Data Transfer Protocol

ArdeoNova Innovation Technologies, Inc

Revision date: **26/05/2017**

Device discovery mechanism

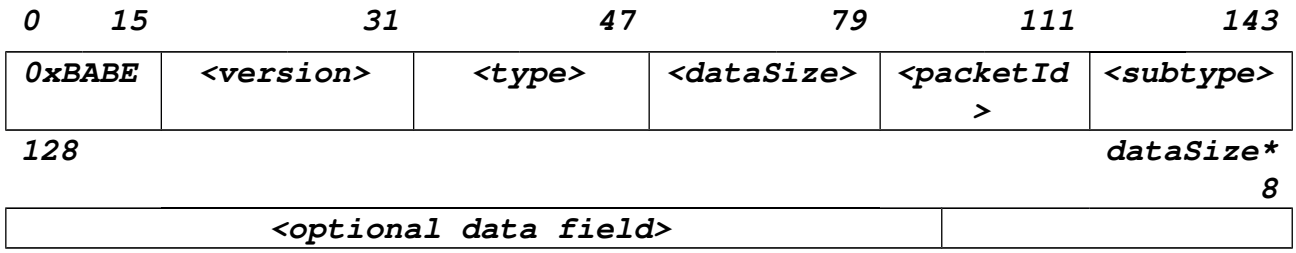
Device supports two modes of discovery, passive broadcast and discovery on demand. In both modes discovery packets are sent on ***12345*** UDP port. Discovery on demand occurs when device receives 2 bytes datagram that contains ***0xBABE*** value in *big-endian* byte order on ***44433*** UDP port. Discovery packet contains identification information and it has following structure(all fields in *big-endian* byte order):

Marker	***Device IP***	***Device port***	***Video port***	***Profile port***	***Device id***	***Device serial***
0xEAEA	***4 bytes int***	***2 bytes int***	***2 bytes int***	***2 bytes int***	***2 bytes int***	***4 bytes int***

Marker field is used to identify discovery packet, device IP address is contained in second field, TCP connection port is a third field, followed by two streaming ports. Last fields are used to identify device model.

Device control protocol

Device control protocol uses TCP transport layer and uses generic request-response convention. Request and response have same structure – data block prepended by header, header fields are integers in *big-endian* byte oreder. 18 bytes header has following structure:



First filed is header marker, its always 0xBABE, second field is protocol version, third field defines packet type.

PacketId and ***subtype*** fields meaning are dependent on packet type. Data field is optional and it is in JSON format.

There are 5 packet types:

```
enum PacketType : packetType_t {
    Heartbeat     = 0x1,
    SetParams     = 0x10,
    GetParams     = 0x20,
    Command       = 0x30,
    GetGenXml     = 0x40,
    Error         = 0xFF
};
```

Heartbeat packets contain no data field and is used to detect if connection is alive, all fields in this packet type are ignored except type field. Heartbeat packets are sent every 2 seconds by client connected to device.

0 15	31	47	79	111	143
0xBABE	***<version>***	***<type>***	***<dataSize>***	***<packetId>***	***<subtype>***
-//-	***0x1***	***0x1***	***0***	***0***	***0***

Both parameters packets GetParams and SetParams, have same structure and header fields meaning. Data field contains JSON data, with two main blocks – *description* block and *data* block.

0 15	31	47	79	111	143
0xBABE	***<version>***	***<type>***	***<dataSize>***	***<packetId>***	***<subtype>***
-//-	***0x1***	***0x10 or 0x20***	***dataSize***	***0***	***0***

Description block contains parameter visibility and access modes, data block contains parameter names and values:

```
{
    "description" : {
        "visibility" : "user",
        "types" : ["const", "rw", "ro"]
    },
    "data": {
        "param-name1" : "param-value1",
        "param-name2" : "param-value2"
    }
}
```

If ***subtype*** header field contains ‘1’ value, then parameter addresses are used instead of names.

Command packets are used to issue a command to device, data field is optional for this packet type, and is not used in current implementations, ***subtype*** header field contains command id.

