelope_elem_destroy(msg_envelope_elem_body_t ***elem)**

Helper function to destroy a message envelope element.

Parameters

- **elem** : – Element to destroy

msgbus_ret_t**msgbus_msg_envelope_put(**msg_envelope_t ***env, const char *key,** msg_envelope_elem_body_t ***data)**

Add (key, value) pair to the message envelope.

Note

{If the message envelope is set to be a CT_BLOB , then it will act differently than a

message set to a different content type. For a blob the data can only be set once for the message and the key value will be ignored and the key BLOB will be used.

Additionally, the value body must be a blob as well.}

| **Return**

msgbus_ret_t

| **Parameters**

- env : – Message envelope
- key : – Key for the value
- data : – Value to be added

msgbus_ret_t**msgbus_msg_envelope_remove(**msg_envelope_t *****env**, *const* **char *****key**)**

Remove the (key, value) pair with the given key.

| **Return**

msgbus_ret_t

| **Parameters**

- env : – Message envelope
- key : – Key to remove

msgbus_ret_t**msgbus_msg_envelope_get(**msg_envelope_t *****env**, *const* **char *****key**, msg_envelope_elem_body_t *****data**)**

Get the value for the given key in the message bus.

| **Note**

{If the content type is CT_BLOB , then use "BLOB" as the key to retrieve the blob

data.}

| **Return**

msgbus_ret_t

| **Parameters**

- [in] env : – Message envelope

- **[in] key** : – Key for the element to find
- **[out] data** : – Data for the key

---

int msgbus_msg_envelope_serialize(msg_envelope_t ***env,** msg_envelope_serialized_part_t *****parts)**

Serialize the data in the envelope into the given message parts buffer based on the content type given when msgbus_msg_envelope_new() was called.

| **Return**

Number of serialized message parts

| **Parameters**

- **[in] env** : – Message envelope
- **[out] parts** : – Serialized parts

---

msgbus_ret_t**msgbus_msg_envelope_deserialize(**content_type_t**ct,** msg_envelope_serialized_part_t *****parts,** **int** *num_parts,* *const* **char ****name,** msg_envelope_t *****env)**

Deserialize the given data into a msg_envelope_t.

If the content type is set to CT_BLOB, then this method assumes that there will only be one serialized message part.

Additionally, if the content type is CT_JSON, then this method assumes that there will be either one or two message parts. The first part MUST always be a JSON string. If a second part is present it MUST be a binary blob of data.

| **Return**

msgbus_ret_t

| **Parameters**

- **[in] ct** : – Message content type
- **[in] parts** : – Serialized parts to deserailize
- **[in] num_parts** : – Number of message parts
- **[in] name** : – Topic name
- **[out] env** : – Output message envelope

msgbus_ret_t msgbus_msg_envelope_serialize_parts_new(int *num_parts,* msg_envelope_serialized_part_t **parts*)

Create a new list of serialized message parts.

Return

msgbus_ret_t

Parameters

- **[in] num_parts** : – Number of serialized message parts
- **[out] parts** : – Serialzied parts

void msgbus_msg_envelope_serialize_destroy(msg_envelope_serialized_part_t **parts,* int *num_parts*)

Destroy the serialized parts of a message

Return

msgbus_ret_t

Parameters

- **parts** : – Serialized parts
- **num_parts** : – Number of serialized parts

void msgbus_msg_envelope_destroy(msg_envelope_t **msg*)

Delete and clean up a message envelope structure.

Parameters

- **msg** : – Message envelope to delete

owned_blob_t *owned_blob_new(void **ptr,* void (**free_fn*)(void *), *const* char **data,* size_t *len,*)

Helper for initializing owned blob pointer.

Note

Assumes data is owned

owned_blob_t *owned_blob_copy(owned_blob_t **to_copy*)

Copy a shared blob, except assume the underlying data is NOT owned by the copy of the blob.

void owned_blob_destroy(owned_blob_t ***shared)**

Helper for destroying owned blob pointer.

***struct*owned_blob_t**

#include <msg_envelope.h>

Shared object structure for message bus data blobs.

Public Members

void *ptr

void (*free)(void *)

bool owned

size_t len

const **char *bytes**

***struct*msg_envelope_blob_t**

#include <msg_envelope.h>

Message envelope blob data type.

Public Members

owned_blob_t ***shared**

uint64_t len

const **char *data**

***struct*msg_envelope_elem_body_t**

#include <msg_envelope.h>

Message envelope element body type.

Public Members

msg_envelope_data_type_t**type**

int64_t integer

double floating

char *string

bool boolean

msg_envelope_blob_t ***blob**

hashmap_t *object

linkedlist_t *array

*union*msg_envelope_elem_body_t::**[anonymous]body**

*struct*msg_envelope_t

#include <msg_envelope.h>

Message envelope around a given message that is to be sent or received over the message bus.

Public Members

char *name

char *correlation_id

content_type_t**content_type**

hashmap_t *map

msg_envelope_elem_body_t ***blob**

*struct*msg_envelope_serialized_part_t

#include <msg_envelope.h>

Part of a serialized message envelope.

Public Members

owned_blob_t ***shared**

size_t len

const **char *bytes**

Message Bus API (**msgbus.h** **)**

Messaging abstraction interface.

Typedefs

typedef **void *publisher_ctx_t**

Publisher context

Functions

void *msgbus_initialize(config_t ******config***)**

Initialize the message bus.

Note

{The message bus context takes ownership of the **config_t** object at this point and the caller does not have to free the config object.}

Return

Message bus context, or NULL

Parameters

- **config** : – Configuration object

void msgbus_destroy(void **ctx***)**

Delete and clean up the message bus.

msgbus_ret_t**msgbus_publisher_new(void ****ctx***,** *const* **char ****topic***,** publisher_ctx_t *******pub_ctx***)**

Create a new publisher context object.

Note

{The **get_config_value()** method for the configuration will be called to retrieve values needed for the underlying protocol to initialize the context for publishing.}

Return

msgbus_ret_t

Parameters

- **[in] ctx** : – Message bus context
- **[out] pub_ctx** : – Publisher context

msgbus_ret_t**msgbus_publisher_publish(void** **ctx,* publisher_ctx_t ***pub_ctx,** msg_envelope_t ***message)**

Publish a message on the message bus.

Return

msgbus_ret_t

Parameters

- **ctx** : – Message bus context
- **pub_ctx** : – Publisher context
- **message** : – Messsage object to publish

void **msgbus_publisher_destroy(void** **ctx,* publisher_ctx_t ***pub_ctx)**

Destroy publisher

Parameters

- **ctx** : – Message bus context
- **pub_ctx** : – Publisher context

msgbus_ret_t**msgbus_subscriber_new(void** **ctx, const* char **topic,* user_data_t **user_data,* recv_ctx_t ****subscriber)**

Subscribe to the given topic.

Return

msgbus_ret_t

Parameters

- **[in] ctx** : – Message bus context
- **[in] topic** : – Subscription topic string
- **[in] user_data** : – User data attached to the receive context
- **[out] subscriber** : – Resulting subscription context

void msgbus_recv_ctx_destroy(void *ctx, recv_ctx_t ***recv_ctx)**

Delete and clean up a service, request, or subscriber context.

Parameters

- **ctx** : – Message bus context
- **recv_ctx** : – Receive context

msgbus_ret_t**msgbus_request(void *ctx,** recv_ctx_t ***service_ctx,** msg_envelope_t ***message)**

Issue a request over the message bus.

Return

msgbus_ret_t

Parameters

- **ctx** : Message bus context
- **service_ctx** : Service context
- **message** : Request

msgbus_ret_t**msgbus_response(void *ctx,** recv_ctx_t ***service_ctx,** msg_envelope_t ***message)**

Respond to the given request.

Return

msgbus_ret_t

Parameters

- **ctx** : – Message bus context
- **service_ctx** : – Service context
- **message** : – Response message

msgbus_ret_t **msgbus_service_get**(void *ctx, const char *service_name, void *user_data, recv_ctx_t **service_ctx)

Create a context to send requests to a service.

Parameters

- **[in] ctx** : – Message bus context
- **[in] service_name** : – Name of the service
- **[in] user_data** : – User data
- **[out] service_ctx** : – Service context
- **msgbus_ret_t** :

msgbus_ret_t **msgbus_service_new**(void *ctx, const char *service_name, void *user_data, recv_ctx_t **service_ctx)

Create context to receive requests over the message bus.

Return

msgbus_ret_t

Parameters

- **[in] ctx** : – Message bus context
- **[in] service_name** : – Name of the service
- **[in] user_data** : – User data
- **[out] service_ctx** : – Service context

msgbus_ret_t **msgbus_recv_wait**(void *ctx, recv_ctx_t *recv_ctx, msg_envelope_t **message)

Receive a message over the message bus using the given receiving context.

Note

{If a response has already been received for a given request, then a
MSG_ERR_ALREADY_RECEIVED will be returned.}

Return

msgbus_ret_t

Parameters

- **[in] ctx** : – Message bus context
- **[in] recv_ctx** : – Context to use when receiving a message
- **[out] message** : – Message received (if one exists)

msgbus_ret_t**msgbus_recv_timedwait**(void *ctx*, recv_ctx_t ***recv_ctx**,
int *timeout*, msg_envelope_t ***message**)

Receive a message over the message bus, if no message is available wait for the given amount of time for a message to arrive.

