Datafox GmbH • Dermbacher Straße 12-14 • D-36419 Geisa • www.datafox.de

Data protocol
HTTP and HTTPS communication

Flexible Data Acquisition Method

Version 1.16 / Firmware Release 04.03.22

Version / Changes:			
Version	**Date**	**Who**	**Description, changed content**
V 1.0 – 1.4	Nov 14, 2016	Bernd Ottmann	Document created based on Level 0 and Level 1
V 1.5	Mar 19, 2017	Sven Meyer	Chapter 2.2 added and Level 1 completed
V 1.6	Apr 27, 2017	Sven Meyer	Chapter 2.2 updated, Index-Structure replace by object-based interface
V 1.7	Jul 04, 2017	Sven Meyer	Chapter 3 updated Chapter 2.2.3: Entities for parameters in http request added
V 1.8	Dec 11, 2017	Sven Meyer	Representation of Comma by its entity within structured messages described. Definition of buffer sizes in chapter 2.4.4 added. Removed Chapter 3 – there is a specific revision history for https now Added Appendices A and B
V 1.9	Jun 25, 2019	Sven Meyer	Chapter 1.7 added 2.2.3.2.9, 2.2.3.2.12 - 2.2.3.2.20 added Appendixes A, B.5-B.7, C, D, E added
V 1.9.1	Dec 06, 2019	Sven Meyer	Appendix B.2: Misleading comment on client certificates not being implemented removed.
V 1.10	Jul 08, 2020	Sven Meyer Michael Wicher	Section 2.2.1.2 now described acknowledge by directly sending an IFF file. Service mode key added in chapter 2.2.3.2.3. Sections 2.2.3.2.22 to 2.2.3.2.26 added; COM.HTTP_MODE[.].SEND_IFF introduced. Appendix B.5 now includes a section on using ECC instead of RSA System Messages concerning Fingerprint added to section 2.3.1
V 1.11	Dec 23, 2020	Sven Meyer	Section 1.6 on basic authentication implementation added Bit definitions of df_ac2 (2.2.3.2.10) changed to match the documentation of the DFCom Communication Library (change is effective with 04.03.16 firmware version) Appendix B.5.3 update (Certificates and Microsoft Chromium Edge) Appendix C update (Initial device configuration using http) Appendix D update (Datafox Sample Web Server) Appendix E.5 added (Reference for HTTP flags)
V 1.12	Jan 13, 2022	Sven Meyer Michael Wicher Sven Meyer	Section 1.7: Http-Header User-Agent added Section 2.2.3.2.10 extended by new relay mask bits, bits are enumerated from 1 Section 2.2.3.2.18: Partial implementation of token “image” Section 2.3.1: System message code 1037 added Sections concerning System messages (2.3) and Encryption (2.4) separated from protocol content Appendix B.5.4 on securing client.key storage added. Appendix E.5: Flags on abs_path and absoluteURI changed, bits are enumerated from 1
V 1.12.1	Jun 03, 2022	Sven Meyer	Section 2.2.3.2.18 df_send_file: Token <finger2> added Appendix A.3.1.3.1 – CRC Computation Algorithm added Appendix C.3 added Appendix E.6 added
V 1.13	Nov 30, 2022	Sven Meyer	Section 2.2.3.2.27 – Firmware update added Section 2.3.1: List of System messages completed Section 2.3.2: added Appendix D: Documentation of Test-Webserver updated Appendix E: Troubleshooting renamed to Appendix T: Troubleshooting New Appendix E on Firmware-Update added

V 1.14	Feb. 15, 2023	Sven Meyer	Appendix B.6: Generate own derived certificates. Appendix E, Section 2.3.1: Handling of CPU mode in µSVC integrated. Section 2.3.1: System messages 3915, 3916, 4502 and 4503 added. Appendix T.7: Use of virtual hosts to realise endpoints with different certificate chains
V 1.14.1	May 03, 2023	Sven Meyer	Section 2.3.1: System message 1039 added Section 2.2.3.1.1: Processing of HTTP status codes 201-299 added Appendix A.1: DFCReset mapped to HTTP protocol
V 1.15	June 6th, 2023	Sven Meyer	Section 1.7: Included Response into documentation Section 1.8: Added Appendix F: Description of API-Level 0 moved here from section 2.1
V 1.15.1	June 29th, 2023	Sven Meyer	Appendix G on distributing firmware updates added
V 1.15.2	August 30th, 2023	Sven Meyer	Section 1.7.1: Header field “Data-Records-In-Device” added Section 2.2.1.3: Not concerning IFF upload process added Section 2.2.3.2.21 and following: Note on sort column and performance considerations concerning in-device data manipulation added. Section 2.3.1 System messages 1500 and 3006 added
V 1.16	April 30th, 2024	Sven Meyer	Appendix A.3.2**: File types**: Transfer of display design described; IFF-Type 0xDF0A marked as obsolete. Appendix C.2**: CRC Implementation of the info telegram**: CRC function corrected Appendix T.5**: Reference to http.flags**: TCP-Proxy added

This document is designed as a device-independent functional description. For device-specific information, the relevant manuals are available.

Content

1. Introduction

This document describes the communication between the devices and a web application using the HTTP(S)-protocol.

> ☞ **Please note:**
> Communication via HTTP is possible in the current firmware versions for devices with communication via TCP / IP. Thus, a module for LAN / WLAN or GPRS is required

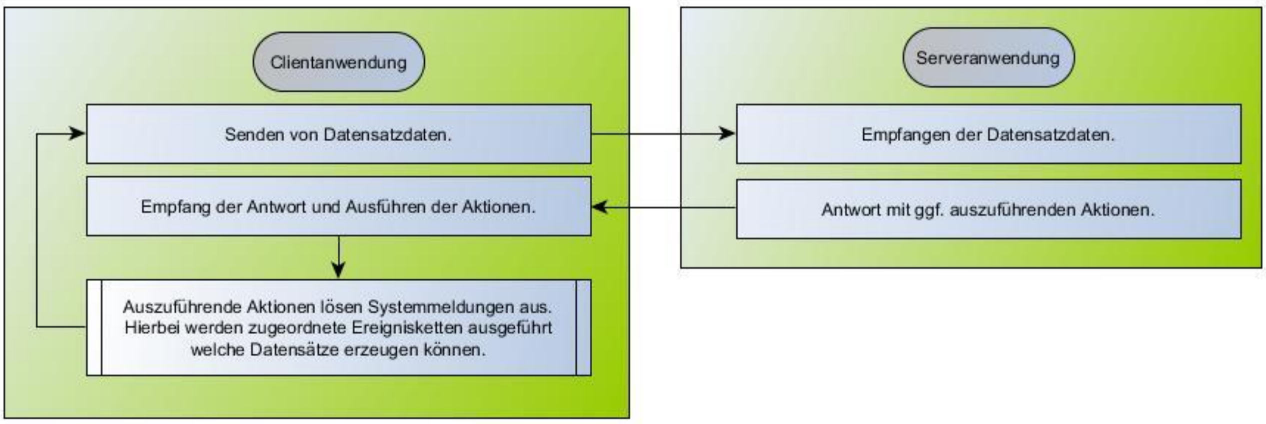
1.1. Explanation of terms used in the document

Term	Meaning
Web application	The application which receives requests from the device and sends appropriate responses to the device.
Request	Request that is sent from the device to the Web application. Contains e.g. data of a record.
Response	Response to a request that is sent from the Web application to the device. Contains e.g. the information that the data has been successfully processed or instructions as set the time, etc. are run.
Client	In this document, the device firmware is meant. Possibly a Web browser.
Server	In this document the Web server is meant, that contains the Web application.
System message	A signal generated by the firmware Message Event. This can be used in the setup to generate a record of an event chain. Especially at http system messages can be used to design the desired action (such as lists update) via a feedback data set to confirm.
Action	An action is defined as a process to be executed or individual instruction. For example, taking over a time, set a global variable or loading a list file a respective single action. To be performed actions can be specified by the Web application in the response and result in the client, if necessary to system messages.

1.2. Formatting

Parameter specifications in Request or Response are in italics.

1.3. Basic Scheme


Clientanwendung
Senden von Datensatzdaten.
Empfang der Antwort und Ausführen der Aktionen.
Auszuführende Aktionen lösen Systemmeldungen aus. Hierbei werden zugeordnete Ereignisketten ausgeführt welche Datensätze erzeugen können.
Serveranwendung
Empfangen der Datensatzdaten.
Antwort mit ggf. auszuführenden Aktionen.

- A record is created in the unit due to user interaction or system events.
- The current data is sent to the server.
- In the server's response, in addition to be performed to confirm the record reception actions can be sent with. For example, a text message to display or list data to be updated.
- Because of the actions that System messages are deleted, whereby records are generated in the device, which can then be sent to the server. About this so-called feedback records can get the message about the timing, success or failure of the action to your web app.

1.4. Feedback - records

In the section “signal processing” of the Device Setup "System Messages" can be connected to a chain of events. Through this event chains corresponding messages can generate records to the various events. The different events existing are described in the corresponding functions below.

! **Danger:**
Please note that no endless loops occur. For instance if you re-send to a reported error, without fixing the same action to the device, there may be an infinite loop.

1.5. API - Level

To document extensions and changes to the interface, requests and responses work with an "API - Level" declaration.

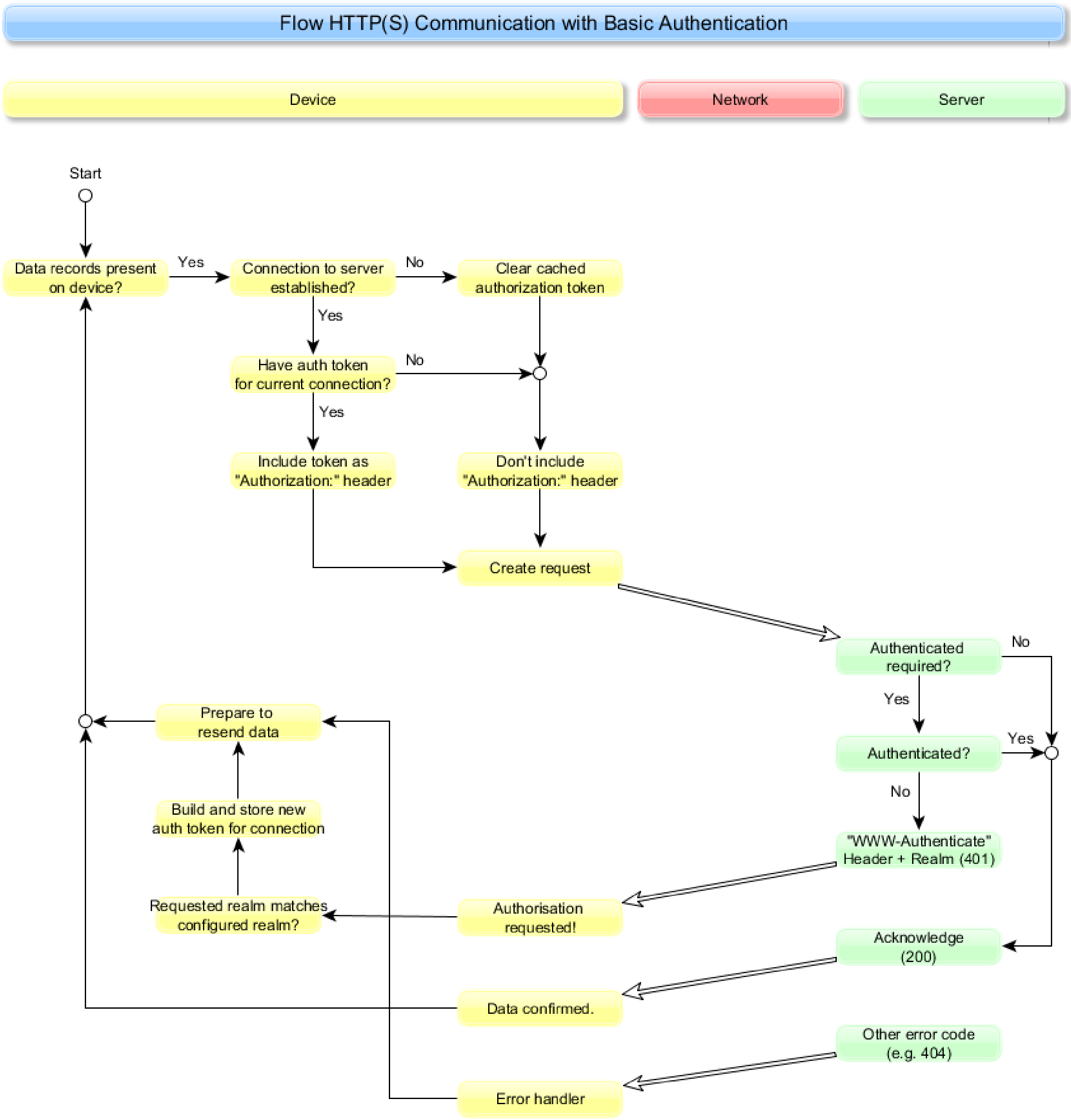
1.6. Basic Authentication

Basic authentication is a technology during HTTP communication workflow that may be used to query login credential for a certain area (so-called realm) by the web server. These credentials – after being requested by the server – are transmitted as an http header field.

