Metrics gathering protocol

Overview

The clients will send their data to the data collection and reporting daemon by sending frames to a unix socket. The location of this socket in the file system will be made available through an Android property (ro.kaios.services.daemon).

Frames are typed and bounded, and the daemon will close the connection with the client if any error occurs when decoding a client frame.

Frame Format

Each frame is made of 3 fields:

- a single byte identifying the type of frame.
- a 32 bits integer transmitted in network byte order specifying the length of the upcoming payload.
- a variable length byte stream consisting of the exact number of bytes expected from decoding the `length` field.

JSON Frames

JSON frames are identified by using 0x01 as the frame type. The JSON serialization must use a utf-8 encoding, and it is recommended to not use pretty printing to limit data size.

If the daemon fails to decode the JSON data, it will close the connection with the client.

JSON Payloads

Data reported as JSON allows for the bulk transmission of several values by the client. Note however that the daemon is ultimately responsible for aggregating data before sending it to the backend to make the best use of available resources.

```
[
{
    "seq_number": 3,
    "timestamp": 1499777073,
    "payload": {
        "field1": value1,
        "fieldN": valueN
    }
```

```
},
{
    "seq_number": 4,
    "timestamp": 1499777253,
    "payload": {
        "field1": value1,
        "fieldN": valueN
    }
}
]
```

Each frame is an array of one or more records. Each record has the following properties:

- `seq_number` is an increasing 64 bits integer used to correlate requests and responses.
- `timestamp` is the number of seconds elapsed since the Unix epoch.

The `payload` object contains the values to report.

Exchange flow

Upon connection, the client will send to the server a JSON frame with the following content:

```
{ "source": "ril_metrics" }
```

The value of the `source` property is a free form string used to identify the client. Only one client with a given source name can be connected to the server.

The serveur will then send to the client a JSON frame with the following content:

```
{ "ready": true }
```

Any client packet received before this initial server packet is sent will be rejected and the connection closed.

After receiving a client frame, the server will answer either with a success or failure message, returning the sequence id of the message for easier tracking by the client.

The error frame is a JSON message of the form

```
{ "error": `ErrorKind`, "seq_number": 12 }
```

where `ErrorKind` is a string which can be:

- “MissingTimestamp” if the `timestamp` property is missing.
- “InvalidTimestamp” if the `timestamp` property is not a positive integer.

- "InvalidJSON" in other cases where the message carries unexpected properties.

If the `sequence_number` is missing of malformed (ie. not a positive, growing integer) the connection will be closed since it becomes impossible to correlate responses with answers.

When receiving an error, the payload is ignored and the client can retry to send it after correcting the error.

The success frame is a JSON message of the form:

```
{ "success": true, "seq_number": 12 }
```

This signals to the client that it can free any bookeeping it is using for this record.

Commands configuration

To better control which data is reported, the clients can receive a filter configuration. Such configurations are made of 3 parameters typed as 64 bits unsigned integers.

The JSON representation is:

```
{ "NC": 0000, "ND": 1111, "NE": 2222 }
```

When starting, the daemon will initialize a default filter with the following values:

- NC = 0x00000003
- ND = 0x07008081
- NE = 0x040003CC

This filter value will be updated each time the JioService sends a new filter value to the daemon on the same socket used to relay data from the daemon to the JioService. The JioService needs to send *a \n terminated JSON string.*

When a client connects, it will get a *JSON frame* with the current filter value. This frame is sent over the socket used to send data from the client to the daemon, right after the initial handshake.

When a new filter value is sent to the daemon by the JioService, the daemon relays that updated filter to connected clients the same way it sends the initial value after a handshake.

Acknowledgement mechanism

The modem is expected to send back an acknoledgement when it receives a filter configuration. This is a JSON frame with the following payload:

```
{"kind":"FilterAck", "success":true, "reason": "..."}
```

The `reason` property is optional and can be used to specify error causes. This is a free form string.

When receiving such a ack frame, the daemon will relay the JSON payload to the JBS like other messages, as a JSON string.