Return

msgbus_ret_t, MSG_RECV_NO_MESSAGE if no message received

Parameters

- **[in] ctx** : – Message bus context
- **[in] recv_ctx** : – Receive context
- **[in] timeout** : – Timeout for waiting to receive a message in microseconds
- **[out] message** : – Received message, NULL if timedout

msgbus_ret_t**msgbus_recv_nowait**(void
ctx, recv_ctx_t ***recv_ctx**, msg_envelope_t ***message**)

Receive a message if available, immediately return if there are no messages available.

Return

msgbus_ret_t, MSG_RECV_NO_MESSAGE if no message is available

Parameters

- **[in] ctx** : – Message bus context
- **[in] recv_ctx** : – Receive context
- **[out] message** : – Received message, NULL if timedout

*struct*user_data_t

#include <msgbus.h>

Request user data type

Public Members

void *data

void (*free)(void *data)

*struct*recv_ctx_t

#include <msgbus.h>

Receive context structure used for service, subscription, and request contexts.

Public Members

void *ctx

user_data_t **user_data**

*struct*recv_ctx_set_t

#include <msgbus.h>

Set of receive context to be used with **msgbus_recv_ready_poll()** method.

Public Members

int size

int max_size

bool *tbl_ready

recv_ctx_t **tbl_ctxs**

Protocol Interface API (protocol.h)

Messaging protocol interface.

*struct*protocol_t

#include <protocol.h>

Underlying protocol interface for messaging through the message bus.

Public Members

void *proto_ctx

config_t *config

void (*destroy)(void *ctx)

msgbus_ret_t (*publisher_new)(void *ctx, *const* char *topic, void **pub_ctx)

msgbus_ret_t (*publisher_publish)(void *ctx, void *pub_ctx, msg_envelope_t *msg)

void (*publisher_destroy)(void *ctx, void *pub_ctx)

msgbus_ret_t (*subscriber_new)(void *ctx, *const* char *topic, void **subscriber)

void (*recv_ctx_destroy)(void *ctx, void *recv_ctx)

msgbus_ret_t (*request)(void *ctx, void *service_ctx, msg_envelope_t *message)

msgbus_ret_t (*response)(void *ctx, void *service_ctx, msg_envelope_t *message)

msgbus_ret_t (*service_get)(void *ctx, *const* char *service_name, void **service_ctx)

msgbus_ret_t (*service_new)(void *ctx, *const* char *service_name, void **service_ctx)

msgbus_ret_t (*recv_wait)(void *ctx, void *recv_ctx, msg_envelope_t **message)

msgbus_ret_t (*recv_timedwait)(void *ctx, void *recv_ctx, int timeout, msg_envelope_t **message)

msgbus_ret_t (*recv_nowait)(void *ctx, void *recv_ctx, msg_envelope_t **message)

intel.

1.1.2 EII Message Bus - Python* APIs

Source path: [EII_repo]/common/libs/EIIMessageBus/python/eii

MsgbusContext (**eii.msgbus.MsgbusContext**)

Class **eii.msgbus.MsgbusContext(***config***)**

EII Message Bus context object

get_service(*service_name***)**

Create a new service context for issuing requests to a service.

Note that this method will expect to find the configuration attributes needed to communicate with the specified service in the configuration given to the constructor.

Parameters

service_name (*str*) – Name of the service

Returns

Service object

Return type

Service

new_publisher(*topic***)**

Create a new publisher object.

Parameters

topic (*str*) – Publisher's topic

Returns

Publisher object

Return type

[Publisher](#)

new_service(*service_name*)

Create a new service context for receiving requests.

Note that this method will expect to find the configuration attributes needed to communicate with the specified service in the configuration given to the constructor.

Parameters

service_name (*str*) – Name of the service

Returns

Service object

Return type

[Service](#)

new_subscriber(*topic*)

Create a new subscriber object.

Parameters

topic (*str*) – Topic to subscribe to

Returns

Subscriber object

Return type

Subscriber

Publisher (eii.msgbus.Publisher)

Class **eii.msgbus.Publisher**

EII Message Bus Publisher object

close()

Close the publisher.

This MUST be called before the program exists. If it is not your program may hang.

Note that this method is not thread-safe.

publish(*message*)

Publish message on the publisher object.

The message object passed to this method can be either a Python bytes object or a Python dictionary. When a bytes object is given to be published, then this method will construct a message envelope for a blob. If a dictionary is given, then a JSON message envelope will be constructed and published.

Parameters

message – Message to publish

Type

bytes or dict

ReceiveContext (eii.msgbus.ReceiveContext)

Class **eii.msgbus.ReceiveContext**

EII Message Bus receive context wrapper object

close()

Close the receive context.

This MUST be called before the program exists. If it is not your program may hang.

Note that this method is not thread-safe.

recv(*blocking=True*, *timeout=-1*)

Receive a message on the message bus for the given receive context. Note that the receive context can be a Subscriber or a Service object.

intel.

Additionally, if the timeout is set to -1, then this method will operate based on whether blocking is set to True or False (i.e. block or do not block). However, it the timeout is set to anything > -1, then this method will ignore whether blocking is set and use operate on a timeout.

Parameters

- **blocking** (*bool*) – (Optional) Block until message received, or return immediately.
- **timeout** (*int*) – (Optional) Timeout in milliseconds to receive a message

Returns

Received message

Return type

dict or bytes

Service (eii.msgbus.Service)

Class **eii.msgbus.Service**

EII Message Bus service wrapper object for receiving requests

response(*response*)

Issue a response over the message bus.

Parameters

resp – Response data

Type

bytes or dict

ServiceRequester (eii.msgbus.ServiceRequester)

Class **eii.msgbus.ServiceRequester**

EII Message Bus service wrapper object to issue requests

request(*request*)

Issue a request to the service.

Parameters

request – Request to issue to the service

Type

bytes or dict

MsgEnvelope (eii.msgbus.MsgEnvelope)

Class **eii.msgbus.MsgEnvelope**(*name=None, kv=None, blob=None*)

EII Message Envelope

get_blob()

Get the blob data in the message envelope.

Returns

Blob data in the message if any exists

Return type

Union[None, bytes]

get_meta_data()

Get the meta-data in the message envelope if any exists.

Returns

Meta-data in the message envelope

Return type

Union[None, dict]

get_name()

Get the topic string or service name in the message envelope.

Note

This will only be assigned if the MsgEnvelope was received over the message bus.

Returns

Topic string or service name, will be None if not set

Return type

Union[None, str]

1.1.3 EII Message Bus - GO APIs

Source path: [EII_repo]/common/libs/EIIMessageBus/go/EIIMessageBus/eiimsgbus

---

MsgbusContext

NewMsgbusClient(config map[string]interface{}) (*MsgbusClient, error)

Initialize a new message bus context.

Return

1. MsgbusClient – Returns handle of MsgbusClient
2. error – Returns an error if any failure in creating msgbusClient object

Parameters

- **config** : – map[string]interface{}   Configuration object

---

(ctx *MsgbusContext) NewPublisher(topic string) (*Publisher, error)

Create a new publisher on the message bus context.

Return

1. Publisher – Publisher object
2. error – Returns an error if any failure in creating publisher

Parameters

- **topic** : – string   Publisher's topic

(ctx *MsgbusContext) NewSubscriber(topic string) (*Subscriber, error)

Create a new subscriber for the specified topic and Starts goroutine(non-blocking) to receive publications.

Return

1. Subscriber: Created Subscriber structure

```
type Subscriber struct {
      MessageChannel chan MsgEnvelope
      ErrorChannel   chan error
      quitChannel    chan interface{}
      closed         bool
 }
MessageChannel – received message on the message bus
ErrorChannel  – error msg if any failure in subscription
```

2. error – Returns an error if any failure in creating subscriber object

Parameters

- **topic** : – string   Subscriber's topic

(ctx *MsgbusClient) NewService(serviceName string) (*Service, error)

Create a new service to receive requests and send responses over.

Return

1. Service– Service object
2. error – Returns an error object if any failure in creating service

Parameters

- **serviceName** – string   Name of the service
- **topic** : – string   Subscriber's topic

(ctx *MsgbusContext) GetService(serviceName string) (*ServiceRequester, error)

Gets the service requester handle from the service name.

Return

1. ServiceRequester – ServiceRequester object
2. error – Returns an error object if any failure in creating service

Parameters

- **serviceName** – string   Name of the service

Publisher

(pub *Publisher) Publish(msg interface{})

To publish message on the publisher object.

Return

1. None

Parameters

- **msg** – interface{}   Message to publish

(pub *Publisher) Close() error

To close the publisher. This MUST be called before the program exists. If it is not, the program may behave unexpectedly.

Return

1. None

Parameters

- **None**

Subscriber

(ctx *MsgbusContext) newSubscriber(topic string) (*Subscriber, error)

Create a new subscriber for the specified topic.