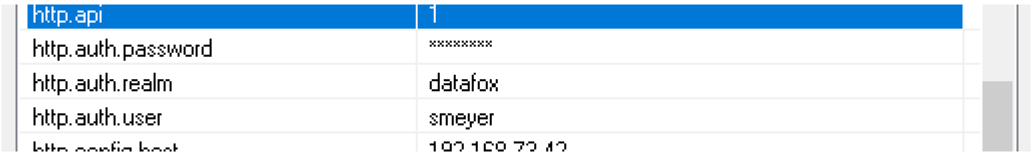
The technology is considered unsafe for unencrypted http communication, since the header field is only transcribed (and not encrypted) when being transferred. Thus – if intercepted – it can easily be decoded.

Since the technology may be safely used in the HTTPS area, Datafox Devices offer this approach starting with Version 04.03.16 as well.

The full communication workflow between a Datafox Device and you web service is as follows:


Flow HTTP(S) Communication with Basic Authentication
Device
Network
Server
Start
Data records present on device?
Yes
Connection to server established?
No
Clear cached authorization token
Yes
Have auth token for current connection?
No
Yes
Include token as "Authorization:" header
Don't include "Authorization:" header
Create request
Authenticated required?
No
Yes
Authenticated?
Yes
No
"WWW-Authenticate" Header + Realm (401)
Prepare to resend data
Build and store new auth token for connection
Requested realm matches configured realm?
Authorisation requested!
Acknowledge (200)
Data confirmed.
Other error code (e.g. 404)
Error handler

To configure basic authentication, you have to set three system variables, e.g. using the Device Communication Settings from the Datafox Studio:



http.api	1
http.auth.password	********
http.auth.realm	datafox
http.auth.user	smeyer
http.config.host	192.168.72.42

The password is shown only when you are entering it.

Additional information is available in RFC 7617: https://datatracker.ietf.org/doc/html/rfc7617.

1.7. Example of a http request using POST method

The device constructs and http request according to the following schematics (the example in this chapter reflects http API-level 1). The varying parts of the request are highlighted using yellow color and described below:

1.7.1. Request

```
POST https://extern.datafox.de:443/putdata HTTP/1.1
Host: extern.datafox.de:443
User-Agent: Datafox/04.03.18.04.http.10 11.3478
Accept-Charset: ISO 8859-1
Accept: application/x-www-form-urlencoded, text/html
Content-Length: 57
Data-Records-In-Device: 12
<weitere Header-Felder>

df_api=1&df_table=Alive&df_col_DT=2019-01-04T10%3A20%3A46
```

The information displayed on a green background is the actual data that the device transmits to the server. These are composed as follows:

- `df_api=1`: Identifier for data
- `df_table=Alive`: Name of the data set description in the device "Alive".
- `df_col_DT=2019-01-04T10%3A20%3A46`: Content of the field "DT" - with HTML atomic representation of colons as %3A. Consequently, it is 4 January 2019 10:20:46.

Please note:
The header field “User-Agent” is provided from Firmware Release 04.03.18.04 on with following structure:

`User-Agent: Datafox/`***`<Firmware incl. Branch-Tag> <Dev-ID>.<SN>`***

The elements shown in bold italics are replaced by actual device parameters when sending an http request:

- Firmware incl. Branch-Tag: The firmware version including a tag identifying a feature- or bugfix-version (e.g. “04.03.18.04.http.10”. The firmware version is 04.03.18.04, the branch tag “http.10”)
- Dev-ID: Internal device type id in decimal representation (11 represents an EVO 4.3)
- SN: Serial number in decimal representation

You can create a unique device identification by combining the Device Type Id (Dev-ID) and the serial number (SN).

Examples:

```
        User-Agent: Datafox/04.03.18.04.http.10 11.3478
        User-Agent: Datafox/04.03.18.05 11.3478
```

The highlighted fields inside the request above can be adopted using the following parameters:

- `POST https://extern.datafox.de:443/putdata HTTP/1.1`

  You can configure the http method (GET or POST) using the first letters of the system variable “com.http_mode[.].send”. If this variable starts by “POST”, a POST request will be generated by the device, otherwise the device will send a GET request.

- `POST https://extern.datafox.de:443/putdata HTTP/1.1`

  The system variable “com.http” configures if either http or https protocol are being used. If you enable https communication here, the “s” is added to the URL provided you are using “absoluteURI” addressing.

- `POST https://extern.datafox.de:443/putdata HTTP/1.1`

  According to RFC 2616 Sec. 5.2.1 there are abs_path and absoluteURI addressing available for “normal” http request. The mode shown above is absoluteURI addressing, a request using abs_path would be “`POST /putdata HTTP/1.1`”. Since absoluteURI is not support by all webservers correctly – despite being required for http/1.1 – you may configure the addressing mode using the system variable “http.flags”. Please see the details in Appendix Troubleshooting.

- `POST https://extern.datafox.de:443/putdata HTTP/1.1`

  The request type may be configured using the system variable “http.type”. The devices currently support types “1.0” and “1.1” which differ concerning the connection close behaviour – an http/1.0 connection is close immediately after the data has been transferred, an http/1.1 connection typically is kept alive for roughly 30 seconds after having transferred the data to enable another communication to be processed.

- `Host: extern.datafox.de:443`

  The host header inside the http request – according to RFC 2616 – may contain the port, which is being accessed. However, this is not supported by all web servers correctly. Thus you may control the transmission of the port using the system variable “http.flags” (again, please consult Appendix Troubleshooting for the details)

- Additional header fields may be includes using the file “header.extensions”. The content of this file will be included into any request the device sends to the server – provided the request type is enabled by the system variable “http.header_extension_flags“. This variable contains the sum of the values (as decimal number), for which additional header flags shall be sent:
  - 1 = data record requests
  - 2 = KVP requests
  - 4 = list data download requests (Setup and access control)
  - 8 = IFF download requests
  - 32768 = other requests

! **Attention:**
Please do not use header extensions excessively. The overall length of the header is limited by the device, please supply **at most 500 bytes** as header extensions.

Please keep in mind, that the http interface does not provide any duplication checks on the header fields. You only may add fields to the header in this way – there is no way to change the fields generated by the device.

Please ensure, that the “header.extensions” file’s line separator is CR and LF – as the HTTP specification requires. A recoding of the line separator is not performed by the device firmware.

1.7.2. Response

The server's response looks like this:

```
HTTP/1.1 200 Ok
Content-Length: 30
Content-Type: text/html; charset=ISO-8859-1
df-action-id: A6ID

df_api=1&df_kvp=var,http.alive
```

The server response is interpreted as follows:

- Checking the HTTP status code

```
HTTP/1.1 200 Ok
```

If this is between 200 and 299, the evaluation of the server response continues.

- Check of the API identifier in the response body:

```
df_api=1&df_kvp=var,http.alive
```

The protocol requires that "df_api=1" is at the beginning of the response body.

- Evaluation of instructions in the body:

```
df_api=1&df_kvp=var,http.alive
```

The body contains the instruction that the device should transmit the current value of the system variable "http.alive". This causes the unit to transmit this value to the web server in a follow-up request.

1.8. Basic configuration of a device for HTTP(S) communication

In order to be able to work as an HTTP-client, some settings are required on the unit. These are realised - after the basic network configuration has been established via LAN, WLAN or mobile radio - via the following system variables:

Name	Function
com.active	Set to "0". Active mode communication has priority over HTTP communication.

com.http	1 = HTTP communication 2 = HTTPS communication When using HTTPS, it is necessary to deposit server certificates!
com.http_mode[0].host com.http_mode[0].port	Defines the server endpoint for communication
com.http_mode[0].send	Defines the path on the server to which records are to be delivered (typically the script that processes the record in the server).
http.alive	Intervall (in Sekunden) zwischen zwei Alive-Meldungen des Geräts
http.api	Set to "1" to use API-Level 1.
http.flags	Flags that influence the behaviour of the HTTP interface.

☞ **Please note:**
As part of the development, you should consider communicating directly via HTTP instead of HTTPS. In this case, you can directly observe the communication between server and device with external tools such as Wireshark.

2. Description to the respective API - Level

Currently two different methods for HTTP communication are available.

2.1. Level 0

Using API-Level 0 is not recommended any more, since controlling the device is very limited. You can find the documentation at Appendix F: Description of API-Level 0.

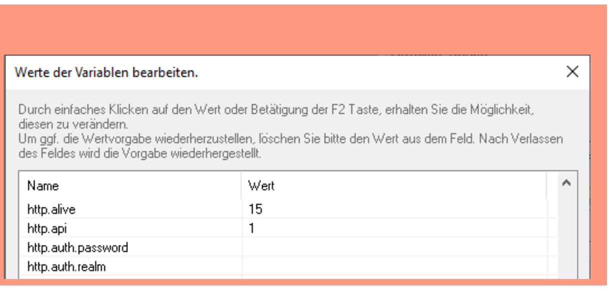
2.2. Level 1

!

Attention:

You will require Hardware V4 devices to use HTTP API-Level 1.

In the device, the HTTP API is selected via the `http.api` parameter in the unit communication settings.



Werte der Variablen bearbeiten.

Durch einfaches Klicken auf den Wert oder Betätigung der F2 Taste, erhalten Sie die Möglichkeit, diesen zu verändern.
Um ggf. die Wertvorgabe wiederherzustellen, löschen Sie bitte den Wert aus dem Feld. Nach Verlassen des Feldes wird die Vorgabe wiederhergestellt.

Name	Wert
http.alive	15
http.api	1
http.auth.password	
http.auth.realm	

2.2.1. Request types in API-Level 1

Http API-Level 1 distinguishes two types of requests.

- The transmission of data records uses URL encoding to transport the data. These requests can be answered with control commands and thus a task can be assigned to the device. See the subsections of section 2.2.3.

- The transmission of (typically IFF-encoded) binary data from the device to the server is based on the FORM upload mechanism.

2.2.1.1. Anatomy of a typical API-Level 1 request

- First field sent is ‘df_api=1’ always. This field is unencrypted even for active encryption requests.
- If it is a data set, its name is transmitted as parameter df_table.
    - All data fields of the record start with "df_col_" followed by the name of the field set in the unit setup.
- In fields of the type "Date and Time" the date and time are separated using a ‘T’.

2.2.1.2. Anatomy of a typical API-Level 1 response

There are two modes of sending a response to a record request in API-Level 1. You can send instructions to the device for execution. These are introduced by df_api=1 and must satisfy the following general conditions:

- The parameters are generally transmitted in the response body and stored URL-coded.
- If encryption is active, the parameters df_c, df_cb and df_ce are sent.

  Please also note that this encryption system is quite weak and HTTPS is available on hardware IV devices. Therefore, the description in section 2.4 has been separated out.

- All characters except letters, digits, - (minus), . (full stop), _ (underscore) and ~ (tilde) shall be encoded as %xx, where xx is the hexadecimal ASCII code of the character to be encoded [RFC 3986 section 2.3 / 2.4].
- Files to be downloaded from the device can only be provided by the web server itself to which the request was sent. It is not possible for the unit to send a request to another web server here.
- Path specifications with which a download is triggered are always considered absolute on the web server.

Alternatively to the “df_api=1” response, you may **directly send an IFF file** to the device as described in **A.3: Structure of a transfer file**. If the file sent conforms the IFF structure, the data record is considered to be acknowledged. The device will then evaluate the IFF file and apply its content.

Please send a status code between 200 and 299 in the HTTP response (see sectionn 2.2.3.1.1).

☞ **Please note:**
Please note that if the response from the server does not transmit the status code 200 and neither begins with df_api=1 nor represents a valid IFF file, the unit does not consider the data set as acknowledged.

You do not express your agreement with the content of the data set with the status code 200, but merely acknowledge receipt and assume responsibility for the reported data.

2.2.1.3. Transfer of image data and long barcodes (since 04.03.18.04)

To transport images (as collected either from the camera or from the signature function) or long barcode values an IFF file upload request of type 0xDF06 is generated – provided the system variable COM.HTTP_MODE[.].SEND_IFF is not empty (please see Appendix A.3 for details on the transfer file). This data transfer is attempted once as soon as the device records an image/long barcode. If this connection fails, the transfer is not attempted again automatically – you then may request the image explicitly using the df_send_file command.

☞ **Please note:**
With firmware release 04.03.20.06 is has been added, that an alive record is send immediately after uploading an image or a barcode to the server. During development we observed the Alive arriving ~ 200 ms after the IFF upload to the server is completed, using HTTP(S) and LAN. The measured runtime depends on many factors, so it may deviate in real customer networks.

The process of sending IFF file to an HTTP server follows the process detailed in section 1.7, however the Datafox protocol described in this document does not apply here. The following aspects shall be mentioned explicitly:

- The transfer of data of done to the SEND_IDD-Endpoint using a POST-Request.
- The data transfer uses “multipart/form-data” with MIME Multipart Media Encapsulation.
- You may **not** send an action to the device using the response. It is required that an HTTP status code 200 and an empty response body sent with the response.

A sample transaction where EVO 3.5 Pure (192.168.2.14) uploads fingerpint data to a server (192.168.1.162) can be accessed as Wireshark capture at

https://www.datafox.de/download/sample-device-upload.pcapng

2.2.2. Request

Request from the client (device) to the server.

