DSM Communication Protocol

Contents

Server side configuration source code

NOTE: This is a "test server" version, dedicated for initial functionality testing (which could also be done via cloudview platform) with 1-2 cameras. Multithreaded (and multiple device) handling on the server side is up to the client to do additional development and/or necessary optimizations and modifications, to support more than 1-2 cameras at the same server instance.

Optionally - platforms that already support camera integrations could be used.

Together with the communication protocol, this server side source code can be downloaded and implemented upon personal servers for the ease of use and configuration, please see index.js, debug.js and protocol.js which can be copied into the server.

Server files & source code can be downloaded here.

Please see index.js, debug.js and protocol.js files. These files contain required support for camera downloads and they can be inserted straight into server. The folder can also be run as server and will work with file downloads, so if personal server is not available, there is always a possibility to use this default server.

All that is required is an open Port on the computer and that port has to be written in run_server.bat file. Also by clicking on Readme file, you will be able to see the settings which you can enter into that run_server.bat file by right clicking the mouse button on it and choosing edit option. These settings include choices like DualCam download, ADAS file download or Auto mode, Metadata inclusion and etc.

In this case, Notepad++ is chosen but you can choose any text editor in order to update the file.

File upload procedure

When a snapshot file is requested via SMS/GPRS command firstly the device checks if it has SD card inserted and if it has the requested snapshot in the SD card. The file is present, file upload is immediately triggered, otherwise the device checks if DSM camera is connected to FMX6yz and tries to download the file – file upload to the server starts only after the file was saved to FMXs SD card.

General command structure

General communication packet structure is as in table below. It consists of the command ID (2 bytes), data length of command and payload.

Table 5 General command structure

Command ID (2 bytes)	**Data length (2 bytes)**	**Data ([data length] number of bytes)**
0xFFFF	0xFFFF	<command data>

Modified file transfer protocol

Initialization packet

On connection, the device sends an initialization packet.

Table 6 Initialization packet structure

Header (2 bytes)	**Protocol ID (2 bytes)**	**IMEI (8 bytes)**	**Settings (4 bytes)**
0x0000	0xFFFF	0xFFFFFFFFFFFFFFFF	0xFFFFFFFF

Protocol ID – just a reference for the protocol version that is running on device (for server cross compatibility with older versions).

Settings flag contains information on what is available for download. Structure provided:

Table 7 Settings bytes structure

Settings, 4 B			
Byte 3	**Byte 2**	**Byte 1**	**Byte 0**
			*

* This bit will be set to 1 to indicate, that device has a file ready to upload to server. Server has to send file path request (0x000C) command to know what file is ready for upload.

Close session command (CMD_ID 0x0000)

In case when device connects to the server, but server does not expect it to connect, server will respond by sending close command (0x0000) after which connection will be terminated. This command is also used when the device connects to server for custom file sending and server finishes to send all custom files to device.

Table 8 Close session command structure

Command ID (2 bytes)	Data length (2 bytes)
0x0000	0x0000

Request file path (CMD_ID 0x000C)

First after connecting to server and seeing bit set in init packet the server should send file path request command (0x000C).

Table 9 Request file path command structure

Command ID (2 bytes)	Data length (2 bytes)	Blank field (2 bytes)
0x000C	0x0002	0x0000

File path response (CMD_ID 0x000D)

This is a response to file path request (0x000C) command.

Table 10 File path response command structure

Command ID (2 bytes)	Data length (2 bytes)	Blank field (Variable)
0x000D	1-512	<file path, or \0>

File metadata request command (CMD_ID 0x000A)

This command is used for retrieving extra information about the file requested by the server with the command 0x000C.

Table 11 File metadata request command structure

Command ID (2 bytes)	Data length (2 bytes)
0x000A	0x0000

File metadata response command (CMD_ID 0x000B)

This command is a response to the metadata request command 0x000A.

Response command returns information about a file: Unix timestamp, file type, trigger, GPS coordinates and video length. All values are sent as unsigned integers.