Return

1. Subscriber – Subscriber object
2. error – Returns an error if any failure in creating subscriber object

Parameters

- **topic** – string   Subscriber's topic

(pub *Subscriber) Close() error

Close the subscriber. This MUST be called before the program exists. If it is not, the program may behave unexpectedly.

Return

1. None

Parameters

- **None**

Service

(service *Service) ReceiveRequest(timeout int) (*MsgEnvelope, error)

Receive a request issued to the service.

Return

1. MsgEnvelope – MsgEnvelope object
2. error – Returns an error object if any failure

Parameters

- **timeout** – int   Determines how the receive call will function. If the timeout is less than 0, then it will block for ever. If it is set 0, then it will return immediately. If the caller wishes there to be a timeout, then the timeout should be specified in milliseconds. For the no wait and timeout modes, if no message is received then both return values will be nil.

(service *Service) Response(response interface{}) error

Send a response to a request received by the service.

Return

1. None

Parameters

- **response** – interface{}   Response data

(service *Service) Close()

Close the service.

Return

1. None

Parameters

- **None**

Requester

(service *ServiceRequester) ReceiveResponse(timeout int) (*MsgEnvelope, error)

Receive a response to a previously sent request.

Return

1. MsgEnvelope – MsgEnvelope object
2. error – Returns an error object if any failure

Parameters

- **timeout** – int   Determines how the receive call will function. If the timeout is less than 0, then it will block for ever. If it is set 0, then it will return immediately. If the caller wishes there to be a timeout, then the timeout should be specified in milliseconds. For the no wait and timeout modes, if no message is received then both return values will be nil.

(service *ServiceRequester) Request(request interface{})

Send a response to a request received by the service.

Return

1. None

Parameters

- **response** – interface{}   Response data

(service *ServiceRequester) Close()

Close the Requester.

Return

1. None

Parameters

Intel® Edge Insights for Industrial 2.5
May 2021   API Reference Guide
Document Number: 634811-1.0   Intel Confidential   39

- **None**

1.2 EII ConfigMgr APIs

This section show the EII ConfigManager APIs for Python*, GO and C with its details for application developer.

1.2.1 Python* APIs

Source path: [EII_repo]/common/libs/ConfigMgr/python/cfgmgr

config_mgr contains the apis of the main ConfigMgr python instance

---

def get_app_config(self):

Gets AppCfg object respective applications config

Parameters:	None
Return value:	Return object of class AppCfg
Raises:	None

---

def get_watch_obj(self):

This function to Fetch the object to call watch APIs

Return value:	obj **:** Watch class object
Raises:	None

---

def get_app_name(self):

This function is to get the AppName for any application

Return value:	str : App name
Raises:	None

---

def get_publisher_by_name(self, name):

To fetch a publisher interface using its name

Parameters:	name : Name of the publisher : string
Return value:	obj: Publisher class object
Raises:	None

---

def get_publisher_by_index(self, index):

intel.

To fetch a publisher interface using its index

Parameters:	index: Index of the publisher : int
Return value:	obj: Publisher class object
Raises:	None

def get_subscriber_by_name(self, name):

To fetch a subscriber interface using its name

Parameters:	name: name of the subscriber : string
Return value:	obj: Subscriber class object
Raises:	None

def get_subscriber_by_index(self, index):

To fetch a subscriber interface using it's index

Parameters:	index: Index of the subscriber : int
Return value:	obj: Subscriber class object
Raises:	None

def get_server_by_name(self, name):

To fetch a server interface using its name

Parameters:	name: name of the server : string
Return value:	obj: Server class object
Raises:	None

def get_server_by_index(self, index):

To fetch a server interface using its index

Parameters:	index : Index of the server : string
Return value:	obj: Server class object
Raises:	None

def get_client_by_name(self, name):

To fetch a client interface using its name

Parameters:	name : Name of the client : string
Return value:	obj: Client class object

intel

Raises:	None

---

def get_client_by_index(self, index):

To fetch a client interface using it's index

Parameters:	index : Index of the client : int
Return value:	obj: Client class object
Raises:	None

---

def get_num_publishers(self):

Get total number of publishers from publisher's interface

Return value:	int: number of publishers in interface
Raises:	None

---

def get_num_subscribers(self):

Get total number of subscribers from subscriber's interface

Return value:	int: number of subscribers in interface
Raises:	None

---

def get_num_servers(self):

Get total number of servers from server's interface

Return value:	int: number of servers in interface
Raises:	None

---

def get_num_clients(self):

Get total number of servers from client's interface

Return value:	int: number of clients in interface
Raises:	None

Publisher python class contains publisher related APIs

def get_msgbus_config(self):

Constructs message bus config for Publisher

Return value:	dict : Messagebus config

Raises: None

---

def get_interface_value(self, key):

To get particular interface value from Publisher interface config

Parameters: key : Key on which interface value will be extracted: string

Return value: string : Interface value

Raises: None

---

def get_endpoint(self):

To get endpoint for particular publisher from its interface config

Return value: string : Endpoint config

Raises: None

---

def get_topics(self):

To get topics from publisher interface config on which data will be‹→published

Return value: List : List of topics

Raises: None

---

def get_allowed_clients(self):

To get the names of the clients allowed to get publishers data

Return value: List : List of clients

Raises: None

---

def set_topics(self, topics_list):

To set new topics for publisher in publishers interface config

Return value: int : Whether topic is set – 0 is success

Raises: None

Subscriber python class contains publisher related APIs

---

def get_msgbus_config(self):

Constructs message bus config for Subscriber

Return value: dict : Messagebus config

Raises: None

---

def get_interface_value(self, key):

To get particular interface value from Subscriber interface config

Parameters: key : Key on which interface value will be extracted: string

Return value: string : Interface value

Raises: None

---

def get_endpoint(self)

To get endpoint for particular subscriber from its interface config

Return value: string : Endpoint config

Raises: None

---

def get_topics(self):

To gets topics from subscriber interface config on which subscriber↵→receives data

Return value: List : List of topics

Raises: None

---

def set_topics(self, topics_list):

To sets new topics for subscriber in subscribers interface config

Return value: int : whether topic is set – *0 is success*

Raises: None

Server python class contains publisher related APIs

def get_msgbus_config(self):

Constructs message bus config for Server

Return value: dict : Messagebus config

Raises: None

def get_interface_value(self, key):

To get particular interface value from Server interface config

Parameters: key : Key on which interface value will be extracted: string

Return value: string : Interface value

Raises: None

def get_endpoint(self):

To get endpoint for particular server from its interface config

Return value: string : Endpoint config

Raises: None

def get_allowed_clients(self):

To get the names of the clients allowed to connect to server

Return value: List : List of clients

Raises: None

Client python class contains publisher related APIs

def get_msgbus_config(self):

Constructs message bus config for Client

Return value: dict : Messagebus config

Raises: None

def get_interface_value(self, key):

To get particular interface value from Client interface config

Parameters: key : Key on which interface value will be extracted: string

Return value: string : Interface value

Raises: None

def get_endpoint(self):

To get endpoint for particular client from its interface config

Return value:	string : Endpoint config
Raises:	None

Further details of Python* APIs, refer to
[WORK_DIR]/IEdgeInsights/common/libs/ConfigMgr/python directory

To refer to Python examples on the EII ConfigMgr client wrapper, follow:
[WORK_DIR]/IEdgeInsights/common/libs/ConfigMgr/python/example

1.2.2   Go* APIs

config_mgr contains the apis of the main ConfigMgr python instance

Source path: [EII_repo]/common/libs/ConfigMgr/go/ConfigMgr/eiiconfigmgr

---

func ConfigManager() (*ConfigMgr, **error)**

Initialize a new config manager context

Return value:	ConfigMgr : ConfigMgr object – ConfigManager instance
	error –Error on failure
Raises:	None

---

func (ctx *ConfigMgr) GetAppConfig() (**map**[**string**]**interface**{}, **error)**

Get Applications config.

Return value:	map[string] : interface - map-string interface
	error - Error on failure
Raises:	None

---

func (ctx *ConfigMgr) GetAppName() (string, error)

Get Applications name.