2.2.2.1. Method: GET

> ☞ **Please note:**
> If you need a fixed parameter e.g. a client ID, which is sent with every request, then you can have it included into the http header (see section 1.7.1, “header extensions”) or set this in the URI of the system variables com.http_mode[n].send.
>
> Example: /path/to/script.php?clientid=1234&
>
> Please make sure that the string has a length limit of 63 characters and keep in mind, that “df_api=1” then will not be the first parameter in the request.

Parameter name	Meaning
df_table	Name of data set description to which the following data fields are assigned.
df_type	It is a device message not generated by a record (e.g. if the server asks the device for the value of a variable, it will receive the response as a "df_type=kvp"-coded response)
df_record_state	Encoding, of the state is considered “online” or “offline”: 1 = online 2 = online, record is being resend 3 = offline 4 = offline, record is being resend Please note: Including the `df_record_state` into the request is controlled by the system variable `http.record_state`. ☞ **Please note:** The `df_record_state` is only available for data records associated to operating the device. System messages, alive records or records from the access control subsystem, etc. do not carry this property.
df_col_ {Field Name}	Value of the field of the data record description. The name of the field as it appears in the record description, the static part “df_col_” prepended. “Col” is the abbreviation of “column”. For fields of the type “Date Time” the content is encoded as follows: Format: YYYY-MM-DDThh:mm:ss Example time = 2016-11-17T12:13:14

2.2.3. Response

Response from the server to the client (device).

Content-Type: application/x-www-form-urlencoded; charset: iso-8859-1

With the response you may send commands to the device as shown in the following table.

Instruction Name	Meaning
df_time=2016-11-17T12:13:14	Set the date and time on the device.
df_beep=1 (1-11)	OK signal / generate beep on the device
df_service=1,www.datafox.de,10047	Connect to the DFCom Communication library using active mode. Also possible with the DatafoxStudioIV. Specification of IP/URL and port possible.
df_var=setup.1,value	Change the value of a global variable in the setup.
df_ek=name	Trigger an action in the device. Start an input chain in signal processing.
df_msg=This\ris\ra\rMessage,5,1,0	Send a text message to the display.
df_msg_icon=2	Defines the icon to be used when showing a message in the device. The icon is taken from the design and associate to an input sequence (F2 in this example)
df_backlight=0,5,255,255,0,192	Defines the colour of a device's backlight – for a certain period of time as a RGBW value.
df_info_msg=Info\rMessage,0	Defines the text of an info message.
df_ac2=010,1,10,20,5	AC = access control. Trigger access control actions.
df_custom_msg_ac2=010,1,1,0,Hello%20 World	Sends a message to a device that is connect to the access control bus.
df_ao_ac2=0,1234	Acknowledges an action of the pre-checked access control.
df_trigger_ac2=1,011,6543210,0	Simulates a clocking performed at an access control RFID reader.
df_kvp=var,ID	Instructs the device to send the value of a system variable. The value is sent as a key-value-pair to the server.
df_set_relay=2,close,5	Defines the state of a relay for given period of time that is not handled by the access control module.
df_toggle_relay=2,5	Changes the state of a relay for a given period of time. The relay may not be handled by the access control module.
df_load_file=/path/on/server	Instructs the device to download a file from the server.
df_send_file=/logs/,syslog,0	Instructs the device to upload a file to the server.
df_remove_file=root:datafox.cert	Instructs the device to delete a specific file.

Instruction Name	Meaning
df_remove_finger=1980,all	Remove fingers from a fingerprint sensor.
df_setup_list=Personal,/path/to/list.txt	Give the device a new list of personnel, for example.
df_ac2_list=Identification,/path/to/list.txt	Give the device a new access control list.
df_table_count=list.PID	Counts the number of entries within a list stored on the device.
df_table_select=list.PID,/upload/form,Unit=Development,PID=5	Selects on or more entries from a list and uploads them to the server.
df_table_append=list.PID,9999,,Visitor,	Appends a record to a list stored on the device.
df_table_update=list.PID,,,Unit=	Changes values within a list stored on the device.
df_table_delete=list.PID,Unit=Development	Removes rows from a list stored on the device.

☞ **Please note:**
The response is used to send jobs to the terminal. The encoding follows the URL encoding as being used by a web browser when sending parameters to a web server. The parameters are composed out of key-value pairs – key and value are separated by ‘=’, pairs are separated by ‘&’.

Should you happen to need a ‘=’, ‘?’, ‘&’ or comma character inside your message, these will have to be represented by their entities: %3d (‘=’), %3f (‘?’), %26 (‘?’) or %2C (Comma).

2.2.3.1. Required parameters details

Parameter name	Meaning
df_api	Descripts the API-Level of the response. Please use 1. ! **Danger:** Please make sure that there are no endless loop by constantly sending missing or wrong information. The client sends the data set until it is acknowledged by a response ‘df_api=1’ and the HTTP Result “200 OK”.

2.2.3.1.1. Acknowledging records

A data set is acknowledged by replying ‘df_api=1’ from the server with an HTTP response having result code “200 Ok”. If the specification of ‘df_api’ is missing or the HTTP server sends a status code different from 200, the data record is not considered acknowledged and will be sent again.

Since firmware release 04.03.20.11 you may supply a status code 201-299 to express, that the data record should not be acknowledged but the instructions from the response (see following section) shall be processed.

Instead of sending a protocol level response “df_api=1…” a valid IFF file as defined in Appendix A.3 may be sent directly. If the device discovers this it checks the consistency of the file and – upon verifying the integrity – acknowledges the data record and evaluates / processes the IFF file’s content.

2.2.3.2. Optional parameter details

Optional parameters can be transmitted to the device in the HTTP response to trigger an action. This chapter describes the different instructions and their parameters.

2.2.3.2.1. Setting the device clock (df_time)

Parameter name	Meaning
df_time	The date and time that will be set by the device. The data and time supplied will be applied by the device when deviating more than +/- 10 seconds from the device’s clock. Format: YYYY-MM-ddThh:mm:ss Example: df_time = 2016-11-17T12:13:14

Setting the time in the unit is initiated by returning the df_time command.

```
df_time=2023-05-25T06:30:14
```

2.2.3.2.2. Emitting a beep sound at the device (df_beep)

<table>
<tr><th>Parameter name</th><th>Meaning</th></tr>
<tr><td>df_time</td><td>The date and time that will be set by the device. The data and time supplied will be applied by the device when deviating more than +/- 10 seconds from the device’s clock.
Format: YYYY-MM-ddThh:mm:ss
Example: df_time = 2016-11-17T12:13:14</td></tr>
<tr><td>df_beep</td><td>Beep signal.
The table is a ‘+’ used to represent a long tone and, - for a short tone.
<table>
<tr><td>1</td><td>OK signal</td></tr>
<tr><td>2</td><td>ERROR signal</td></tr>
<tr><td>3</td><td>+</td></tr>
<tr><td>4</td><td>- +</td></tr>
<tr><td>5</td><td>- -</td></tr>
<tr><td>6</td><td>+ +</td></tr>
<tr><td>7</td><td>- - -</td></tr>
<tr><td>8th</td><td>+ + +</td></tr>
<tr><td>9</td><td>- + -</td></tr>
<tr><td>10</td><td>+ - +</td></tr>
</table>
</td></tr>
</table>

	11	SMS signal

To emit a beep code at a device, send the instruction df_beep=<value> to the device. Please refer to the table above for the different, possible tone sequences.

2.2.3.2.3. Service mode (df_service)

Parameter name	Meaning
Flag	A value of 1 causes the client to enter the service mode after having transmitted all data records. A value of 2 causes the device to enter service mode even if data records are present on the device. ☞ **Please note:** Standard HTTP behaviour is that a connection is closed by the Web server. The client will switch to service mode only when the web server closed the connection. To end a connection, you can supply "Connection: close" in the HTTP header of the server's response. This causes the web server to end the connection.
Host	If the parameter is omitted, the value of the system variable com.http_mode[n].host is used.
Port	If the parameter is omitted, the value of the system variable com.http_mode[n].port is used.
Key	This parameter defines, if the service mode connection is - Unencrypted (parameter omitted or empty) - Encrypted with the first active mode server's key ("key0") - Encrypted with the second active mode server's key ("key1") ☞ **Please note:** This parameter requires at least firmware version 04.03.14.09.

The activation of the service mode required sending the following parameters

```
df_service=<Flag>,<Host>,<Port>,<Key>
```

with the server's response, e.g.

```
df_service=1,active-mode-server.my.net,8000
df_service=1,second-active-mode-server.my.net,8000,key1
```

2.2.3.2.4. Global, setup- or system variables (df_var)

Parameter name	Meaning
Name	Name of a global variable with their index 1-8. Example: setup.1 Name of a global variable. Example: setup.GlobVar1
Value	Value to be set

To set the value of a global variable, send

```
df_var=<Name>,<Value>
```

with the server's response, e.g.

```
df_var=setup.1,4711
```

2.2.3.2.5. Chain of events (df_ek)

Parameter name	Meaning
Name	Name of a chain of events, which is to be executed. ☞ **Please note:** Event chains are chains as used in signal processing. These do not allow user interaction, so if you want to post a message to a user at the terminal, e.g. in response to an entry/leave event, please consider using df_msg.

2.2.3.2.6. Show message on display (df_msg)

Parameter name	Meaning
Message	Text message to be shown on the display. A newline can be inserted into the text by specifying "\r". ! **Danger:** The message is only displayed if the 'server online' option is active in the device's setup. This option is found on the page 'default settings'. Additionally, is it required, that the "df_msg"-reply is sent **in response to an online record base on a user's device operation** – a system message or an alive data record are **never** online data records.

<table>
<tr><td>Duration</td><td>Specifies the time in seconds for how long the message is displayed.</td></tr>
<tr><td>Beep</td><td>Beep signal.
The table is a '+' used to represent a long tone and, - for a short tone.
<table>
<tr><td>0</td><td>No Signal</td></tr>
<tr><td>1</td><td>OK signal</td></tr>
<tr><td>2</td><td>ERROR signal</td></tr>
<tr><td>3</td><td>+</td></tr>
<tr><td>4</td><td>- +</td></tr>
<tr><td>5</td><td>- -</td></tr>
<tr><td>6</td><td>+ +</td></tr>
<tr><td>7</td><td>- - -</td></tr>
<tr><td>8th</td><td>+ + +</td></tr>
</table></td></tr>
<tr><td>Font</td><td>Specifies the font size and style.
<table>
<tr><td>0</td><td>Standard font</td></tr>
<tr><td>1</td><td>16 pixel (7 lines)</td></tr>
<tr><td>2</td><td>16 pixel, fixed width (7 lines)</td></tr>
<tr><td>3</td><td>19 pixel (6 lines)</td></tr>
<tr><td>4</td><td>19 pixel, fixed width (6 lines)</td></tr>
<tr><td>5</td><td>21 pixel (5 lines)</td></tr>
<tr><td>6</td><td>21 pixel, fixed width (5 lines)</td></tr>
</table>
! <b>Attention:</b>
The pixel values indicated in the table and lines are approximations and may vary depending on the device used.</td></tr>
</table>

To display a message on the receiving terminal, you may add the following parameter to the server‘s response:

```
df_msg=<Message>,<Duration>,<Beep>,<Font>
```

The following parameter

```
df_msg=This\ris\ra\rmessage,5,1,0
```

will trigger displaying “This is a message” in the terminal’s display (a single word per line) for 5 seconds. Additional an “ok”-beep code is generated.

! **Attention:**
Please be aware or the entity representation rules of ‚=‘, ‚?‘, ‚&‘ or comma.

2.2.3.2.7. Select the icon for displaying a message (df_msg_icon)

Parameter name	Meaning
Number of the function key	With this message you can specify the icon of the function key to be used when displaying a message. If provided, the icon associated in the design is being used, otherwise the device uses the built-in icon.

Example:

```
   df_msg_icon=2
```

Show the icon associated the F2 function (typically „leaving“).

```
   df_msg=Message\rwith%20Icon,5,1,0&df_msg_icon=4
```

This message sets the icon to be to the one associated to function F4 and displays the message using this icon.

2.2.3.2.8. Define a backlight (df_backlight)

You can instruct the device to show a specific backlight color in response to a dataset record. For this, you may send the `df_backlight` message.

Parameter name	Meaning
Backlight ID	0: Transponder 1: Logo 2: Touch-Keyboard (EVO 4.3) 3: User defined 4: Fingerprint
Delay	Duration in seconds, for that the backlight is set. 0 = infinite
Intensity red	Value between 0 (no red light) and 255 (full red light)
Intensity green	Value between 0 (no green light) and 255 (full green light)
Intensity blue	Value between 0 (no blue light) and 255 (full blue light)
Intensity white	Value between 0 (no white light) and 255 (full white light)

Remark:

- Not every devices has each of the above described backlights equipped (e.g. the EVO 4.3 is the only with a touch keyboard backlight currently)
- Not all backlights are full-color backlights (e.g. the touch keyboard backlight of the EVO 4.3 is a white-only backlight)

Example:

```
   df_backlight=0,5,255,255,0,192
```

sets the transponder’s backlight to show bright (192) yellow (255, 255, 0) for five seconds.