0 15	31	47	79	111	143
0xBABE	***<version>***	***<type>***	***<dataSize>***	***<packetId>***	***<subtype>***
-//-	***0x1***	***0x30***	***dataSize***	***0***	***Command code***

Currently supported commands:

```
enum CommandMessage : subType_t {
    FactoryReset        = 1,        ///< Reset all parameters to factory values
    DeviceReboot        = 2,        ///< Reboot device
    CloseConnections    = 3,        ///< Force to close all active TCP
connections
    CustomCommand       = 0xFFFF    ///< Specific for custom devices.
};
```

Error packets hold error code in ***subtype*** field, data field contains optional string message, error packet may be received in response for any request packet(parametric packets or command packets).

0 15	31	47	79	111	143
0xBABE	<version>	<type>	<dataSize>	<packetId>	<subtype>
-//-	0x1	0xff	dataSize	0	error code

GetGenXml packet requests GeniCam XML description of device, data field contains entire XML, ***packetId*** and ***subtype*** header fields are ignored, it contains all parameters with its description – names and addresses.

0 15	31	47	79	111	143
0xBABE	<version>	<type>	<dataSize>	<packetId>	<subtype>
-//-	0x1	0x40	dataSize	0	0

Streaming protocol

UDP streaming protocol uses separate UDP sockets for video streaming and profile streaming, by default video is streamed to port 13377 and profiles are streamed to 13378 port. Data frame layout looks like this:

HEADER	DESCRIPTION BLOCK	DATA BLOCK

Header block has fixed size of 16 bytes and has folowing structure:

Header marker	Version	Flags	Description size	Data size	Frame number
0xA5A5	2 bytes	2 bytes	2 bytes	4 bytes	4 bytes

All fields in header are in *big-endian* byte order, header marker is always *0xA5A5*, followed by *version field*.

Flags field contains frame flags.

Flags are:

```
#define FLAG_BINARY_DESCRIPTION      0x0001
#define FLAG_LE                      0x0002
```

Binary description flag determines format of description block, it is either binary or JSON.

If ***FLAG_LE*** is set then data block is in *little-endian* byte order.

Description size field contains size of ***description block*** in bytes.

Data size field contains size of ***data block*** in bytes.

Frame number field is a counter of sent frames.

Description block contains frame format info, dimensions and other. There is two formats for description block – JSON and binary, it is determined by ***FLAG_BINARY_DESCRIPTION*** in header flag field.

JSON description contains following fields:

```
#define FRAME_TYPE_FIELD            "type"
#define FRAME_FORMAT_FIELD          "format"
#define FRAME_WIDTH_FIELD           "width"
#define FRAME_HEIGHT_FIELD          "height"
#define FRAME_DENOM_FIELD           "denom"
#define PROFILE_COUNTER_FIELD       "pcounter"
#define MEASUR_COUNTER_FIELD        "mcounter"
#define GENERIC_COUNTERS_FIELD      "gcounters"
```

```
#define FRAME_CRC_FIELD             "crc"
```

FRAME_TYPE field is an integer value and it determines what is stored in ***data block*** – video, or measurements.

FRAME_FORMAT field is an integer value that determines format of data block, values are specific to what is stored in data block.

FRAME_WIDTH integer, frame width.

FRAME_HEIGHT integer, frame height.

FRAME_DENOM integer, denominator for conversion to floating point types from integers.

PROFILE_COUNTER integer.

MEASUR_COUNTER integer.

FRAME_CRC – option CRC-16 value, crc is calculated over data block only.

Description layout in binary form.

Binary description block is used when ***FLAG_BINARY_DESCRIPTION*** is set, binary description has following layout:

Frame type	Frame format	Frame width	Frame height	Frame denominator	Profile counter	Measurement counter
2 bytes integer	2 bytes integer	2 bytes integer	2 bytes integer	4 bytes integer	4 bytes integer	4 bytes integer

All values are in big-endian byte order.

Data transmission.

Header is always sent as single 14-bytes datagram. Description block and data block are split into 1400 bytes datagrams, where first two bytes are the number of datagram, last block may have size less than 1400 bytes:

Header	Data chunk	Data chunk	...	Data chunk
14 bytes	1400 bytes	1400 bytes	...	<= 1000 bytes

Data chunk:

Block number	Data
2 bytes integer, big-endian byte order	1398 bytes

Numbers in data chunks allow to detect missing datagrams, and restore data in proper order.