Return value:	App name : string – Application name
	error **–** Error on failure
Raises:	None

---

func (ctx *ConfigMgr) IsDevMode() (bool, error)

To check if application running in dev_mode or prod_mode

Return value:	bool value : bool – true if dev_mode, false if prod_mode

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetNumSubscribers() (int, error)

Get number of subscribers in Subscriber array in interface

Return value: num_of_subscribers : int - Number of subscribers

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetNumServers () (int, error)

Get number of servers in Server array in interface

Return value: num_of_client : int - Number of servers

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetNumClients() (int, error)

Get number of clients in Client array in interface

Return value: num_of_client : int - Number of clients

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetPublisherByName(name string) (*PublisherCfg, error)

Get the respective publisher based on the name if publisher array has multiple ↪→
endpoints in interface

Return value: PublisherCfg : PublisherCfg obj - PublisherCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetPublisherByIndex(index int) (*PublisherCfg, error)

Get the respective publisher based on the index if publisher array has multiple ↪→
endpoints in interface

Return value: PublisherCfg : PublisherCfg obj – PublisherCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetSubscriberByName(name string) (*SubscriberCfg, error)

Get the respective subscriber based on the name if subscriber array has multiple↵→ endpoints in interface

Return value: SubscriberCfg: SubscriberCfg obj- SubscriberCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetSubscriberByIndex(index int) (*SubscriberCfg, error)

Get the respective subscriber based on the index if subscriber array has multiple↵→ endpoints in interface

Return value: SubscriberCfg: SubscriberCfg obj- SubscriberCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetServerByName(name string) (*ServerCfg, error)

Get the respective server based on the name if server array has multiple endpoints↵→in interface

Return value: ServerCfg: ServerCfg obj- ServerCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetServerByIndex(index int) (*ServerCfg, error)

Get the respective server based on the index if server array has multiple endpoints↵→in interface

Return value: ServerCfg: ServerCfg obj- ServerCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetServerByIndex(index int) (*ServerCfg, error)

Get the respective server based on the index if server array has multiple endpoints↵→in interface

Return value: ServerCfg: ServerCfg obj- ServerCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetClientByName(name string) (*ClientCfg, error)

Get the respective client based on the name if client array has multiple endpoints‹→in interface interface

Return value: ClientCfg: ClientCfg obj- ClientCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetClientByIndex(index int) (*ClientCfg, error)

Get the respective client based on the index if client array has multiple endpoints‹→in interface

Return value: ClientCfg: ClientCfg obj- ClientCfg instance

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) GetAppConfig() (appConfig map[string]interface{}, error)

Gets value from respective application's configinterface

Return value: appConfig map[string]interface{}

error **–** Error on failure

Raises: None

---

func (ctx *ConfigMgr) Destroy()

To delete ConfigMgr contextinterface

Return value: None

Raises: None

---

func (cfg ConfigValue) GetInteger() (integer, error)

GetInteger gets integer value from the value received from GetInterfaceValue

Return value: integer value: integer - returns integer value

error **–** Error on failure

Raises: None

intel.

func (cfg ConfigValue) GetFloat() (float, error)

GetFloat gets float value from the value received from GetInterfaceValue

Return value: float value: float- returns float value

error **–** Error on failure

Raises: None

func (cfg ConfigValue) GetString() (string, error)

GetString gets string value from the value received from GetInterfaceValue

Return value: string value : string- returns string value

error **–** Error on failure

Raises: None

func (cfg ConfigValue) GetBool() (boolean, error)

GetBool gets boolean value from the value received from GetInterfaceValue

Return value: bool value: bool - returns bool value

error **–** Error on failure

Raises: None

func (cfg ConfigValue) GetJson() (object, error)

GetJson gets json value from the value received from GetInterfaceValue

Return value: json value: json - returns json value

error **–** Error on failure

Raises: None

func (cfg ConfigValue) GetArray() (array, error)

GetArray gets array value from the value received from GetInterfaceValue

Return value: array value: array - returns array value

error **–** Error on failure

Raises: None

Publisher GO interface contains publisher related APIs

func (pubctx *PublisherCfg) GetEndPoints() (string, error)

GetEndPoints for application to fetch Endpoint associated with message bus config

Return value: string - Endpoints value in string

error **–** Error on failure

Raises: None

func (pubctx *PublisherCfg) GetTopics() ([]string, error)

GetTopics gets topics from publisher interface config on which data will be↵published

Return value: topics : string array - array of topics

error **–** Error on failure

Raises: None

func (pubctx *PublisherCfg) GetAllowedClients() ([]string, error)

GetAllowedClients gets the names of the clients allowed to get publishers data

Return value: allowed_clients : string array - array of allowed clients

error **–** Error on failure

Raises: None

func (pubctx *PublisherCfg) GetMsgbusConfig() (map[string]interface{}, error)

GetMsgbusConfig to fetch client msgbus config for application to communicate over↵EII message bus

Parameters :   map[string]interface{}

Return value: bool value : bool - true if success, false on failure

Raises: None

func (pubctx *PublisherCfg) SetTopics(topics []string) bool

SetTopics sets new topic for publisher in publishers interface config

Parameters :   topics : string array - array of topics that needs to be set

Return value: bool value : bool - true if success, false on failure

Raises: None

func (pubctx *PublisherCfg) GetInterfaceValue(key string) (*ConfigValue, error)

intel.

GetInterfaceValue fetch interface value for application to communicate over EII↵→ message bus

Parameters :   key: string - Key on which interface value is extracted

Return value: Config value : ConfigValue object - Interface value

error **–** Error on failure

Raises: None

---

func (pubctx *PublisherCfg) Destroy()

delete publisher context

Return value: None

Raises: None

Subscriber GO interface contains subscriber related APIs

---

func (subctx *SubscriberCfg) GetEndPoints() (string, error)

GetEndPoints for application to fetch Endpoint associated with message bus config

Return value: string - Endpoints value in string

error **–** Error on failure

Raises: None

---

func (subctx *SubscriberCfg) GetTopics() ([]**string**, **error**)

GetTopics gets topics from subscriber interface config on which subscriber↵→receives data

Return value: topics : string array- array of topics

error **–** Error on failure

Raises: None

---

func (subctx *SubscriberCfg) GetMsgbusConfig() (**map[string]interface{}**, **error**)

GetMsgbusConfig to fetch client msgbus config for application to communicate over↵→EII message bus

Return value: map[string]interface{}

error **–** Error on failure

Raises: None

func (subctx *SubscriberCfg) SetTopics(topics []string) bool

> SetTopics sets new topic for subscriber in subscribers interface config
>
> **Parameters :**   topics : string array - array of topics that needs to be set
>
> **Return value:**   bool value : bool– true if success, false on failure
>
> **Raises:**   None

func (subctx *SubscriberCfg) GetInterfaceValue(key string) (*ConfigValue, error)

> GetInterfaceValue fetch interface value for application to communicate over EII↵→
> message bus
>
> **Parameters :**   key: string – Key on which interface value is extracted
>
> **Return value:**   Config value : ConfigValue object – Interface value
>
> error **–** Error on failure
>
> **Raises:**   None

func (subctx *SubscriberCfg) Destroy()

> To delete subscriber context
>
> **Return value:**   None
>
> **Raises:**   None

Server GO interface contains server related APIs

func (serverctx *ServerCfg) GetEndPoints() (string, error)

> GetEndPoints for application to fetch Endpoint associated with message bus config
>
> **Return value:**   string: float– Endpoints value in string
>
> error **–** Error on failure
>
> **Raises:**   None

func (serverctx *ServerCfg) GetAllowedClients() ([]string, error)

> GetAllowedClients gets the names of the clients allowed to connect to server
>
> **Return value:**   allowed_clients : string array – array of allowed clients
>
> error **–** Error on failure
>
> **Raises:**   None

func (serverctx *ServerCfg) GetInterfaceValue(key string) (*ConfigValue, error)

GetInterfaceValue fetch interface value for application to communicate over EII↵→ message bus

Parameters :   key: string – Key on which interface value is extracted

Return value: Config value : ConfigValue object – Interface value

error **–** Error on failure

Raises: None

func (serverctx *ServerCfg) Destroy()

To delete server context

Return value: None

Raises: None

Client GO interface contains client related APIs

func (clientctx *ClientCfg) GetEndPoints() (string, error)

GetEndPoints for application to fetch Endpoint associated with message bus config

Return value: string - Endpoints value in string

error **–** Error on failure

Raises: None

func (clientctx *ClientCfg) GetMsgbusConfig() (map[string]interface{}, error)

GetMsgbusConfig to fetch client msgbus config for application to communicate over↵→EII message bus

Return value: map[string]interface{}

error **–** Error on failure

Raises: None

func (clientctx *ClientCfg) GetInterfaceValue(key string) (*ConfigValue, error)

GetInterfaceValue fetch interface value for application to communicate over EII↵→ message bus

Parameters :   key: string – Key on which interface value is extracted

Return value: Config value : ConfigValue object – Interface value

error **–** Error on failure

Raises: None

---

func (clientctx *ClientCfg) Destroy()

To delete client context

Return value: None

Raises: None

Further details of Go* APIs, refer to
[WORK_DIR]/IEdgeInsights/common/libs/ConfigMgr/go/**ConfigMgr** directory

To refer to Go examples on the EII ConfigMgr client wrapper, follow:
[WORK_DIR]/IEdgeInsights/common/libs/ConfigMgr/go/ConfigMgr/example

1.2.3   C APIs

1. **ConfigMgr**

This API contains the APIs of the main ConfigMgr instances

Source path: [EII_repo]/common/libs/ConfigMgr/cpp

AppCfg* getAppConfig();

This function is to get the AppCgg object

AppCfg* = getAppConfig()

Return value: AppCfg*   – AppCfg class object

Raises: None

---

int getNumPublishers();

This function to get total number of publishers from publisher's interface.