2.2.3.2.9. Setting the background message (df_info_msg)

<table>
<tr><th>Parameter name</th><th>Meaning</th></tr>
<tr><td>Message</td><td>Message to be displayed on the device’s screen. A linefeed can be added into the message by sending a „\r“.</td></tr>
<tr><td>Font</td><td>Specifies the font size and style.
<table>
<tr><td>0</td><td>Standard font</td></tr>
<tr><td>1</td><td>16 pixel (7 lines)</td></tr>
<tr><td>2</td><td>16 pixel, fixed width (7 lines)</td></tr>
<tr><td>3</td><td>19 pixel (6 lines)</td></tr>
<tr><td>4</td><td>19 pixel, fixed width (6 lines)</td></tr>
<tr><td>5</td><td>21 pixel (5 lines)</td></tr>
<tr><td>6</td><td>21 pixel, fixed width (5 lines)</td></tr>
</table>
! <b>Attention:</b>
The pixel values indicated in the table and lines are approximations and may vary depending on the device used.</td></tr>
</table>

The set the background message, please add the following section into the http Response

```
df_info_msg=<Message>,<Font>
```

e.g.

```
df_info_msg= This\ris\ra\rMessage,0
```

! **Attention:**
Please be aware or the entity representation rules of ‚=‘, ‚?‘, ‚&‘ or comma.

2.2.3.2.10. Online function of the access control (df_ac2)

Parameter name	Meaning
Module	The value of the string, the format of the field "TM" of the "reader" list. Therefor it must consist out of 3 digits always.
Master	Id for the RS485 bus AC; AC describes the bus strand. RS485 bus ID 1 RS485 bus ID 2 etc. You need to specify ‘Master’ along with ‘Module’.

<table>
<tr><td>Mask</td><td>Values mask for the device specified. Sum up the individual bit values. If you – for example - want to turn on the red LED and the relay 3, then passed by value "4 + 32 = 36".
<table>
<tr><th>Value / Bit</th><th>Description</th></tr>
<tr><td>1 / 1</td><td>If the bit is set, the buzzer is activated.</td></tr>
<tr><td>2 / 2</td><td>If the bit is set, the green LED is addressed.</td></tr>
<tr><td>4 / 3</td><td>If the bit is set, the red LED is activated.</td></tr>
<tr><td>8 / 4</td><td>If the bit is set, the 1st relay is addressed.</td></tr>
<tr><td>16 / 5</td><td>If the bit is set, the 2nd relay is addressed.</td></tr>
<tr><td>32 / 6</td><td>If the bit is set, the 3rd relay is addressed.</td></tr>
<tr><td>64 / 7</td><td>If the bit is set, the 4th relay is addressed.</td></tr>
<tr><td>128 / 8</td><td>If the bit is set, the yellow LED is addressed (from 04.03.16) / 5th relay (until 04.03.15) addressed.</td></tr>
<tr><td>256 / 9</td><td>If the bit is set, the 5th relay (from 04.03.16) / 6th relay (until 04.03.15) is addressed.</td></tr>
<tr><td>[from 04.03.16]</td><td></td></tr>
<tr><td>512 / 10</td><td>If the bit is set, the 6th relay is addressed.</td></tr>
<tr><td>1024 / 11</td><td>If the bit is set, the 7th relay is addressed.</td></tr>
<tr><td>2048 / 12</td><td>If the bit is set, the 8th relay is addressed.</td></tr>
<tr><td>4096 / 13</td><td>If the bit is set, the 9th relay is addressed.</td></tr>
<tr><td>8192 / 14</td><td>If the bit is set, the 10th relay is addressed.</td></tr>
<tr><td>16384 / 15</td><td>If the bit is set, the 11th relay is addressed.</td></tr>
<tr><td>32768 / 16</td><td>If the bit is set, the 12th relay is addressed.</td></tr>
<tr><td>65536 / 17</td><td>If the bit is set, the 13th relay is addressed.</td></tr>
<tr><td>131072 / 18</td><td>If the bit is set, the 14th relay is addressed.</td></tr>
<tr><td>[from 04.03.18]</td><td></td></tr>
<tr><td>… / 19</td><td>If the bit is set, the 15th relay is addressed.</td></tr>
<tr><td>… / 20</td><td>If the bit is set, the 16th relay is addressed.</td></tr>
<tr><td>… / 21</td><td>If the bit is set, the 17th relay is addressed.</td></tr>
<tr><td>… / 22</td><td>If the bit is set, the 18th relay is addressed.</td></tr>
<tr><td>… / 23</td><td>If the bit is set, the 19th relay is addressed.</td></tr>
<tr><td>… / 24</td><td>If the bit is set, the 20th relay is addressed.</td></tr>
<tr><td>… / 25</td><td>If the bit is set, the 21st relay is addressed.</td></tr>
<tr><td>… / 26</td><td>If the bit is set, the 22nd relay is addressed.</td></tr>
<tr><td>… / 27</td><td>If the bit is set, the 23rd relay is addressed.</td></tr>
<tr><td>… / 28</td><td>If the bit is set, the 24th relay is addressed.</td></tr>
<tr><td>… / 29</td><td>If the bit is set, the 25th relay is addressed.</td></tr>
<tr><td>… / 30</td><td>If the bit is set, the 26th relay is addressed.</td></tr>
<tr><td>… / 31</td><td>If the bit is set, the 27th relay is addressed.</td></tr>
<tr><td>… / 32</td><td>If the bit is set, the 28th relay is addressed.</td></tr>
</table></td></tr>
<tr><td>State / Function</td><td>State which will be approved by the specified units.
<table>
<tr><td>0</td><td>Off</td></tr>
<tr><td>1</td><td>On</td></tr>
<tr><td>2</td><td>Toggle (600ms on, 600ms off)</td></tr>
<tr><td>3</td><td>3 times turn for 500ms</td></tr>
</table></td></tr>
<tr><td>Duration</td><td>[applies only for state/function = 1]: The duration, for that the mask is applied.</td></tr>
</table>

	Interpretation: - 0 = always on - 1 - 40 = duration in seconds, for that the mask is active

To trigger an access control action, add the parameter

```
   df_ac2=<Modul>,<Master>,<Mask>,<State>,<Duration>
```

to your server’s http response.

2.2.3.2.11. Sending a custom message to an access control device (df_custom_msg_ac2) [in preparation]

In order to send data directly to a device in the access control system, the http command df_custom_message_ac2 may be used. If the access controller receives this command it will relay it to the references module directly.

In order to enable the transfer of binary data, data may be encoded hexadecimally during the HTTP transfer. If this encoding is selected, the access controller combines every pair of subsequent characters into one byte – the resulting binary data array is sent to the access control device.

Parameter name	Meaning
Module	The value of the string, the format of the field "TM" of the "reader" list. Therefor it must consist out of 3 digits always.
Master	Id for the RS485 bus AC; AC describes the bus strand. RS485 bus ID 1 RS485 bus ID 2 etc. You need to specify ‘Master’ along with ‘Module’.
Function	Code for the function to be executed by the access controller 1 = Relay message to specified device
Encoding	0 = ASCII (URL encoded) 1 = Hexadecimal
Data	Data conforming above encoding.

Example:

```
   df_custom_msg_ac2=010,1,1,1,48616c6c6f2057656c74
   df_custom_msg_ac2=010,1,1,0,Hallo%20Welt
```

Both instructions send the text “Hallo Welt” (German for “Hello World”) to the access control device 010 at access control bus 1.

2.2.3.2.12. Pre-checked online access control (df_ao_ac2)

In the context of pre-checked online the AC controller computes the action it would perform in offline mode. The decision is the sent to the server as a dataset. The AC controller waits for acceptance or denial of the action.

The data is build according to normal AC datasets. The server's response is expected in the following form:

Parameter name	Meaning
Mode	Decision of the server. • 0 = reject • 1 = accept
Group	ID of a group. The group is required for granting access if the transponder id was not present in the identification list – if the action2 list requires a group id.

Example:

```
df_ao_ac2=1
df_ao_ac2=0,1234
```

2.2.3.2.13. Trigger access control event by server (df_trigger_ac2)

This command can be used to trigger an AC2 action. The behaviour is equivalent to a transponder being present to an RFID reader. For this, the reader and transponder ids have to be sent to the access control master. The reader has to be connected to the AC master as well – the reader id has to be taken from the reader AC2 list used by the master.

Parameter name	Meaning
Access Master ID	Defines the bus to be used. Corresponds to the ZM column from the Reader list.
Door Module ID	Address of the reader device within the selected bus. Corresponds to TM column from the Reader list.
Transponder ID	The value to be simulated as having been read
Pin	Pin code for access control processing. The content should correspond to the Pin column from the Identification list.

Example:

```
df_trigger_ac2=1,011,876543210,0
```

2.2.3.2.14. Fetch key-value pair from the device (df_kvp)

Some data being present on a device is not provided to the server by means of a data set. To collect there information, the server can request this data to be sent as a key-value-pair by sending a "df_kvp" request to the device. The following data may be accessed in this manner:

- Current Time and data	[time],
- The device serial number	[serialnumber],
- The firmware version	[firmwareversion],
- Setup or global variable	[var],
- State one or more relais	[relais],

- State of the flash memory	[flashstate] and
- Configuration Information	[info]
- Device Hardware Information	[hw]

Sending a „df_kvp“ request to the device requires a token (in brackets in above enumeration). Some of the tokens require additional parameter:

Token-Name	Parameter	Result (Example)
firmwareversion	---	kv=firmwareversion,\ 04.03.15.05.EVO35
flashstate	---	
info		The following information is sent: - serial number kv=serial-number,1234 - device type kv=device,11 - Setup kv=setup <fn>,<CRC> kv=setup4 <fn>,<CRC> - certificates kv=cert <fn>,<CRC> CRC is a 32-Bit CRC in hexadecimal representation. Please check Appendix C.2 for details on the CRC computation.
relais	ID of the relay to be reported, starting at 1. If omitted, the states of all relais are being sent.	kv=relais1,open
serialnumber	---	kv=serialnumber,1234
time	---	kv=time,2020-09-17T07:00:00
var	Name of the variable	kv=var Ausweis,876543210

hw	---	kv=board,50006,4.7a& (MB)
		kv=module,29,1.5c,14& (PoE)
		kv=module,37,1.4c,6& (Ser)
		kv=module,102018,1.2b,6.1& (TP)
		kv=module,11,1.5a,8& (Eth)
		kv=module,30,1.0b,11& (Gfx)
		kv=module,110003,1.1c,11.1 (Dis)
		Art.No. Version Location

extinfo
 	
extinfo,ac, \
<module>, \
<master>
	---	kv=serialnumber,8675&kv=device,23& (info)
		kv=setup EVO 2.8.aes,0x12345678& (obs)
		kv=setup4 EVO 2.8.aes,0x12345678&
		kv=cert test.cert,0x087654321&
		
		kv=firmwareversion,04.03.15.05.EVO35& (fw)
		
		kv=board,50006,4.7a& (hw)
		kv=module,29,1.5c,14&

		`kv=module,37,1.4c,6&`
		`kv=module,102018,1.2b,6.1&`
		`kv=module,11,1.5a,8&`
		`kv=module,30,1.0b,11&`
		`kv=module,110003,1.1c,11.1`

> ☞ **Note:**
> The “info” telegram contains two CRC values. The CRC value “setup” is computed for internal data structures, the value “setup4” is computed for the original AES file that has been transported using the HTTP/HTTPS API. The CRC algorithm is described in Appendix “**C.2: CRC Implementation of the info telegram**”.
>
> If the setup has been uploaded to the device using the Datafox Studio directly, the “setup4” value will not be present in the “info” telegram.

Example (Request):

```
df_kvp=serialnumber&df_kvp=var,Transponder&df_kvp=relais1&df_kvp=re-
lais2
```

The computed result is send as a GET request to the server. The request does contain a df_type=kvp field rather than a df_table value. The key-value-pairs will be sent as comma separated values.

The answer to above query will look as follows:

Part of the URL	Explanation
`http://<host>:<port>/<base-url>?`	
`    df_api=1&`	
`    df_type=kvp&`	No dataset – key-value-pairs!
`    kv=serialnumber,2045&`	Serial number
`    kv=var Transponder,876543210&`	Global variable: „Transponder“
`    kv=relais1,open&`	Relay 1 opened
`    kv=relais2,closed`	Relay 2 closed

2.2.3.2.15. Set or reset a non-access control relay (df_set_relay)

Relays assigned to the access control may be set or reset in online mode of the access control explicitly. For all other relays you may use the df_set_relay command to manipulate them:

Parameter name	Meaning
Relay-ID	ID of the relay to be manipulated, starting at 1
State (directly) after this command	Use codes „close“ or „open“ here.
Duration	Duration in seconds (1 up to 60 seconds, 0 for infinite)

Example:

`df_set_relay=2,close,5` (closes relay 2 for 5 seconds, then it is opened)

2.2.3.2.16. Toggle a non-access control relay (df_toggle_relay)

Alternatively you may toggle a relay for a specific duration.

Parameter name	Meaning
Relay-ID	ID of the relay to be manipulated, starting at 1
Duration	Duration in seconds (1 up to 60 seconds, 0 for infinite)

Example:

`df_toggle_relay=1,2` (switch relay 1 for a duration of 2 seconds)

2.2.3.2.17. Load a file from server to the device (df_load_file)

Using this function you instruct the device to download a transfer file from the server. The encoding of a transfer file is described in Appendix A. Upon successful download of the file the device checks the integrity of the file before applying it.
If this check fails, the entire transfer file is being ignored. System messages according are being generated – provided they are active (see section 2.3.1).

