inbus Documentation

Release 1.0.2

Maarten Los

Mar 06, 2018

Contents

Release 1.0.2 (*What's new?*).

inbus stands for **in**connu message **bus** and is targeted at small devices running a limited number of applications exchanging small messages. It has a single goal: simple, connectionless brokering of messages between one or more publishers and one or more subscribers.

- Central broker: no complex relationships
- Limited scope: Does not try to be all things to all people
- Connectionless: No flow control, no guaranteed message delivery
- Simple JSON based protocol

CHAPTER 1

Table of Contents

1.1 Installation

```
$ pip install inbus-server
```

1.2 Usage

```
from inbus.server.inbus import Inbus

Inbus().run()
```

Now any Inbus client that adheres to the *Protocol* can publish and subscribe to messages.

Check ReadTheDocs for inbus.client

1.3 Protocol

NOTE Use of the words, must, should, could, etc. adheres to the best practice suggested in RFC2119 (https://www.ietf.org/rfc/rfc2119.txt)

1.3.1 Description

Protocol messages MUST be specified in the following JSON format:

```
{
    "version" : <inbus-version>,
    "opcode" : <opcode>,
    "application" :[ <app-key>, <app-type> ],
```

```
    "address" : [ <ip-number>, <port> ],
    "payload" : <payload>
}
```

All messages MUST contain all elements, even if they are not used.

Elements that do not apply to a particular type of message (as defined by its `<opcode>`), SHOULD be an empty string or zero, depending on the data type.

<inbus-version> Integer specifying the Inbus protocol version. MUST be 1.

<opcode> Integer specifying the type of message

- 0: reserved
- 1: subscribe
- 2: unsubscribe
- 3: publish
- 4-999: reserved

<app-key> String identifying the application to which the message applies.

The values `*` and `_inbus` are reserved for future use.

<app-type> Integer, specifying an application defined value. Can be used to distinguish multiple messages related to the same application.

The element only applies to *publish* messages.

<ip-number> String containing an IP number part of the subscriber address. In case of a publish message, the element does not apply.

<port> Integer containing the port number of the subscriber address. In case of a *publish* message, the element does NOT apply.

The subscriber address, together with the `app-key` uniquely identifies a subscription.

<payload> String specifying a user defined payload. This implies that binary data must be string-encoded. The element only applies to *publish* messages.

1.3.2 Infrastructure

The protocol SHOULD use port 7222

1.3.3 Example messages

Subscribe

```
{   "version" : 1 ,
    "opcode" : 1,
    "application" : [ "upnp", 0 ],
    "address" : [ "127.0.0.1", 3456 ],
    "payload" : ""
}
```

Subscription message indicating that the subscriber wants to receive messages from an application that publishes messages under the “upnp” app-key.

Unsubscribe

```
{   "version" : 1 ,
    "opcode" : 2,
    "application" : [ "upnp", 0 ],
    "address" : [ "127.0.0.1", 3456 ],
    "payload" : ""
}
```

Message indicating that the subscriber no longer wants to receive messages from the application that publishes messages under the "upnp" app-key.

Publish

```
{   "version" : 1 ,
    "opcode" : 3,
    "application" : [ "upnp", 17 ],
    "address" : [ "", 0 ],
    "payload" : "Omega - Gammapolis I. - 0:45"
}
```

Message sent by the application using the app-key "upnp", using app-type 17.

1.4 Design

The project is a first attempt to explore the thoughts presented in Object Thinking, by David West, Microsoft Press, 2004

This section describes the objects in the system, their responsibility, collaborators, as well as their methods.

MessageReceiver

Responsibilities Waits for raw Inbus network messages and passes them to the MessageTranslator

Collaborators

- (System)
- MessageTranslator

Methods

- waitForMessage

IncomingMessageTranslator

Responsibilities Translates raw Inbus messages to either a Subscribe, Unsubscribe or Publish method, and invokes those methods on its InbusMethodObserver

Collaborators List of InbusMethodObservers

Methods translate

Broadcaster isA **InbusMethodObserver**

Responsibilities Broadcasting Publications to a list of Subscribers

Collaborators

- Registry

- OutgoingMessageTranslator
- MessageSender

Methods publish

Registry isA **MesssageListener**

Responsibilities Manages a list of subscribers.

Collaborators None

Methods

- subscribe add a subscriber
- unsubscribe: remove from registry
- subscribers: returns a list of subscribers

OutgoingMessageTranslator

Responsibilities Translate the publish method into a raw Inbus network message

Collaborators None

Methods translate

MessageSender

Responsibilities Sends a raw Inbus network message to the network

Collaborators (System)

Methods send

1.5 ChangeLog

Version	Description	Date
1.0.0 1.0.1 1.0.2 1.0.3	• Initial version • Fixed broadcast bug • Include this changelog in package • Public release on PyPI	08-NOV-2017 21-DEC-2018 21-FEB-2018 04-MAR-2018