Int value = getNumPublishers()

Return value: int – number of subscriber interfaces

Raises: None

---

int getNumSubscribers();

This function to get total number of subscribers from subscriber's interface

Int value = getNumSubscribers()

Return value: int – number of subscriber interfaces

Raises: None

---

int getNumServers();

This function to get total number of servers from server's interface

Int value = getNumServers()

Return value: int – number of subscriber interfaces

Raises: None

---

int getNumClients();

This function to get total number of clients from server's interface

Int value = getNumClients()

Return value: int – number of subscriber interfaces

Raises: None

---

bool isDevMode();

This function to check if application is running in dev or prod mode

bool value = isDevMode()

Return value: bool– True if dev mode & false if prod mode

Raises: None

---

std::string getAppName();

This function to get the AppName for any application

std::string = getNumServers()

Return value: std::string – AppName string

Raises: None

ServerCfg* getServerByIndex(**int** index);

To fetch a server interface using its index

ServerCfg* = getServerByIndex(int index)

Parameters: index – These servers are in array for which index is sent ↩→to get the respective server config

Return value: ServerCfg* – ServerCfg class object

Raises: None

ServerCfg* getServerByName(**const char*** name);

To fetch a server interface using its name

ServerCfg* = getServerByName(const char* name)

Parameters: name – These servers are in array for which name is sent to ↩→get the respective server config.

Return value: ServerCfg* – ServerCfg class object

Raises: None

ClientCfg* getClientByIndex(**int** index);

To fetch a client interface using it's index

ClientCfg * = getClientByIndex(int index);

Parameters: index – These clients are in array for which index is sent ↩→to get the respective client config.

Return value: ClientCfg* – ClientCfg class object

Raises: None

ClientCfg* getClientByName(**const char*** name);

To fetch a client interface using its name

ClientCfg*   = getClientByName (const char* name)

Parameters: name – These clients are in array for which name is sent to ↩→get the respective client config.

Return value: ClientCfg* – ClientCfg class object

Raises: None

PublisherCfg* getPublisherByIndex(**int** index);

To fetch a publisher interface using its index

PublisherCfg * = getPublisherByIndex(int index);

Parameters: index – These publishers are in array for which index is ↩→sent to get the respective publisher config.

Return value: PublisherCfg* – PublisherCfg class object

Raises: None

intel.

PublisherCfg* getPublisherByName(**const char*** name);

To fetch a publisher interface using its name

PublisherCfg * = getPublisherByName(const char* name)

Parameters: name – These publishers are in array for which name is sent ↵→to get the respective publisher config.

Return value: PublisherCfg* – PublisherCfg class object

Raises: None

SubscriberCfg* getSubscriberByIndex(**int** index);

To fetch a subscriber interface using its index

ServerCfg* = getSubscriberByIndex(int index)

Parameters: index – These subscribers are in array for which name is ↵→sent to get the respective subscriber config.

Return value: SubscriberCfg* – SubscriberCfg class object

Raises: None

SubscriberCfg* getSubscriberByName(**const char*** name);

To fetch a subscriber interface using its name

SubscriberCfg * = getSubscriberByName(const char* name)

Parameters: name – These subscribers are in array for which name is ↵→sent to get the respective subscriber config.

Return value: SubscriberCfg* – SubscriberCfg class object

Raises: None

2. **ConfigMgr**

config_value_t* getConfigValue(**char*** key);

Gets value from respective application's config

config_value_t * = getConfigValue(char* key)

Parameters: *key – Key for which value is needed*

Return value: config_value_t* – config_value_t object

Raises: None

bool watch(**char*** key, callback_t watch_callback, **void*** user_data);

Register a callback to watch on any given key

bool value = watch(char* key, callback_t watch_callback, void* user_data);

Parameters: key – key to watch

Parameters: watch_callback – callback object

Parameters: user_data – user data to be sent to callback

Return value: *bool – Boolean whether the callback was registered*

Raises: None

bool watchPrefix(**char*** prefix, callback_t watch_callback, **void*** user_‹→data);

Register a callback to watch on any given key prefix

Bool value = watchPrefix(char* prefix, callback_t watch_callback, void* user_‹→data)

Parameters: *prefix – key prefix to watch*

Parameters: watch_callback – callback object

Parameters: user_data – user data to be sent to callback

Return value: *bool – Boolean whether the callback was registered*

Raises: None

bool watchConfig(callback_t watch_callback, **void*** user_data);

Register a callback to watch on any given key

Bool value = watchConfig(callback_t watch_callback, void* user_data)

Parameters: key – key to watch

Parameters: watch_callback – callback object

Parameters: user_data – user data to be sent to callback

Return value: bool – Boolean whether the callback was registered

Raises: None

bool watch(**char*** key, callback_t watch_callback, **void*** user_data);

Register a callback to watch on app config

bool value = watch(char* key, callback_t watch_callback, void* user_data)

Parameters: watch_callback – callback object

Parameters: user_data – user data to be sent to callback

Return value: *bool - Boolean whether the callback was registered*

Raises: None

---

bool watchInterface(callback_t watch_callback, **void*** user_data);

Register a callback to watch on app interface

bool value = watchInterface(callback_t watch_callback, void* user_data);

Parameters: watch_callback – callback object

Parameters: user_data – user data to be sent to callback

Return value: *bool - Boolean whether the callback was registered*

Raises: None

3. **Publisher**

publisher_cfg is the sub class of app_cfg which contains publisher related APIs

---

config_t* getMsgBusConfig();

Constructs message bus config for Publisher

config_t*   = getMsgBusConfig()

Return value: config_t* - On Success, JSON msg bus publisher config of ↪type config_t

                     - On failure, On success, returns NULL

Raises: None

---

std::string getEndpoint();

To get endpoint for particular publisher from its interface config

std::string = getEndpoint()

Return value: std::string – On Success, returns Endpoint of server config

                     - On Failure, returns empty string

Raises: None

---

config_value_t* getInterfaceValue(**const char*** key);

To get particular interface value from Publisher interface config

config_value_t* = getInterfaceValue(const char* key)

Parameters: key – Key on which interface value is extracted

Return value: config_value_t* – On Success, returns config_value_t object

                     - On Failure, returns NULL

Raises: None

std::vector<std::string> getTopics() ;

To get topics from publisher interface config on which data will‹→be published

std::vector<std::string> = getTopics()

Return value: *vector<string>* – On Success, returns Topics of publisher‹→config

- On Failure, returns empty vector

Raises: None

bool setTopics(std::vector<std::string> topics_list)

To set new topics for publisher in publishers interface config

bool value = setTopics(std::vector<std::string> topics_list)

Parameters: topics_list – List of topics to be set

Return value: bool – Boolean whether topics were set

Raises: None

std::vector<std::string> getAllowedClients();

To get the names of the clients allowed to get publishers data

std::vector<std::string> = getServerByName(const char* name)

Return value: *vector<string>* – On Success, Allowed client of publisher‹→config

- On Failure, returns empty vector

Raises: None

4. **Subscriber**

subscriber_cfg is the sub class of app_cfg which contains subscriber related APIs

config_t* getMsgBusConfig();

Constructs message bus config for Subscriber

config_t* = getMsgBusConfig();

Return value: config_t* – On Success, JSON msg bus subscriber config of‹→type config_t

- On failure, returns NULL

Raises: None

config_value_t* getInterfaceValue(**const char*** key);

To get particular interface value from Subscriber interface config

config_value_t* =getInterfaceValue(const char* key)

Parameters: key - Key on which interface value is extracted

Return value: config_value_t* - On success, returns config_value_t object

> **-***On failure, On success, returns NULL*

Raises: None

---

std::string getEndpoint();

To get endpoint for particular subscriber from its interface config

std::string = getEndpoint();

Parameters: key - Key on which interface value is extracted

Return value: *std::string* – On Success, returns Endpoint of server config

> - On Failure, returns empty string

Raises: None

---

std::vector<std::string> getTopics()

To gets topics from subscriber interface config on which ↵→subscriber receives data

std::vector<std::string> = getTopics()

Parameters: key - Key on which interface value is extracted

Return value: *vector<string>* – On Success, returns Topics of subscriber ↵→config

> - On Failure, returns empty vector

Raises: None

---

bool setTopics(std::vector<std::string> topics_list);

To sets new topics for subscriber in subscribers interface config

bool value =setTopics(std::vector<std::string> topics_list)

Parameters: *topics_list - List of topics to be set*

Return value: bool – Boolean whether topics were set

Raises: None

5. **Server**

    server_cfg is the sub class of app_cfg which contains server related APIs

---

config_t* getMsgBusConfig();

Constructs message bus config for Subscriber

config_t* = getMsgBusConfig();

Return value: config_t* – On Success, JSON msg bus subscriber config of↵→type config_t

 - On failure, returns NULL

Raises: None

---

config_value_t* getInterfaceValue(**const char*** key);

To get particular interface value from Subscriber interface config

config_value_t* =getInterfaceValue(const char* key)

Parameters: key - Key on which interface value is extracted

Return value: config_value_t* – On success, returns config_value_t object

 -*On failure, On success, returns NULL*

Raises: None

---

std::string getEndpoint();

To get endpoint for particular server from its interface config

std::string = getEndpoint();

Parameters: key - Key on which interface value is extracted

Return value: *std::string* – On Success, returns Endpoint of server config

 - On Failure, returns empty string

Raises: None

---

std::vector<std::string> getAllowedClients();

To get the names of the clients allowed to connect to server

std::vector<std::string> =getAllowedClients();

Parameters: *topics_list - List of topics to be set*

Return value: vector<string> – On Success, returns Allowed client of↵→server config