Table 12 File metadata response command structure

Command ID	Data length	Unix timestamp (4 bytes)	File type (1 byte)	Trigger (1 byte)	Latitude (4 bytes)	Longitude (4 bytes)	DSM event (2 bytes)	Driver name (10 bytes)
0x000B	0x000F	0xFFFFFFFF	0xFF	0xFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFF	0xFFFFFFFFFFFFFFFFFFFF

Table 13 File metadata description

Data	Description
Unix timestamp	Timestamp at the time of triggering an upload or the timestamp of an SMS request. For periodic images (current snapshots) the timestamp is updated when file info is received from the camera.
File type	File media type. 2 – current snapshot
Trigger	These trigger values are not to be confused with the values of configurable trigger parameters. 0 – none 1 – DIN1 2 – DIN2 3 – DIN3 4 – DIN4 5 – crash 6 – towing 7 – idling 8 – geofence 9 – unplug 10 – green driving 11 – server request (camreq SMS or GPRS command) 12 – periodic 13 – DSM event 14 - DSM file retransmit
Latitude Longitude	GPS coordinates at the time of triggering an upload. For periodic images (current snapshots) the coordinates are updated when file info is received from the camera. To get coordinates with a decimal point, these values should be divided by 1000000.
DSM Event	Multiple DSM events can be triggered at the same time and represented with multiple set bits: 0 bit – Drowsiness 1 bit – Distraction 2 bit – Yawning 3 bit – Phone 4 bit – Smoking 5 bit – Driver absence 6 bit – Mask 7 bit – Seat belt 8 bit – G-sensor
Driver name	A 10-character reversed string for the recognized driver’s name.

File request command (CMD_ID 0x0008)

After device is connected for file upload server initiates file transfer by sending this command.

Table 14 File request command structure

Command ID (2 bytes)	Data length (2 bytes)	File identifier
0x0008	0xFFFF	*File path has to be requested first via 0x000C command.*

Device should answer with start command (0x0001) indicating size and CRC of requested file.

Start file transfer command (CMD_ID 0x0001)

After device received file request command from server (0x0008) device sends start command (0x0001) with file data (file size in bytes, CRC).

Table 15 Start file transfer command structure

Command ID (2 bytes)	Data length (2 bytes)	File Size (4 bytes)	File CRC (2 bytes)
0x0001	0x0006	0xFFFFFFFF	0xFFFF

Resume file transfer command (CMD_ID 0x0002)

In a response to the start command (0x0001) a resume command (0x0002) must be sent from server.

Table 16 Resume file transfer command structure

Command ID (2 bytes)	Data length (2 bytes)	Packet offset (4 bytes)
0x0002	0x0004	0x00000000

To begin file transfer from the start, offset should be set to 0 (4 bytes value). In case when the file transfer was interrupted, to resume file transfer, offset can be set to the desired value (1 ≤ <offset> ≤ <file_packets>).

Synchronize file transfer command (CMD_ID 0x0003)

In a response to the resume command (0x0002), sync command (0x0003) is sent from device.

Table 17 Synchronize file transfer command structure

Command ID (2 bytes)	Data length (2 bytes)	File offset (4 bytes)
0x0003	0x0004	0xFFFFFFFF

By sending the sync command it is ensured that next data command will contain file data starting from the specified offset.

File data transfer command (CMD_ID 0x0004)

After sending the sync command (0x0004) file data transfer is started by sending data commands (0x0004).

Table 18 File data transfer command structure

Command ID (2 bytes)	Data length (2 bytes)	File data (up to 1024 bytes)	Data CRC (2 bytes)
0x0004	Up to 0x0402	0xFF…	0xFF

File data is split into 1024-byte parts, each part wrapped into data command and sent.

Note: *if command with a bad CRC is received, the resume command (0x0002) should be sent with last valid file offset. After receiving the resume command, server will stop sending data commands (0x0004) and continue communication from "Resume file transfer command (CMD_ID 0x0002)" step.*

CRC polynomial expression: ***0x8408***

When calculating CRC, the initial value is the same as the CRC value from the previously received packet (0x0004).

File transfer status command (CMD_ID 0x0005)

After file transfer is completed and no more files are required from device, server should send the file transfer command (0x0005) with completed status (0x00000000) to the device.

Note: this command does not work after executing repeat init command (0x0009) – in this case the server should send close session command (0x0000) mentioned above.

Table 19 File transfer status command structure

Command ID (2 bytes)	Data length (2 bytes)	Status (4 bytes)
0x0005	0x0004	0xFFFFFFFF

In case of server using invalid arguments, commands or not following the file request flow, the device will send this command with status field set to one of the few possible error codes (provided below).

Table 20 File transfer possible status values

Status value (hexadecimal)	Description	Notes
0x00000000	File transfer process completed	Sent from server
0x00000002	Failed to close GPRS	Sent from device
0x00000003	Failed to close socket	Sent from device
0x00000005	Invalid response from server to init packet	Sent from device
0x00000011	This error code forces the device to disconnect from server	Sent from device. Possible causes: · The requested file is not available by camera

After receiving the transfer completed command device should disconnect from the server.

Initialization packet repeat command (CMD_ID 0x0009)

When sent, the initialization packet is repeated. This is used, when all of the files are downloaded and additional check is carried out for any additional files, that may have been captured during the download operation.