Example:

```
df_load_file=/path/on/server?with=optional&server=parameters
```

☞ **Note:**
There are – as described in Appendix A.3.2 – different content types inside the IFF file. These types determine how an IFF file is applied to or processed by the device. If you want to transfer a file directly to a device's flash filesystem, please use type 0xDF00.

☞ **Note:**
If you want to send a single file in response to a request, you may send the IFF file directly as HTTP response (see 2.2.1.2)

! **Caution:**
This function may be used to abuse the device's storage. A device with a filled up filesystem will not work as expected – you may render it unusable in this way.

2.2.3.2.18. Transfer data from a device to the server (df_send_file)

This function allows sending specific file data from a device to the server.

- The target upload form on the server is passed as first parameter to the df_send_file command
- The data to be sent is passed along with the upload request as a token – which may require additional parameters separated by commas.

Parameter name	Meaning
Target path	Path on the server, where the file requested is uploaded to
Upload content token	See next table on tokens
Optional parameters	Additional parameters as defined by next table

Token-Name	Parameter name	Meaning
finger	<PID>,<FID> or <PID>,all or all	Transfer a single fingerprint template, the fingerprint template of a single person or all fingerprint templates. ☞ **Note:** This function is currently available only for the optical fingerprint sensor „Saturn 01“.
finger2	<format>,<PID>,<FID> or <format>,<PID>,all or <format>,all	Similar to „finger“. <format> may be chosen to be either - 1 (0xDF0E) or - 2 (0xDF18)
flash	<Filename>	Reads a file from the device’s filesystem and sends it to the server.
list	setup,<list name> ac2,<list name>	Sends a setup list or a list from the access control system to the server. The list is identified by its <list name>.
syslog	<restore-flag>	Transfers the device’s system log to the server. If the <restore-flag> is set to 1, the device reconstructs the whole log present on the device. ☞ **Note:** The system log is prepared by the device and afterwards transferred to the server. Preparing the system log may consume roughly 2 minutes, if the device has not been read for a longer period. A syslog upload file may be up to 512 kB in size.
setup	---	Transfers the setup currently being used by the device to the server.
structure	record setup	Transfer the dataset or list structure of the current setup to the server.

	ac2	
dir	user root image *[not yet impl.]*	Transmits a list of files with absolute path information currently stored inside the device memory. The different files are separated by \r\n.
file *image*	*All [not yet impl.]* single *single,<image name> [not yet impl.]*	Send Image or big barcode content: *Send all files [not yet impl.]* The next file (in chronological order) is being send *The file with filename <image name> is being send. [not yet impl.]*
hip	---	*[not yet implemented]* *Transfers the device's HIP data block to the server. This data is transmitted encrypted.*
language		*[not yet implemented]*
network		*[not yet implemented]*

Example:

```
   df_send_file=/upload-form.html,flash,root:datafox.cert
   df_send_file=/setup-storage-form.pl,setup
   df_send_file=/fingerprint-backup-form.asp,finger,1980,all
   df_send_file=/list-data-form.js,list,ac2,Identification
   df_send_file=/device-logs-form.cgi,syslog
   df_send_file=/list-desc/,structure,record
```

2.2.3.2.19. Remove a file stored on the device (df_remove_file)

With this function you may remove a file stored on the device. To erase a file, you will have to specify the file name as parameter.

Instruction	Meaning
Path	Specifies the storage location of the file to be removed on the device. The storage location has to start by „user:“ or „root:“ and is followed by the file path / name.

Example:

```
   df_remove_file=root:datafox.cert
```

2.2.3.2.20. Delete fingerprint data from the device (df_remove_finger)

To remove fingers from the built-in fingerprint sensor of the MasterIV-device, you may use the df_remove_finger command. You can choose to remove a single finger, all fingers of a person or all fingers from the sensor.

The following instructions are available:

Instruction	Meaning
df_remove_finger=<PID>,<FID>	Remove fingertemplate <FID> of person <PID>
df_remove_finger=<PID>,all	Remove all fingertemplates of person <PID>
df_remove_finger=all	Clear the entire module

Examples:

```
df_remove_finger=1980,all
df_remove_finger=1980,6
```

(Finger 6 is the thumb at the right-hand)

☞ **Please note:**
This function is available only for the optical finger print sensor „Saturn 01“.

2.2.3.2.21. Update of list data (df_setup_list or df_ac2_list)

The files of the lists are accessed using the same HOST name to which the request was made.

! **Danger:**
The data provided for lists has to **be sorted according to the criterion specified as search condition** in the setup with sort column. Access control lists are sorted in lexicographical order based on the first column; other lists defined by the setup in lexicographical order according to their sort column.
List data supplied in a different sorting will be accepted by the client and will be handled as unsorted data – which may result in long data access time.

Parameter name	Meaning
Name	Name of the list description.

Path	Path including file name of the file to be downloaded.
	Example:
	`df_setup_list=HR,/path/to/liste.txt`
	oder
	`df_ac2_list=Identification,/path/to/liste.txt`

Please be aware, that lists in the setup and the access control module may have the same names. You have to select the right location, where the list is stored by the parameter name you are sending.

2.2.3.2.22. Count number of records within a table (df_table_count)

Returns the number of records stored inside a table. These values are differentiated into

- The total number of active table rows (rows, that are not marked as having been deleted),
- The number of lines added to the table without being sorted,
- The number of deleted lines.

Parameter name	Meaning
Table/List name	LIST.<name> or ACCESS.<name>

The result is sent as a KVP records (see 2.2.3.2.14) with following structure:

URL	Description
`http://<host>:<port>/<base-url>?`	
`df_api=1&`	
`df_type=kvp&`	No data record – a key value pair is being sent.
`kv=table,list.PID&`	Name of the table for which statistical data is being sent.
`kv=count,220&`	The table contains 220 active entries.
`kv=appended,12&`	There are 12 rows that are appended to the table.
`kv=deleted,18`	Within the table there are 18 rows marked as having been deleted.

! **Attention:**
The performance for accessing a table’s data will deteriorate as more and more data is either being deleted or appended in an unsorted way. If the access times are not good enough anymore, consider reading the table’s data, sorting it and rewriting the table to the device.

2.2.3.2.23. Select a row from a table (df_table_select)

This command allows retrieving list data – or a part thereof – and allows sending it to the server.

Parameter name	Meaning
Table/List name	LIST.<name> or ACCESS.<name>
Server path	Path on the server where the file containing the data is to be up-loaded to.
Fist filter criterion	<Column name>=<Value>

Second filter criterion	<Column name>=<Value>

Examples (the table PID consists out of four columns: PID, Name, Department ("Abteilung"), Date ("Datum"):

```
df_table_select=list.PID,/upload/form
df_table_select=list.PID,/upload/form,Abteilung%3dEntwicklung
df_table_select=list.PID,/upload/form,Abteilung%3dEntwicklung,PID%3d5
```

Above selection provide

- all list entries from the PID table,
- all list entries where the Department is set to „Entwicklung"
- all list entries where the Department is set to „Entwicklung" and the PID is 5

2.2.3.2.24. Append data to a table (df_table_append)

Append a single line to a table. For sorted table the entry is appended in an unsorted section of the table. It is required to send all table fields as a comma separated list.

Parameter name	Meaning
Table/List name	LIST.<name> or ACCESS.<name>
Data records	The data to be appended to the list.

Examples (the table PID consists out of four columns: PID, Name, Department ("Abteilung"), Date ("Datum"):

```
df_table_append=list.PID,5,Sven%20Meyer,Entwicklung,
df_table_append=list.PID,9999,,Besucher,
```

The operation will be confirmed by sending a system message that informs on success or error when appending the data row.

! **Attention:**
The performance for accessing a table's data will deteriorate as more and more data is appended. If the access times are not good enough anymore, consider reading the table's data, sorting it and rewriting the table to the device.

2.2.3.2.25. Update a table (df_table_update)

Parameter name	Meaning
Table/List name	LIST.<name> or ACCESS.<name>

Fist filter criterion	<Column name>=<Value>	
Second filter criterion	<Column name>=<Value>	
First column to be changed	<Column name>=<Value>	Values from maximally 4 columns may be changed in this way.
Optional additional columns to be changed	<Column name>=<Value>	

Examples (the table PID consists out of four columns: PID, Name, Department (“Abteilung”), Date (“Datum”):

```
df_table_update=list.PID,,,Abteilung%3d
df_table_update=list.PID,Abteilung%3dEntwicklung,,Abteilung%3dDevelopment
df_table_update=list.PID,Abteilung%3dEntwicklung,PID%3d5,Datum%3d2019-09-06T06:23:00
```

The first instruction clears the “Abteilung”-column from the PID table.

The second instruction changes the “Abteilung” column’s value to “Development” wherever the column contained “Entwicklung” previously.

The third instruction changes the “Datum” column’s value to 06:23:00 on September 6th 2019 for all rows where PID column has the value 5. You may change the values of up to four columns by a single df_table_update statement.

! **Attention:**
The performance for accessing a table’s data will deteriorate as more and more data is being deleted and appended. Internally the rows selected are marked as deleted and new rows are appended to the table.
If the access times are not good enough anymore, consider reading the table’s data, sorting it and rewriting the table to the device.

2.2.3.2.26. Delete data from a table (df_table_delete)

Parameter name	**Meaning**
Table/List name	LIST.<name> or ACCESS.<name>
Second filter criterion	<Column name>=<Value>
First column to be changed	<Column name>=<Value>

Examples (the table PID consists out of four columns: PID, Name, Department (“Abteilung”), Date (“Datum”):

```
df_table_delete=list.PID
df_table_delete=list.PID,Abteilung%3dEntwicklung
df_table_delete=list.PID,Abteilung%3dEntwicklung,PID%3d5
```

Similar to the `df_table_select` command, this command deletes the rows matched by the filter criteria. Above examples delete

- All rows from the table PID,
- All rows from the table PID where the column „Abteilung“ contains the value „Entwicklung“
- All rows from the table PID where the column „Abteilung“ contains the value „Entwicklung“ and „PID“ contains the value „5“.

> ☞ **Please note:**
> Remove a row from a sorted list does not have a significant impact on the lookup performance.

2.2.3.2.27. Firmware Update using https (df_load_firmware)

The command “df_load_firmware” is one of three ways that may be used to deploy a firmware update using HTTP to a device (please check Appendix E).
In this process the server informs the device on the firmware package as well as the firmware file to be downloaded.

Examples:

```
df_load_firmware=04.03.19.21.dfz,evo3.5_04.03.19.21.iff
df_load_firmware=04.03.19.21.dfz,evo_intera_II_49004_04.03.19.21.iff
```

The device creates an HTTP request for accessing the firmware update endpoint defined by the system variables `http.update.host`, `http.update.port` und `http.update.send`. Depending on the device’s communication settings this request is using HTTP or HTTPS.

Example:

GET https://update.host:443/update-server-send-path/04.03.19.21.dfz/evo3.5_04.03.19.21.iff

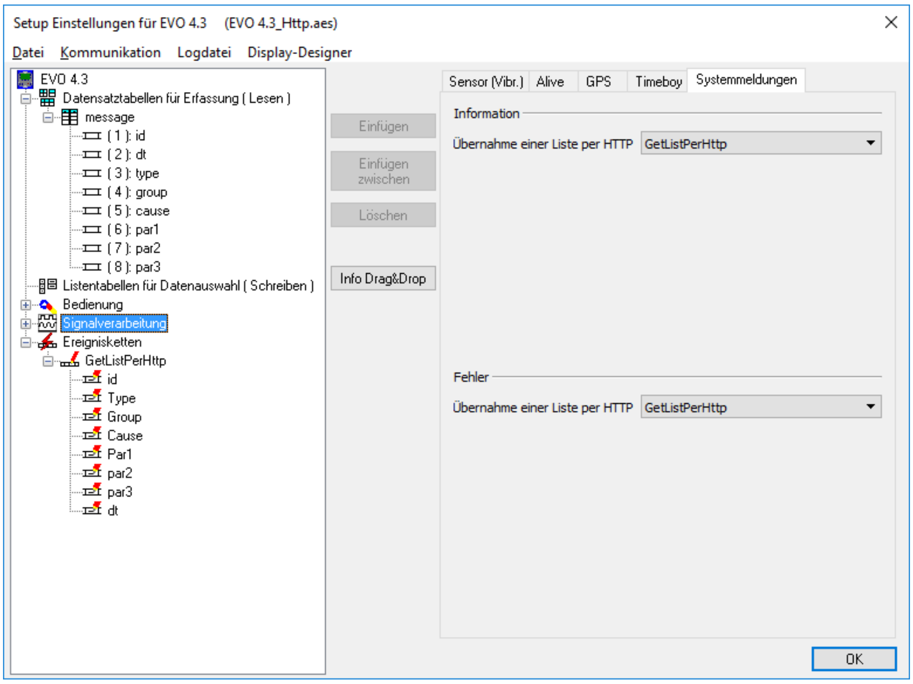
> ☞ **Please note:**
> The update server is now responsible to fetch and deliver the content of the requested IFF file. This might be achieved by using a “standard” webserver that offers access to the files from the extracted DFZ packages.