 - On Failure, returns empty vector

Raises: None

6. **Client**

client_cfg is the sub class of app_cfg which contains client related APIs

config_t* getMsgBusConfig();

Constructs message bus config for Subscriber

config_t* = getMsgBusConfig();

Return value: config_t* – On Success, returns JSON msg bus server confi

- On failure, returns NULL

Raises: None

config_value_t* getInterfaceValue(**const char*** key);

To fetch particular interface value from Client interface config

config_value_t* =getInterfaceValue(const char* key)

Parameters: key - Key on which interface value is extracted

Return value: config_value_t* - On success, returns config_value_t object

-*On failure, On success, returns NULL*

Raises: None

std::string getEndpoint();

To fetch Endpoint for particular client from its interface config

std::string = getEndpoint();

Parameters: key - Key on which interface value is extracted

Return value: *std::string* – Endpoint of client config of type std::string

Raises: None

1.3   Sample TCP publisher-many-prefix-match output:

```
./publisher-many-prefix-match ./configs/tcp_publisher_no_security.json 5

[Fri Jan 17 11:52:34 2020]  INFO:main:165: Initializing msgbus context with
config './configs/tcp_publisher_no_security.json'

[Fri Jan 17 11:52:34 2020] DEBUG:msgbus_initialize:42: Checking if vendor is
Intel

[Fri Jan 17 11:52:34 2020] DEBUG:msgbus_initialize:53: Running on GenuineIntel

[Fri Jan 17 11:52:34 2020] DEBUG:msgbus_initialize:55: Initilizing message bus

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_initialize:137: Initilizing zeromq
message bus

[Fri Jan 17 11:52:34 2020] ERROR:get_config_value:138: JSON does not contain
key: zmq_recv_hwm
```

```
[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_initialize:218: Initializing ZeroMQ
for TCP communication

[Fri Jan 17 11:52:34 2020] ERROR:get_config_value:138: JSON does not contain
key: allowed_clients

[Fri Jan 17 11:52:34 2020]  WARN:zap_initialize:204: Running ZeroMQ TCP
sockets without ZAP authentication

[Fri Jan 17 11:52:34 2020]  INFO:main:179: Initializing 5 publishsers

[Fri Jan 17 11:52:34 2020]  INFO:main:209: Initializing publisher for topic:
pub/A/B/-0

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub/A/B/-0'

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 127.0.0.1

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: :

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 5569

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI: tcp://127.0.0.1:5569

[Fri Jan 17 11:52:34 2020] ERROR:get_config_value:138: JSON does not contain
key: server_secret_key

[Fri Jan 17 11:52:34 2020]  WARN:init_curve_server_socket:1529: ZeroMQ TCP
socket running without encryption

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:440: ZeroMQ publisher
created

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: NWLR

[Fri Jan 17 11:52:34 2020] DEBUG:shared_sock_new:101: Creating socket monitor
for inproc://NWLR

[Fri Jan 17 11:52:34 2020] DEBUG:shared_sock_new:120: Connecting monitor ZMQ
socket

[Fri Jan 17 11:52:34 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub/A/B/-0

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized

[Fri Jan 17 11:52:34 2020]  INFO:main:209: Initializing publisher for topic:
pub/A/B/-1

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub/A/B/-1'

[Fri Jan 17 11:52:34 2020]  INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 127.0.0.1

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: :

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 5569

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI: tcp://127.0.0.1:5569

[Fri Jan 17 11:52:34 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub/A/B/-1

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized
```

```
[Fri Jan 17 11:52:34 2020]  INFO:main:209: Initializing publisher for topic:
pub/A/B/-2

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub/A/B/-2'

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 127.0.0.1

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: :

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 5569

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI: tcp://127.0.0.1:5569

[Fri Jan 17 11:52:34 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub/A/B/-2

[Fri Jan 17 11:52:34 2020]  INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized

[Fri Jan 17 11:52:34 2020]  INFO:main:209: Initializing publisher for topic:
pub/A/B/-3

[Fri Jan 17 11:52:34 2020]  INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub/A/B/-3'

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 127.0.0.1

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: :

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 5569

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI: tcp://127.0.0.1:5569

[Fri Jan 17 11:52:34 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub/A/B/-3

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized

[Fri Jan 17 11:52:34 2020]  INFO:main:209: Initializing publisher for topic:
pub/A/B/-4

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub/A/B/-4'

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 127.0.0.1

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: :

[Fri Jan 17 11:52:34 2020] DEBUG:concat_s:68: 5569

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI: tcp://127.0.0.1:5569

[Fri Jan 17 11:52:34 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub/A/B/-4

[Fri Jan 17 11:52:34 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized

[Fri Jan 17 11:52:34 2020]  INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:34 2020]  INFO:pub_run:77: Publishing message
```

```
[Fri Jan 17 11:52:35 2020]   INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:35 2020]   INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:35 2020]   INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:35 2020]   INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:35 2020]   INFO:pub_run:77: Publishing message

[Fri Jan 17 11:52:36 2020]   INFO:pub_run:77: Publishing message
```

1.4 Sample TCP Subscriber output which subscribes to multiple published topics:

```
./subscriber ./configs/tcp_subscriber_no_security_prefix_match.json pub/

[Fri Jan 17 11:53:15 2020] DEBUG:msgbus_initialize:42: Checking if vendor is
Intel

[Fri Jan 17 11:53:15 2020] DEBUG:msgbus_initialize:53: Running on GenuineIntel

[Fri Jan 17 11:53:15 2020] DEBUG:msgbus_initialize:55: Initilizing message bus

[Fri Jan 17 11:53:15 2020] DEBUG:proto_zmq_initialize:137: Initilizing zeromq
message bus

[Fri Jan 17 11:53:15 2020] ERROR:get_config_value:138: JSON does not contain
key: zmq_recv_hwm

[Fri Jan 17 11:53:15 2020] DEBUG:proto_zmq_initialize:218: Initializing ZeroMQ
for TCP communication

[Fri Jan 17 11:53:15 2020] ERROR:get_config_value:138: JSON does not contain
key: allowed_clients

[Fri Jan 17 11:53:15 2020]  WARN:zap_initialize:204: Running ZeroMQ TCP
sockets without ZAP authentication

[Fri Jan 17 11:53:15 2020] ERROR:get_config_value:138: JSON does not contain
key: zmq_tcp_publish

[Fri Jan 17 11:53:15 2020] DEBUG:proto_zmq_initialize:229: ZeroMQ TCP not
configured for publishing

[Fri Jan 17 11:53:15 2020]  INFO:main:108: if topic name explicitly given

[Fri Jan 17 11:53:15 2020] DEBUG:proto_zmq_subscriber_new:543: ZeroMQ
subscribing to pub/

[Fri Jan 17 11:53:15 2020] DEBUG:concat_s:68: 127.0.0.1

[Fri Jan 17 11:53:15 2020] DEBUG:concat_s:68: :

[Fri Jan 17 11:53:15 2020] DEBUG:concat_s:68: 5569

[Fri Jan 17 11:53:15 2020] DEBUG:proto_zmq_subscriber_new:557: ZeroMQ creating
socket for URI: tcp://127.0.0.1:5569

[Fri Jan 17 11:53:15 2020] DEBUG:concat_s:68: NWLR

[Fri Jan 17 11:53:15 2020] DEBUG:shared_sock_new:101: Creating socket monitor
for inproc://NWLR

[Fri Jan 17 11:53:15 2020] DEBUG:shared_sock_new:120: Connecting monitor ZMQ
socket
```

```
[Fri Jan 17 11:53:15 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub/

[Fri Jan 17 11:53:15 2020] ERROR:get_config_value:138: JSON does not contain
key: server_public_key

[Fri Jan 17 11:53:15 2020] ERROR:get_config_value:138: JSON does not contain
key: client_public_key

[Fri Jan 17 11:53:15 2020] ERROR:get_config_value:138: JSON does not contain
key: client_secret_key

[Fri Jan 17 11:53:15 2020]  WARN:init_curve_client_socket:1633: ZeroMQ TCP
client socket running in insecure mode

[Fri Jan 17 11:53:15 2020] DEBUG:proto_zmq_subscriber_new:633: ZeroMQ
subscription finished

[Fri Jan 17 11:53:15 2020]  INFO:main:119: Running...

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1082: Receiving all of the message

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-4'

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1175: Received 25 bytes

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-4


[Fri Jan 17 11:53:16 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-4


[Fri Jan 17 11:53:16 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1082: Receiving all of the message

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-3'

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1175: Received 25 bytes

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-3


[Fri Jan 17 11:53:16 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-3


[Fri Jan 17 11:53:16 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1082: Receiving all of the message

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-2'

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1175: Received 25 bytes

[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-2


[Fri Jan 17 11:53:16 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-2
```

```
[Fri Jan 17 11:53:16 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-0'
[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-0


[Fri Jan 17 11:53:16 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-0


[Fri Jan 17 11:53:16 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-1'
[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:16 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-1


[Fri Jan 17 11:53:16 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-1


[Fri Jan 17 11:53:16 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-4'
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-4


[Fri Jan 17 11:53:17 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-4


[Fri Jan 17 11:53:17 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-0'
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-0


[Fri Jan 17 11:53:17 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-0


[Fri Jan 17 11:53:17 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
```

```
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-2'
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-2


[Fri Jan 17 11:53:17 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-2


[Fri Jan 17 11:53:17 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-3'
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-3


[Fri Jan 17 11:53:17 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-3


[Fri Jan 17 11:53:17 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-1'
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:17 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-1


[Fri Jan 17 11:53:17 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-1


[Fri Jan 17 11:53:17 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-4'
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-4


[Fri Jan 17 11:53:18 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-4


[Fri Jan 17 11:53:18 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1082: Receiving all of the message
```

```
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-0'
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-0


[Fri Jan 17 11:53:18 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-0


[Fri Jan 17 11:53:18 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-3'
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-3


[Fri Jan 17 11:53:18 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-3


[Fri Jan 17 11:53:18 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-1'
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-1


[Fri Jan 17 11:53:18 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-1


[Fri Jan 17 11:53:18 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1082: Receiving all of the message
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-2'
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1175: Received 25 bytes
[Fri Jan 17 11:53:18 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-2


[Fri Jan 17 11:53:18 2020]  INFO:main:129: Topic in the received message on
subscriber is pub/A/B/-2


[Fri Jan 17 11:53:18 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Fri Jan 17 11:53:19 2020] DEBUG:base_recv:1082: Receiving all of the message
```

```
[Fri Jan 17 11:53:19 2020] DEBUG:base_recv:1103: Received message for
'pub/A/B/-4'

[Fri Jan 17 11:53:19 2020] DEBUG:base_recv:1175: Received 25 bytes

[Fri Jan 17 11:53:19 2020] DEBUG:base_recv:1201: env->name = pub/A/B/-4
```

1.5   Sample IPC mode multi-topic publisher's output:

```
H270M-
3H:~/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/build/exampl
es$  ./publisher-many ./configs/ipc_example_config_multi_topics.json 5

[Sat Jan 18 02:20:06 2020]  INFO:main:165: Initializing msgbus context with
config './configs/ipc_example_config_multi_topics.json'

[Sat Jan 18 02:20:06 2020] DEBUG:msgbus_initialize:42: Checking if vendor is
Intel

[Sat Jan 18 02:20:06 2020] DEBUG:msgbus_initialize:53: Running on GenuineIntel

[Sat Jan 18 02:20:06 2020] DEBUG:msgbus_initialize:55: Initilizing message bus

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_initialize:137: Initilizing zeromq
message bus

[Sat Jan 18 02:20:06 2020] ERROR:get_config_value:138: JSON does not contain
key: zmq_recv_hwm

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_initialize:188: Initializing ZeroMQ
for IPC communication

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_initialize:208: ZeroMQ IPC socket
directory:
/home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/b
uild/examples/.socks

[Sat Jan 18 02:20:06 2020]  INFO:main:179: Initializing 5 publishsers

[Sat Jan 18 02:20:06 2020]  INFO:main:209: Initializing publisher for topic:
pub-0

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub-0'

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68:
/home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/b
uild/examples/.socks

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: /

[Sat Jan 18 02:20:06 2020] DEBUG:create_uri:1228: Initial IPC uri:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/

[Sat Jan 18 02:20:06 2020] DEBUG:create_uri:1253: Using socket file: multi-
topics

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: multi-topics

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/multi-topics
```

```
[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:440: ZeroMQ publisher
created

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: NWLR

[Sat Jan 18 02:20:06 2020] DEBUG:shared_sock_new:101: Creating socket monitor
for inproc://NWLR

[Sat Jan 18 02:20:06 2020] DEBUG:shared_sock_new:120: Connecting monitor ZMQ
socket

[Sat Jan 18 02:20:06 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub-0

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized

[Sat Jan 18 02:20:06 2020]  INFO:main:209: Initializing publisher for topic:
pub-1

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub-1'

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68:
/home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/b
uild/examples/.socks

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: /

[Sat Jan 18 02:20:06 2020] DEBUG:create_uri:1228: Initial IPC uri:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/

[Sat Jan 18 02:20:06 2020] DEBUG:create_uri:1253: Using socket file: multi-
topics

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: multi-topics

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/multi-topics

[Sat Jan 18 02:20:06 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub-1

[Sat Jan 18 02:20:06 2020]  INFO:main:209: Initializing publisher for topic:
pub-2

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub-2'

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68:
/home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/b
uild/examples/.socks

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: /

[Sat Jan 18 02:20:06 2020] DEBUG:create_uri:1228: Initial IPC uri:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/

[Sat Jan 18 02:20:06 2020] ERROR:get_config_value:138: JSON does not contain
key: pub-2

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: pub-2

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI:
```

```
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/pub-2

[Sat Jan 18 02:20:06 2020]INFO:pub_run:77: Publishing message for 'pub-1'

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:440: ZeroMQ publisher
created

[Sat Jan 18 02:20:06 2020]INFO:pub_run:77: Publishing message for 'pub-0'

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: BMQB

[Sat Jan 18 02:20:06 2020] DEBUG:shared_sock_new:101: Creating socket monitor
for inproc://BMQB

[Sat Jan 18 02:20:06 2020] DEBUG:shared_sock_new:120: Connecting monitor ZMQ
socket

[Sat Jan 18 02:20:06 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub-2

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized

[Sat Jan 18 02:20:06 2020]  INFO:main:209: Initializing publisher for topic:
pub-3

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub-3'

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68:
/home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/b
uild/examples/.socks

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: /

[Sat Jan 18 02:20:06 2020] DEBUG:create_uri:1228: Initial IPC uri:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/

[Sat Jan 18 02:20:06 2020] ERROR:get_config_value:138: JSON does not contain
key: pub-3

[Sat Jan 18 02:20:06 2020] DEBUG:concat_s:68: pub-3

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/pub-3

[Sat Jan 18 02:20:06 2020]  INFO:pub_run:77: Publishing message for 'pub-2'

[Sat Jan 18 02:20:06 2020] DEBUG:proto_zmq_publisher_new:440: ZeroMQ publisher
created

[Sat Jan 18 02:20:07 2020] DEBUG:concat_s:68: CDAR

[Sat Jan 18 02:20:07 2020] DEBUG:shared_sock_new:101: Creating socket monitor
for inproc://CDAR

[Sat Jan 18 02:20:07 2020] DEBUG:shared_sock_new:120: Connecting monitor ZMQ
socket

[Sat Jan 18 02:20:07 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub-3

[Sat Jan 18 02:20:07 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized
```

```
[Sat Jan 18 02:20:07 2020]  INFO:main:209: Initializing publisher for topic:
pub-4

[Sat Jan 18 02:20:07 2020] DEBUG:proto_zmq_publisher_new:377: Creating ZeroMQ
publisher for topic 'pub-4'

[Sat Jan 18 02:20:07 2020] DEBUG:concat_s:68:
/home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/b
uild/examples/.socks

[Sat Jan 18 02:20:07 2020] DEBUG:concat_s:68: /

[Sat Jan 18 02:20:07 2020] DEBUG:create_uri:1228: Initial IPC uri:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/

[Sat Jan 18 02:20:07 2020] ERROR:get_config_value:138: JSON does not contain
key: pub-4

[Sat Jan 18 02:20:07 2020] DEBUG:concat_s:68: pub-4

[Sat Jan 18 02:20:07 2020] DEBUG:proto_zmq_publisher_new:386: ZeroMQ publisher
URI:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/pub-4

[Sat Jan 18 02:20:07 2020] DEBUG:proto_zmq_publisher_new:440: ZeroMQ publisher
created

[Sat Jan 18 02:20:07 2020] INFO:pub_run:77:Publishing message for 'pub-3'

[Sat Jan 18 02:20:07 2020] DEBUG:concat_s:68: OWKK

[Sat Jan 18 02:20:07 2020] DEBUG:shared_sock_new:101: Creating socket monitor
for inproc://OWKK

[Sat Jan 18 02:20:07 2020] DEBUG:shared_sock_new:120: Connecting monitor ZMQ
socket

[Sat Jan 18 02:20:07 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub-4

[Sat Jan 18 02:20:07 2020] DEBUG:proto_zmq_publisher_new:493: Publisher
successfully initialized

[Sat Jan 18 02:20:07 2020]INFO:pub_run:77: Publishing message for 'pub-4'

[Sat Jan 18 02:20:07 2020]INFO:pub_run:77: Publishing message for 'pub-0'

[Sat Jan 18 02:20:07 2020]INFO:pub_run:77: Publishing message for 'pub-1'

[Sat Jan 18 02:20:07 2020]INFO:pub_run:77: Publishing message for 'pub-2'

[Sat Jan 18 02:20:08 2020]INFO:pub_run:77: Publishing message for 'pub-3'

[Sat Jan 18 02:20:08 2020]INFO:pub_run:77: Publishing message for 'pub-4'

[Sat Jan 18 02:20:08 2020]INFO:pub_run:77: Publishing message for 'pub-0'

[Sat Jan 18 02:20:08 2020]INFO:pub_run:77: Publishing message for 'pub-1'

[Sat Jan 18 02:20:08 2020]INFO:pub_run:77: Publishing message for 'pub-2'

[Sat Jan 18 02:20:09 2020]INFO:pub_run:77: Publishing message for 'pub-3'

[Sat Jan 18 02:20:09 2020]INFO:pub_run:77: Publishing message for 'pub-4'

[Sat Jan 18 02:20:09 2020]INFO:pub_run:77: Publishing message for 'pub-0'

[Sat Jan 18 02:20:09 2020]INFO:pub_run:77: Publishing message for 'pub-1'
```

```
[Sat Jan 18 02:20:09 2020]INFO:pub_run:77: Publishing message for 'pub-2'

[Sat Jan 18 02:20:10 2020]INFO:pub_run:77: Publishing message for 'pub-3'
```