2.3. Feedback through System Messages

Independent of you using the DFCom interface or the http API, the device can inform you on the outcome of actions.

The result is provided by means of system messages. This chapter shows the configuration of system messages in the context of list data transfer.

System messages are configured in the section ‘signal processing’ of the setup.


Setup Einstellungen für EVO 4.3 (EVO 4.3_Http.aes)

Datei Kommunikation Logdatei Display-Designer

- EVO 4.3
  - Datensatztabellen für Erfassung (Lesen)
    - message
      - (1): id
      - (2): dt
      - (3): type
      - (4): group
      - (5): cause
      - (6): par1
      - (7): par2
      - (8): par3
  - Listentabellen für Datenauswahl (Schreiben)
  - Bedienung
  - Signalverarbeitung
  - Ereignisketten
    - GetListPerHttp
      - id
      - Type
      - Group
      - Cause
      - Par1
      - par2
      - par3
      - dt

Einfügen | Einfügen zwischen | Löschen | Info Drag&Drop

Sensor (Vibr.) | Alive | GPS | Timeboy | Systemmeldungen

Information

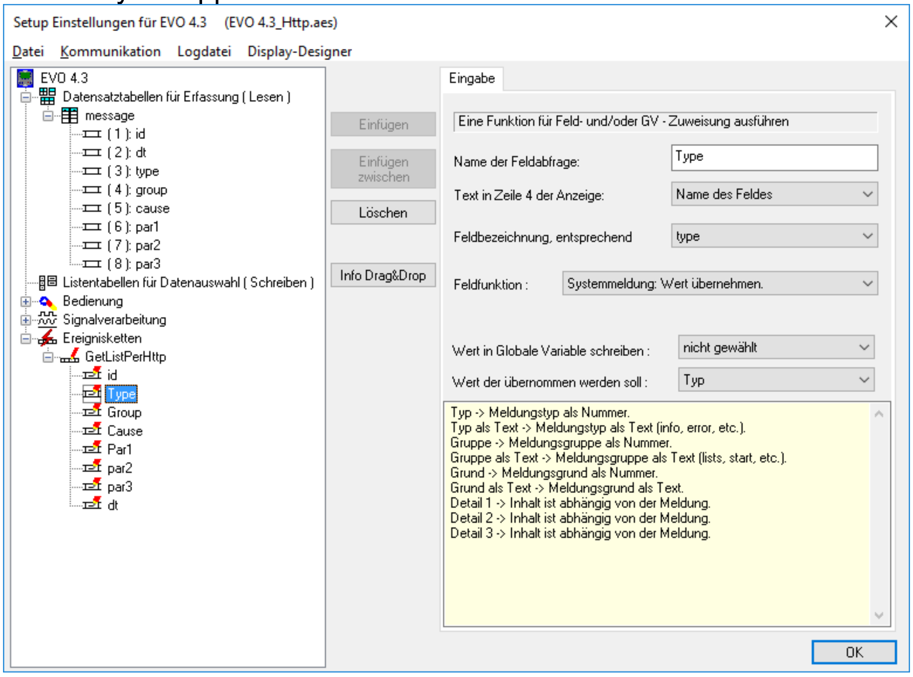
Übernahme einer Liste per HTTP: GetListPerHttp

Fehler

Übernahme einer Liste per HTTP: GetListPerHttp

OK

The assigned sequence of events triggers the creation of a data records at the device. These data records are provided to your application.


Setup Einstellungen für EVO 4.3 (EVO 4.3_Http.aes)

Datei Kommunikation Logdatei Display-Designer

- EVO 4.3
  - Datensatztabellen für Erfassung (Lesen)
    - message
      - (1): id
      - (2): dt
      - (3): type
      - (4): group
      - (5): cause
      - (6): par1
      - (7): par2
      - (8): par3
  - Listentabellen für Datenauswahl (Schreiben)
  - Bedienung
  - Signalverarbeitung
  - Ereignisketten
    - GetListPerHttp
      - id
      - Type
      - Group
      - Cause
      - Par1
      - par2
      - par3
      - dt

Einfügen | Einfügen zwischen | Löschen | Info Drag&Drop

Eingabe

Eine Funktion für Feld- und/oder GV - Zuweisung ausführen

Name der Feldabfrage: Type

Text in Zeile 4 der Anzeige: Name des Feldes

Feldbezeichnung, entsprechend: type

Feldfunktion : Systemmeldung: Wert übernehmen.

Wert in Globale Variable schreiben : nicht gewählt

Wert der übernommen werden soll : Typ

Typ -> Meldungstyp als Nummer.
Typ als Text -> Meldungstyp als Text (info, error, etc.).
Gruppe -> Meldungsgruppe als Nummer.
Gruppe als Text -> Meldungsgruppe als Text (lists, start, etc.).
Grund -> Meldungsgrund als Nummer.
Grund als Text -> Meldungsgrund als Text.
Detail 1 -> Inhalt ist abhängig von der Meldung.
Detail 2 -> Inhalt ist abhängig von der Meldung.
Detail 3 -> Inhalt ist abhängig von der Meldung.

OK

Using the field function "System message: Accept value" you can select the data to be sent back to the server.

☞ **Please note:**
In order to identify the device sending the acknowledgment, you have various options.
Here is a small selection:

- In the event chain you can set the device name and the serial number of the device as Id.

	- You can also provide a global variable on the response which is sent back as the field value in the generated system messages. This way of creating a session id, was also used in the above example.

2.3.1. System messages to the list data download (feedback - records)

Some of the following system message values can occur via HTTP during the download of list data:

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
General results			
0	100 (http)	1	http action completed successful, e.g. List data has been applied.
	- / - / -		
1	100	2	Generic error
	- / - / -		
2	0	1	Most recent command was successful
	- / - / -		
3	0	2	The most recent server command contained unsuccessful commands – for these individual system messages are created.
	- / - / -		
List data processing			
1001	100	2	Invalid parameter
	- / - / -		
1002	100	2	Unknown List
	Unknown List / <List name> / <List type>		
1003	100	2	Parameter missing
	- / - / -		
1004	100	2	Error in list line
	<Details on the error> / <List name> / -		
1005	100	2	List will be ignored (due to other errors)
	- / - / -		
1006	100	2	Duplicate list
	- / - / -		
1007	100	2	An update is currently pending. You list update therefor cannot be applied (please retry later) (obsolete, not used anymore)

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
	- / - / -		
1008	100	2	Encoding is not supported. With firmware 04.03.10.xx only “ISO-8859-1” and binary encoding (missing encoding chunk) are support. (obsolete, not used anymore)
	- / - / -		
1009	100	2	The CRC inside of the transfer file is wrong. The index of the FORM with the wrong CRC and the filename are available through the status message. (obsolete, not used anymore)
	- / - / -		
1010	100	2	The parameter is not supported for http upload yet. Detail 2 contains the parameter causing the problem.
	Command unknown / <Parameter> / -		
1011	100	1	The server requested update of access control lists, but access control is deactivated
	no access control / <List name> / <List typ>		
1012	100	2	The IFF file to be uploaded could not be created. Detail 2 contains the name of the file requested by the server, Detail 3 the temporary filename created to perform delivering the data.
	<Detail1> / <Detail2> / <Detail3>		
1013	100	2	The parameters supplied are invalid. Detail 1 contains a detailed description, Detail 2 and 3 may contain the offending parameters.
	<Detail1> / <Detail2> / <Detail3>		
1014	100	1	A file upload request has been received and is being processed. Detail 2 contains the request details.
	Received an upload request / <File type> / -		
1015	100	2	Error during upload of the IFF file.
	HTTP-Upload failed. Could not connect to server / <file name> / <path>		
1016	100	1	File has been transferred successfully to the server, the server acknowledged the last chunk sent.
	HTTP-Upload finished - response ok / <file name> / <path>		
1017	100	2	File has been transferred successfully to the server. The server did not acknowledge the last chunk sent.

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
			In order to acknowledge the upload, the server shall send HTTP status code 200 and an **empty** body.
	HTTP-Upload finished - response error / <file name> / <path>		
1018	100	2	During an update no other update can be processed.
	Update already in progress / - / -		
1019	100	2	List type is unknown.
	There are no lists with this type / <List name> / <Type>		
1020	100	2	List (name) is unknown.
	Unkown list / <List name> / <Type>		
1021	100	2	Fingerprint, PID invalid.
	Error reading PID / <PID> / -		
1022	100	2	Fingerprint, FID invalid.
	Error reading FID / <FID> / -		
1023	100	1	File download was successful.
	Download ok / <file name> / <path>		
1024	100	2	Error during file download.
	Download error / <file name> / <path>		
1025	100	1	An IFF file has been scheduled to be uploaded.
	IFF-file added to upload / <Filename> / <Path>		
1026	100	2	Insufficient memory for processing the request.
	- / - / -		
1027	100	2	No setup is run by the device currently. Please Note: The device received a `df_send_file` request to send the setup to the server. The device cannot fulfil this request, since the device was not transferred using http.
	No valid setup / - / -		
1030	100	1	File download successful. (obsolete, not used anymore)
	HTTP-Download OK / <Name> / <Path>		
1031	100	2	File download failed. (obsolete, not used anymore)

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
	HTTP-Download Error / <Name> / <Path>		
1032	100	2	List data transfer: Too many columns in row.
	Line <LineNo> / <Data> / <Error code>		
1033	100	2	List data transfer: Not enough columns in row.
	Line <LineNo> / <Data> / <Error code>		
1034	100	2	List data transfer: Column content too large to fit into list record.
	Line <LineNo> / <Data> / <Error code>		
1035	100	2	List data transfer: Odd number of data bytes while transferring HEX ASCII data.
	Line <LineNo> / <Data> / <Error code>		
1036	100	2	List data transfer: Transfer of HEX ASCII data contained an invalid character (that is non HEX ASCII)
	Line <LineNo> / <Data> / <Error code>		
1037	100	2	List data transfer: Sending access control lists to device with disabled access control module has been ignored.
	Ignore access list / <Name> / -		
1038	100	2	No setup file present at the device.
	- / - / -		
1039	100	2	Http header too large
	http Header too large / <headerSize> / <maxSize>		
Processing IFF files			
1500	150	1	Setup uploaded to the device successfully. Please note: This system message is created using the new setup – provided, system messages are activated there.
	Setup installed / <Name des Setups> / -		
1501	150	2	Error while reading data. Detailed information is supplied as Detail 1 and 2.
	<Error description> / <Details concerning the error> / -		
1502	150	2	Error while writing data. Detailed information is supplied as Details 1 and 2.
	<Error description> / <Details concerning the error> / -		

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
1503	150	2	The setup does not match the type of the device.
	AES file not valid on this device / <Device Type Name> / -		
1504	150	2	Error while checking the IFF's CRC
	<Type of the IFF file> / - / -		
1505	150	2	Version of IFF file is not supported
	<Type of the IFF file> / - / -		
1506	150	2	The selected encoding is not supported by the device firmware
	<Type of the IFF file> / - / -		
1507	150	2	Wrong encoding of IFF file
	<Type of the IFF file> / - / -		
1508	150	2	Error saving IFF file (while creating)
	<Type of the IFF file> / - / -		
Routing			
1800	180	1	Entry to be applied found
	Routing destination found / <Name of routing file> / <Address addition>		
1801	180	1	Entry to be relayed found
	Routing forward file / <Name of routing file> / <Received Address>		
1802	180	1	Routing file received
	Routing info received / <Name of routing file> / -		
1803	180	1	Entries left to be processed, which have not been processed yet
	Routing entries left / <Name of routing file> / -		
1804	180	1	Routine file completely processed. File will be deleted.
	Routing finished / <Name of routing file> / -		
1805	180	2	No valid entry found. File will be deleted.
	No valid entry / <Name of routing file> / -		
1806	180	2	Routing file could not be applied
	Update failed / <Name of routing file> / <Address addition>		
1807	180	2	Error while relaying
	Forwarding failed / <Name of routing file> / <Received Address>		

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
Fingerprint			
2001	200	2	Error while reading fingerprint templates
	- / - / -		
2002	200	2	Error while writing fingerprint templates
	- / - / -		
2003	200	2	Error while erasing a fingerprint template
	Error deleting template / - / -		
2004	200	1	Fingerprint template enrolled at sensor
	<PID> / <FID> / <Qualität>		
2005	200	1	Fingerprint template optimized by sensor
	<PID> / <FID> / -		
2006	200	1	Fingerprint template added to sensor
	<PID> / <FID> / -		
2007	200	1	Fingerprint template removed from sensor
	<PID> / <FID> / -		
2008	200	1	All fingerprint template of specified PID have been removed
	<PID> / - / -		
2009	200	1	All fingerprint templates have been removed
	- / - / -		
2010	200	2	Fingerprint is not available.
	Fingerprint is not available / - / -		
2011	200	2	Incompatible template types in device and IFF file
	Bad Template Type / <Template Type Device> / <Template Type IFF>		
Filesystem			
2500	250	1	File removed successfully.
	File successfully deleted / <file name> / -		
2501	250	2	File not found.
	File not found / <file name> / -		
2502	250	2	Unknown partition / storage location (obsolete, this code was generate up to 04.03.13)

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
	- / - / -		
2503	250	2	Error while removing the file.
	Error while deleting the file / <file name> / -		
Table Manipulation			
3000	300	2	Opening a list to access if with append, update or delete failed.
	Error opening the list / <List name> / -		
3001	300	2	Error while adding a data record to an existing list.
	Error adding list data / <Data > / -		
3002	300	2	The filter could not be applied during update or delete command.
	List selection not possible / <List name> / -		
3003	300	2	Error in parameters of select or update command.
	Parameter missing / <List name> / -		
3004	300	2	Error deleting an entire list.
	Error deleting the entire list / <List name> / -		
3005	300	2	Internal Error accessing flash memory
	Memory error / - / -		
3006	300	1	List data has been sent to the device, that is not sorted according the setup's sort criterion. This will result in a performance impact with long lists.
	Unsorted List Data / <List name> / <Line number>		
Firmware-Update using HTTP(S)			
3500	350	1	Update successful
	FW update success / <New Firmware Version> / <opt. Branch-Name>		
3501	350	2	Download of update data failed
	Server request failed / <Parameters of `df_update_firmware`> / <http.update.send>		
3502	350	2	Update-Server reports error processing the update instructions.
	<Parameters of `df_update_firmware`> / <http.update.send> / <Errorcode reported by the Update-Server>		

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
3503	350	2	Firmware does not support a hardware module built into the device.
	Module not supported / <Module Id> <Index> / <Name of Module>		
3504	350	2	Firmware for different device
	Invalid device type / <Device Type ID> / <Device-Type ID from IFF file>		
Firmware-Update using HTTP(S) / Availablility-µS			
3800	380	1	Firmware version is available
	<Name of DFZ-file> / - / -		
3801	380	2	The query sent to the availability service is not supported.
	unsupported query mode / - / -		
3802	380	2	No firmware found
	no match / - / -		
3803	380	2	Internal error while processing the query.
	internal error / - / -		
Firmware-Update using HTTP(S) / Compatibility-µS			
3900	390	1	Device is compatible with firmware
	<Name of DFZ file> / <Name of IFF file> / -		
3901	390	2	Unknown parameter when calling the service
	unhandled parameter / <Parameter> / -		
3902	390	2	The query did not contain a device type
	device type missing / - / -		
3903	390	2	The query did not contain a serial number
	serial number missing / - / -		
3904	390	2	The query did not contain a board version
	board missing / - / -		
3905	390	2	The query did not contain a firmware version to be checked
	no firmware version specified / - / -		
3906	390	2	The query did not contain hardware modules
	no hardware modules specified / - / -		

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
3907	390	2	There is no MD5 fingerprint associated to the firmware version on the server
	md5 fingerprint missing / - / -		
3908	390	2	The MD5 fingerprint associated to the firmware on the server and the one supplied by the query mismatch.
	md5 fingerprint wrong / - / -		
3909	390	2	There is no directory containing the uncompressed firmware files
	directory open error / - / -		
3910	390	2	There is no firmware IFF file associated to the device type.
	no iff file found / - / -		
3911	390	2	The Firmware IFF file’s COMP chunk has a version different from 2 – and thus unsupported.
	no acceptable compatibility info / - / -		
3912	390	2	The Aux-Chunk (FAUX) of the firmware IFF file has an unsupported version different from 1.
	no acceptable aux info / - / -		
3913	390	2	The device type id from the query does not match with a compatible device type id from the firmware IFF file.
	device type mismatch / - / -		
3914	390	2	A hardware module being part of the device is not supported by the firmware.
	unsupported hardware / <fw-idx>,<version>[,<place>] / -		
3915	390	2	The directory specified to contain firmware data does not exist.
	firmware directory not existing / <Directory> / -		
3916	390	2	Firmware Update did not manage to determine the CPU.
	failed to derive MPU / <Device-Type-Id> / -		
Change Date / Time			
4000	400	1	Date / Time has been changed successfully.
	Time changed / <Date/Tme> / -		
4001	400	2	Format error in time passed

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
	Format error / <Date/Time> / -		
4002	400	2	Internal error while setting date and time on the device.
	Internal error / - / -		
Setting system variables			
4500	450	1	The system variable has been set successfully
	Variable changed / <Name>=<Wert> / -		
4501	450	2	Error while setting the system variable
	Error when setting the variable / <Name>=<Value> / <Error code>		
4502	450	2	Error while reading a variable
	read error / <Request> / -		
4503	450	2	Error while reading a variable
	missing parameter / <Request> / -		
Setting Relays			
5000	500	1	The relay has been set
	Relais switched / <Parameter> / -		
5001	500	2	At least one parameter is invalid
	Parameter error / <Parameter> / -		
5002	500	2	Not enough parameters were supplied
	Parameter missing / - / -		
5003	500	1	Das Relais wurde erfolgreich umgeschaltet.
	Relais toggled / <Parameter> / -		
Messages (to be shown on the device’s display)			
5500	500	1	A message has been received.
	Message received / <Message Text> / -		
5501	500	2	An online data record has been set but the server did not send a message in response.
	Message missing / - / -		
5502	500	2	The message is not shown on the display – a different message is being displayed currently.
	Message ignored / <Parameter> / -		
5503	500	1	An info message has been received.

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
	Info message received / - / -		
5504	500	2	The setup is not configured with “Server Online”. Thus, not messages may be displayed.
	Online message disabled / - / -		
5505	500	2	The record to which the df_msg command was sent, was not an online record.
	Non Online Record / - / -		
5509	500	2	The IFF file does not contains data, that may be used b the device.
	No applicable data found / <filename> / -		
5510	500	1	df_msg_icon applied
	Message icon set / <Filename of Icon> / -		
5511	500	2	The image file associate to the icon is not present on the device
	Image not found / <filename> / -		
5512	500	1	The Buzzer code is invalid. The message is displayed without buzzer notification.
	Buzzer invalid / <buzzer> / -		
5513	500	2	The font size supplied is invalid. The message is shown with standard font size.
	Font invalid / <font> / -		
Audio signal			
6000	600	1	Audio signal code is known and being played
	Play Sound Sequence / <Seq.No.> / -		
6001	600	2	Unknown signal code
	Unknown Sound Sequence / <Seq.No.> / -		
Input chain			
6500	650	1	An event chain is being executed
	Execute event chain / <event chain name > / -		
6501	650	2	Unknown event chain sent by web server. Cannot execute.
	Unknown event chain / <event chain name> / -		
Service mode			

Reason	Group	Type	Description
	Detail 1 / Detail 2 / Detail 3		
7000	700	1	Service mode has been established successfully
	Service connection established / <Host:Port> / 0		
7001	700	2	Failed starting the service mode.
	Service connection could not be established / <Host:Port> / <Err-Code>		
Control of Backlights			
7500	750	1	The backlight has been configured accordingly.
	Backlight set / <Backlight-Id> / -		
7501	750	2	Backlight of supplied Id is unknown.
	Unknown backlight / <Backlight-Id> / -		

The following encodings are being used:

Type:

1:	Info – Command has been processed successfully
2:	Error

Reason:

Description of the operation

Group:

100	http module
150	IFF-Processing
180	Routing
200	Fingerprint system
250	File system
300	List-/Table Operations
350	Firmware-Update via HTTP
380	µService Firmware Availability
390	µService Firmware Compatibility
400	Change of Date/Time
450	Changing system variables
500	Switching relays
550	Processing messages to be shown on the display
600	Audio signaling
650	Input chain processing
700	Service mode
750	Backlight control

A result dataset for uploading the AC list „Action“ successfully to the device will look as follows:

URL-Teile	Bemerkung
`http://<host>:<port>/<base-url>?`	
`df_api=1&`	

`df_table=Feedback&`	
`df_col_dt=2017-03-30T13:11:06&`	
`df_col_type=1&`	Type Info
`df_col_group=100&`	Module / Group information
`df_col_cause=0&`	No error
`df_col_par1=action, 4 lines&`	Information on the data
`df_col_par2=Action&`	Name of the list being updated
`df_col_par3=`	

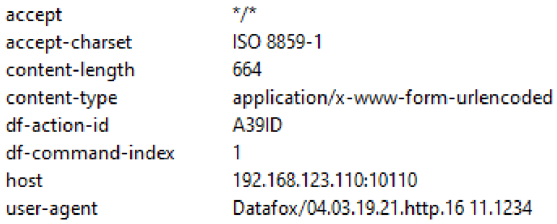
The field `df_col_par3` is reserved for future use.

2.3.2. Associating command and system messages

Since processing in the HTTP interface is asynchronous, the server may transmit an “df-action-id” along with its response. Processing the response will result in creating system message which will contain the action id – thus they can be associated to the trigger.

In order to active the “action id” for a command, the server puts a “df-action-id” header into its HTTP response. The associated value may be up to 16 characters long (e.g., “A39ID”)

The device picks up the action id and sends it along with the resulting system messages – as header field – and adds a sequence number (“df-command-index”). The command index starts at 0 for “df_api=1”, the first command is associated to index 1 accordingly.



```
accept              */*
accept-charset      ISO 8859-1
content-length      664
content-type        application/x-www-form-urlencoded
df-action-id        A39ID
df-command-index    1
host                192.168.123.110:10110
user-agent          Datafox/04.03.19.21.http.16 11.1234
```

Header associate to a system message

The following responses do not contain action id or command index:

- 1013: “Invalid sign” message on parser error
- 20xx: Messages from the fingerprint subsystem triggered by http actions
- 3500: Firmware update confirmation message

- Messages from the access control subsystem

2.4. Encryption

! **Attention:**
In the context of API-Level 1 on Hardware 4 devices, the encryption described in this section should not be used – with HTTPS there is a stronger protocol based on standardized technology available.

The data fields of the data set can be encrypted using a stream cipher RC4. The field contents are then transferred to their hexadecimal representation.

parameter name	importance
df_cb	The parameter specifies that all these fields until and including df_ce have encrypted field contents. The value of df_cb contains the four-digit (1000-9999) public key of the applicable password for the stream cipher.
df_ce	The parameter indicates that all the following fields are not encrypted any more. If the value is decrypted correctly it must match the value of df_cb.

2.4.1. Illustrate the GET request

In plain text (unencrypted) and encrypted:

Plaintext request
df_api=1&df_record_state=1&df_table=Booking&df_col_sn=2042&df_col_recordtype=1&df_col_badge=3974679390&df_col_timestamp=2017-11-22T08:23:39&df_col_status=online
Plaintext Reply
df_api=1&df_time=2017-11-22T08:24:00
Encrypted request
df_api=1&df_cb=6102&df_record_state=CC&df_table=66E9B37516AA8C&df_col_sn=0BDC8F79&df_col_recordtype=AB&df_col_badge=AF9B3A929994A5BD7D88&df_col_timestamp=B237B8CA4FA80FD563359C3EE70FE7FC99AF60&df_col_status=9BACFC1E5E0B&df_ce=A344D33B
encrypted response
df_api=1&df_cb=6102&df_time=e1ba6575855619c4d634f7865c01c4b2bc2ec138670ac2&df_ce=a414ebd6

2.4.2. Detection of encryption

To see whether the data fields are sent encrypted, the initial encryption is with ‘df_cb’ (Datafox Crypt Begin) in and with ‘df_ce’ (Datafox crypt end) in the end. ‘df_cb’ the first field in the request and ‘df_ce’ the last field in the request is.

The value of the field ‘df_cb’, itself is transmitted in plain text and is ‘public key’. It is a random number between 1000 and 9999. The value must be used in conjunction with the communication password for the encryption and decryption.

The encryption of data is thus effected by "private key + public key" as a password key.

In response, the field ‘df_cb’ must be send back with the identical value received with the request. This ensures that the decryption was successful and request and response also fit.

The value of the field ‘df_ce’ is identical to ‘df_cb’ but it is sent encryptedly. While decoding you can verify that the right key has been used. The value of ‘df_ce’ must match ‘df_cb’ after decryption is done.

If there are problems in deciphering ‘df_c=error’ has to set in the response. In addition, the fields ‘df_cb’ and ‘df_ce’ are to be filled with information as follows.

The following errors are to be observed by the evaluating script:

‘df_cb’ is not a number or is outside of its value limit of 1000 - 9999

- Answer: df_c=error&df_cb=range&df_ce=unknown/missing
  - Range describes a range error – the value is outside its limits.
  - Unknown describes an unchecked condition.
  - Missing identifies a missing field in the request.

‘df_cb’ without final ‘df_ce’

- Answer: df_c=error&df_cb=1000&df_ce=missing

‘df_ce’ is not a number or is outside of its value limit of 1000 - 9999

- Answer: df_c=error&df_cb=1000&df_ce=range

‘df_ce’ without incipient ‘df_cb’

- Answer: df_c=error&df_cb=missing&df_ce=unknown

‘df_ce’ is not equal ‘df_cb’

- Answer: df_c=error&df_cb=1000&df_ce=different
  - ‘df_ce’ (after decoding) is not equal to ‘df_cb’.

2.4.3. Response from the web server

The field contents from the request are sequentially decrypted using the RC4 stream cipher. The field contents of the reply is to be seen as part of the entire data stream and thus encrypted using the same RC4 stream cipher instance used for the decryption. The only exception is the first field value ‘dfcb’, that is identical to the one supplied by the request.
The last encrypted field of the reply has to be the ‘dfce’ field. The value of ‘dfce’ must be (after decryption) equal to the value of ‘dfcb’.