1.6 Sample IPC mode multi-topic subscriber's output:

```
H270M-
3H:~/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/build/exampl
es$ ./subscriber ./configs/ipc_example_config_multi_topics.json pub-

[Sat Jan 18 02:20:09 2020] DEBUG:msgbus_initialize:42: Checking if vendor is
Intel

[Sat Jan 18 02:20:09 2020] DEBUG:msgbus_initialize:53: Running on GenuineIntel

[Sat Jan 18 02:20:09 2020] DEBUG:msgbus_initialize:55: Initilizing message bus

[Sat Jan 18 02:20:09 2020] DEBUG:proto_zmq_initialize:137: Initilizing zeromq
message bus

[Sat Jan 18 02:20:09 2020] ERROR:get_config_value:138: JSON does not contain
key: zmq_recv_hwm

[Sat Jan 18 02:20:09 2020] DEBUG:proto_zmq_initialize:188: Initializing ZeroMQ
for IPC communication

[Sat Jan 18 02:20:09 2020] DEBUG:proto_zmq_initialize:208: ZeroMQ IPC socket
directory:
/home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/b
uild/examples/.socks

[Sat Jan 18 02:20:09 2020]  INFO:main:108: if topic name explicitly given

[Sat Jan 18 02:20:09 2020] DEBUG:proto_zmq_subscriber_new:543: ZeroMQ
subscribing to pub-

[Sat Jan 18 02:20:09 2020] DEBUG:concat_s:68:
/home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsights/b
uild/examples/.socks

[Sat Jan 18 02:20:09 2020] DEBUG:concat_s:68: /

[Sat Jan 18 02:20:09 2020] DEBUG:create_uri:1228: Initial IPC uri:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/

[Sat Jan 18 02:20:09 2020] DEBUG:create_uri:1253: Using socket file: multi-
topics

[Sat Jan 18 02:20:09 2020] DEBUG:concat_s:68: multi-topics

[Sat Jan 18 02:20:09 2020] DEBUG:proto_zmq_subscriber_new:557: ZeroMQ creating
socket for URI:
ipc:///home/nagdeepgk/go/src/IEdgeInsights/common/libs/EIIMessageBus/IEdgeInsi
ghts/build/examples/.socks/multi-topics

[Sat Jan 18 02:20:09 2020] DEBUG:concat_s:68: NWLR

[Sat Jan 18 02:20:09 2020] DEBUG:shared_sock_new:101: Creating socket monitor
for inproc://NWLR

[Sat Jan 18 02:20:09 2020] DEBUG:shared_sock_new:120: Connecting monitor ZMQ
socket
```

```
[Sat Jan 18 02:20:09 2020] DEBUG:sock_ctx_new:202: Creating socket context for
pub-
[Sat Jan 18 02:20:09 2020] DEBUG:proto_zmq_subscriber_new:633: ZeroMQ
subscription finished
[Sat Jan 18 02:20:09 2020]  INFO:main:119: Running...
[Sat Jan 18 02:20:10 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:10 2020] DEBUG:base_recv:1103: Received message for 'pub-0'
[Sat Jan 18 02:20:10 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:10 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:10 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:10 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:10 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:10 2020] DEBUG:base_recv:1103: Received message for 'pub-1'
[Sat Jan 18 02:20:10 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:10 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:10 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:10 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:11 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:11 2020] DEBUG:base_recv:1103: Received message for 'pub-0'
[Sat Jan 18 02:20:11 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:11 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:11 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:11 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:11 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:11 2020] DEBUG:base_recv:1103: Received message for 'pub-1'
[Sat Jan 18 02:20:11 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:11 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:11 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1
```

```
[Sat Jan 18 02:20:11 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:12 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:12 2020] DEBUG:base_recv:1103: Received message for 'pub-0'
[Sat Jan 18 02:20:12 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:12 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:12 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:12 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:12 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:12 2020] DEBUG:base_recv:1103: Received message for 'pub-1'
[Sat Jan 18 02:20:12 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:12 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:12 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:12 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:13 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:13 2020] DEBUG:base_recv:1103: Received message for 'pub-0'
[Sat Jan 18 02:20:13 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:13 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:13 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:13 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:13 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:13 2020] DEBUG:base_recv:1103: Received message for 'pub-1'
[Sat Jan 18 02:20:13 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:13 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:13 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:13 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:14 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:14 2020] DEBUG:base_recv:1103: Received message for 'pub-0'
```

```
[Sat Jan 18 02:20:14 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:14 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:14 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:14 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Sat Jan 18 02:20:14 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:14 2020] DEBUG:base_recv:1103: Received message for 'pub-1'

[Sat Jan 18 02:20:14 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:14 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:14 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:14 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Sat Jan 18 02:20:15 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:15 2020] DEBUG:base_recv:1103: Received message for 'pub-0'

[Sat Jan 18 02:20:15 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:15 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:15 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:15 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Sat Jan 18 02:20:15 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:15 2020] DEBUG:base_recv:1103: Received message for 'pub-1'

[Sat Jan 18 02:20:15 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:15 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:15 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:15 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Sat Jan 18 02:20:16 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:16 2020] DEBUG:base_recv:1103: Received message for 'pub-0'

[Sat Jan 18 02:20:16 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:16 2020] DEBUG:base_recv:1201: env->name = pub-0
```

```
[Sat Jan 18 02:20:16 2020]   INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:16 2020]   INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:16 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:16 2020] DEBUG:base_recv:1103: Received message for 'pub-1'
[Sat Jan 18 02:20:16 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:16 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:16 2020]   INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:16 2020]   INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:17 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:17 2020] DEBUG:base_recv:1103: Received message for 'pub-0'
[Sat Jan 18 02:20:17 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:17 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:17 2020]   INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:17 2020]   INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:17 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:17 2020] DEBUG:base_recv:1103: Received message for 'pub-1'
[Sat Jan 18 02:20:17 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:17 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:17 2020]   INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:17 2020]   INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:18 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:18 2020] DEBUG:base_recv:1103: Received message for 'pub-0'
[Sat Jan 18 02:20:18 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:18 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:18 2020]   INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:18 2020]   INFO:main:136: Received: {"hello":42,"world":55.5}
```

```
[Sat Jan 18 02:20:18 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:18 2020] DEBUG:base_recv:1103: Received message for 'pub-1'

[Sat Jan 18 02:20:18 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:18 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:18 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:18 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Sat Jan 18 02:20:19 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:19 2020] DEBUG:base_recv:1103: Received message for 'pub-0'

[Sat Jan 18 02:20:19 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:19 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:19 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:19 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Sat Jan 18 02:20:19 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:19 2020] DEBUG:base_recv:1103: Received message for 'pub-1'

[Sat Jan 18 02:20:19 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:19 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:19 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:19 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Sat Jan 18 02:20:20 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:20 2020] DEBUG:base_recv:1103: Received message for 'pub-0'

[Sat Jan 18 02:20:20 2020] DEBUG:base_recv:1175: Received 25 bytes

[Sat Jan 18 02:20:20 2020] DEBUG:base_recv:1201: env->name = pub-0


[Sat Jan 18 02:20:20 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0


[Sat Jan 18 02:20:20 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}

[Sat Jan 18 02:20:20 2020] DEBUG:base_recv:1082: Receiving all of the message

[Sat Jan 18 02:20:20 2020] DEBUG:base_recv:1103: Received message for 'pub-1'

[Sat Jan 18 02:20:20 2020] DEBUG:base_recv:1175: Received 25 bytes
```

```
[Sat Jan 18 02:20:20 2020] DEBUG:base_recv:1201: env->name = pub-1


[Sat Jan 18 02:20:20 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-1


[Sat Jan 18 02:20:20 2020]  INFO:main:136: Received: {"hello":42,"world":55.5}
[Sat Jan 18 02:20:21 2020] DEBUG:base_recv:1082: Receiving all of the message
[Sat Jan 18 02:20:21 2020] DEBUG:base_recv:1103: Received message for 'pub-0'
[Sat Jan 18 02:20:21 2020] DEBUG:base_recv:1175: Received 25 bytes
[Sat Jan 18 02:20:21 2020] DEBUG:base_recv:1201: env->name = pub-0
[Sat Jan 18 02:20:21 2020]  INFO:main:129: Topic in the received message on
subscriber is pub-0
```

§