2.4.4. Buffer sizes

The Terminals provide buffers for incoming data. These buffers are sized according to the following table (04.03.10.05):

Field	**Buffer size**
http Header	1500 Bytes
http Body	2000 Bytes

Please keep in mind, that data is truncated if exceeding the buffer’s size. Typically, when using cookies, which are not processed at all by the device, the header buffer can reach its limit quickly.

Appendix A: Mapping of Communcation Libarary and Datafox Studio to HTTP Level 1

This appendix compares function provided by the communication library with those provided by the HTTP Level 1 API. Since HTTP is comparable to active mode, functions explicitly for passive mode may not be usable in this context.

Legend
🟩 Available
🟨 Planned for release
🟧 Planned for a future release – no version fixed yet
⬜ Not planned – see comment.

! **Attention:**

Setup files may be transferred using communication library or http from server to the device. The transfer from device to the server can only be done using the same means of communication. It is not possible to write a setup using the DLL and read it back using HTTP.

A.1: Comparison of communication library and HTTP Level 1

Description of function (Communication library documentation)	Communication library function name	HTTP	State / Plan
Setup serial or TCP/IP connection for MasterIV	DFCComOpenIV	Not required – the device initiates the connection	---
Close a previously openend interface	DFCComClose	Not required – the device initiates the connection	---
Check device reachability	DFCCheckAE	Not required – the device initiates the connection	---
Check device reachability	DFCCheckDevice	Not required – the device initiates the connection	---
Set device date and time.	DFCComSetTime	df_time (2.2.3.2)	04.03.10.00
Read device date and time.	DFCComGetTime	df_kvp=time	04.03.12.01
Send a message to be displayed.	DFCComSendMessage	df_msg (2.2.3.2.6)	04.03.10.01
Send a background message.	DFCComSendInfotext	df_info_msg (analog df_msg, siehe 2.2.3.2.9)	04.03.12.01
Read serial number	DFCGetSeriennummer	df_kvp=serialnumber	04.03.12.01
DFCLogOn: Do not use.	DFCLogOn (obsolete)	Will not be implemented.	---

DFCSetLogOn: Enable logging.	DFCSetLogOn (obsolete)	Will not be implemented.	---
DFCLogOff: Do not use.	DFCLogOff (obsolete)	Will not be implemented.	---
DFCSetLogOff: Disable logging.	DFCSetLogOff (obsolete)	Will not be implemented.	---
Set the callback function.	DFCSetCallBack	Realized by http server.	---
DFCSetLogFileName: Set name of logfile.	DFCSetLogFileName (obsolete)	Will not be implemented.	---
Convert error number to error text.	DFCGetErrorText	See system messages (2.3.1)	04.03.10.03
Set a global variable.	DFCSetGlobVar	df_var (for global, setup and system variables)	04.03.10.01
Read a global variable.	DFCGetGlobVar	df_kvp=var,<VAR_NAME>	04.03.12.01
Close a relay for a specific time	DFCCloseRelay	df_set_relay=<RELAIS_ID>,close,duration	04.03.12.01
Get the state of a relay.	DFCGetRelayState	df_kvp=relais,<RELAIS_ID>	04.03.12.01
Open a relay..	DFCOpenRelay	df_set_relay=<RELAIS_ID>,open,duration	04.03.12.01
Toggle a relay.	---	df_toggle_relay=<RELAIS_ID>,duration	04.03.12.01
Configure communication retries	DFCGetDevicePollRetry	Will not be implemented.	---
Query the device handle	DFCGetComPort	Will not be implemented.	---
Set the device handle.	DFCSetComPort	Will not be implemented.	---
Raw write to a channel	DFCWrite	Will not be implemented.	---
Raw read from a channel	DFCRead	Will not be implemented.	---
Send firmware file to a device.	DFCUpload	Transfer file of type 0xDF01 (Firmware) df_load_file=<PATH ON SRV> Please see Appendix E.	04.03.20.01
Read the version of the firmware running on a device..	DFCGetVersionFirmware	df_kvp=firmwareversion	04.03.12.01
Read the version of the firmware file	DFCGetVersionFirmwareFromFile	Impossible using http.	---
Query information about a module.	DFCGetInfo	Will not be implemented.	---
Activate transparent mode.	DFCOpenComServerMode	Impossible using http.	---
Deactivate transparent mode.	DFCCloseComServerMode	Impossible using http.	---

Check if a channel is currently opened.	DFCIsChannelOpen	Impossible using http.	---
Update of a module.	DFCUploadModule	Transfer file of types 0xDF011 - 0xDF16 (RS9100, Biokey 3000/4000/4020, Saturn 01, U&Z Radio Base) df_load_file=<PATH ON SRV>	04.03.12.03
Read firmware options.	DFCGetOptionFirmware	Not scheduled yet.	---
Write firmware options	DFCSetOptionFirmware	Not scheduled yet.	---
Reset a device.	DFCReset	May be triggered by setting the (temporary) system variable through df_var=system.reset,1.	04.03.20.01
Set font for SendMessage and SendInfotext.	DFCSetFontType	Implemented through df_msg (2.2.3.2.6) and df_info_msg (2.2.3.2.9)	04.03.10.01
Set access password for device.	DFCSetPassword	Not planned, security is part of https.	04.03.11.00
Query failover key.	DFCGetPasswordKey	Will not be implemented.	---
Start a chain on the device (F1 - F15)	DFCPressVirtualKey	df_ek (see 2.2.3.2.5)	04.03.10.00
Retrieve current flash state	DFCGetFlashStatus	df_kvp=flashstate Note: This will only send information on the flash’s partitioning, not the wear level details.	04.03.12.01
Set the channel’s communication key	DFCSetCommunicationPassword	A communication password is not usable in http context. Security is achieved by using https.	---
Get information on the device’s hardware	DFCReadHardwareInfo	Transfer file of type 0xDF10 df_send_file=<PATH ON SRV>,hip	04.03.xx.xx
Send file’s content from the PC to the device.	DFCFileUpload	Please use a specialized function (df_setup_list or df_ac_list) it applicable. If no such function is available and no special function is linked to the file, you may use df_fs_load=root/user:<FILENAME ON DEVICE>,<PATH ON SRV>	04.03.12.01
Send file content from the device to the PC.	DFCFileDownload	df_send_file=<PATH ON SRV>,flash,root/user:<File1>…	04.03.12.01
Return most recently encountered error number.	DFCGetLastErrorNumber	Not possible using http	---
Send setup file to the device.	DFCSetupLaden	Transfer file of type 0xDF02 (Setup file) df_load_file=<PATH ON SRV> Note: The device creates a copy when being sent a setup.	04.03.12.01

Read setup file from the device.	DFCDownload	df_send_file=<PATH ON SRV>,setup Note: Sends a copy of the setup. A setup sent through the communication library may not be read back using this function.	04.03.12.01
Modify content of the setup.	DFCModifyStudioFile	Not planned.	---
Open a list on the device.	DFCOpenTable (obsolete, use DFCTableOpen)	Will not be implemented.	---
Close a list on the device.	DFCCloseTable (obsolete, use DFCTableClose)	Will not be implemented.	---
Set the list filter.	DFCSetFilter (obsolete, use DFCTableSetFilter)	Will not be implemented.	---
Retrieve the current list filter.	DFCGetFilter (obsolete, use DFCTableGetFilter)	Will not be implemented.	---
Remove the list filter.	DFCClearFilter (obsolete, use DFCTableRemoveFilter)	Will not be implemented.	---
Move the windows pointer on the list.	DFCSkip (obsolete, use DFCTableSetCurrentRow)	Will not be implemented.	---
Modify the value of a field.	DFCSetField (obsolete, use DFCTableSetCurrentColumnData)	Will not be implemented.	---
Read the value of a file.	DFCGetField (obsolete, use DFCTableGetCurrentColumnData)	Will not be implemented.	---
Open a list on the device.	DFCTableOpen	Will not be implemented.	---
Close a list	DFCTableClose	Will not be implemented.	---
Set the filter for a column.	DFCTableSetFilter	Will not be implemented explicitly, is implicit part of df_table_select, df_table_update and df_table_delete.	---
Read the current filter.	DFCTableGetFilter	Will not be implemented, see DFCTableSetFilter	---
Remove the current filter	DFCTableRemoveFilter	Will not be implemented, see DFCTableSetFilter	---
Get the number of rows of the list.	DFCTableGetRowCount	df_table_count	04.03.15.01
Get the current line number of the window pointer.	DFCTableGetCurrentRow	Will not be implemented – please use filter criterion and df_table_select.	---
Move the current window pointer for the list	DFCTableSetCurrentRow	Will not be implemented	---
Overwrite current row in the list.	DFCTableSetCurrentRowData	Will not be implemented – please use filter criterion and df_table_update.	---
Overwrite a column’s value in the current row of the list.	DFCTableSetCurrentColumnData	Will not be implemented – please use filter criterion and df_table_update.	---
Set a column’s value for all rows matching the current filter.	DFCTableSetAllRowsToColumnData	df_table_update	04.03.15.01

Read the current row.	DFCTableGetCurrentRowData	Will not be implemented – please use filter criterion and df_table_select.	---
Read a column from the current row.	DFCTableGetCurrentColumnData	Will not be implemented – please use filter criterion and df_table_select.	---
Append a row to the list.	DFCTableAppendRowData	df_table_append	04.03.15.01
Remove the current row from the list..	DFCTableDeleteCurrentRow	Will not be implemented – please use filter criterion and df_table_delete.	---
Remove all rows from the list matchin the current filter.	DFCTableDeleteAvailableRows	df_table_delete	04.03.15.01
Erase all datasets from the device.	DFCComClearData	Not scheduled yet.	---
Start collection of device datasets	DFCComCollectData (obsolete, use DFCReadRecord, DFCQuitRecord)	The device transmits data actively.	04.03.12.01
Retrieve a dataset from the device.	DFCComGetDatensatz (obsolete, use DFCReadRecord, DFCQuitRecord)	The device transmits data actively.	04.03.12.01
Read dataset description from the device.	DFCLoadDatensatzbeschreibung	df_send_file=<PATH ON SRV>,structure,datasets	04.03.12.06
Compute the number of dataset descriptions.	DFCDatBCnt	df_send_file=<PATH ON SRV>,structure,datasets	04.03.12.06
Retrieve base information on dataset.	DFCDatBDatensatz	df_send_file=<PATH ON SRV>,structure,datasets	04.03.12.06
Retrieve base information on a dataset's field.	DFCDatBFeld	df_send_file=<PATH ON SRV>,structure,datasets	04.03.12.06
Read next dataset stored on the device.	DFCReadRecord	Not required – the device sends the dataset automatically.	04.03.10.00
Acknowledge most recently read dataset (it will be removed from the device then)	DFCQuitRecord	df_api=1 in the server's response.	04.03.10.00
Restore datasets on the device.	DFCRestoreRecords	Not scheduled yet.	---
Import raw list data into the DLL.	DFCMakeListe	Not necessary.	---
Upload imported list data to the device.	DFCLoadListen	df_setup_list oder df_ac2_list (2.2.3.2.21)	04.03.10.01
Clear list data from imported lists.	DFCClrListenBuffer	Not necessary.	---
Retrieve list data definitions from the device.	DFCLoadListenbeschreibung	df_send_file=<PATH ON SRV>,structure,lists	04.03.12.06
Compute the number of list definitions.	DFCListBCnt	df_send_file=<PATH ON SRV>,structure,lists	04.03.12.06
Retrieve base information on a list	DFCListBDatensatz	df_send_file=<PATH ON SRV>,structure,lists	04.03.12.06
Retrieve base information on a list's field.	DFCListBFeld	df_send_file=<PATH ON SRV>,structure,lists	04.03.12.06
Import raw list data into the DLL.	DFCMakeEntranceList	Not necessary.	---

Upload imported list data to the device.	DFCLoadEntranceList	df_setup_list or df_ac2_list (2.2.3.2.21)	04.03.10.01
Clear list data from imported lists.	DFCClearEntranceListBuffer	Not necessary.	---
Import raw list data into the DLL.	DFCMakeEntrance2List	Not necessary.	---
Upload imported list data to the device.	DFCLoadEntrance2List	df_setup_list or df_ac2_list (2.2.3.2.21)	04.03.10.01
Clear list data from imported lists.	DFCClearEntrance2ListBuffer	Not necessary.	---
Trigger access control virtually	DFCEntrance2Identification (obsolete, use DFCAccessControlIdentification)	Will not be implemented, see df_trigger_ac2 (2.2.3.2.13)	---
Call an access control module.	DFCEntrance2OnlineAction (obsolete, use DFCAccessControlOnlineAction)	Will not be implemented.	---
Trigger access control virtually	DFCAccessControlIdentification	df_trigger_ac2=<Reader-ID>,<Transponder-ID>	04.03.12.06
Call an access control module.	DFCAccessControlOnlineAction	df_ac2 (see 2.2.3.2.10)	04.03.10.01
Append a fingerprint template to the device's sensor	DFCFingerprintAppendRecord / DFCFingerprintRestore	Transfer file of type 0xDF18 df_load_file=<PATH ON SRV> Note: The fingerprint data from the file is integrated into the fingerprint modules content. If the module contains a finger with the same PID and FID as contained in the file, the fingertemplate in the module is replaced by that from the file.	04.03.12.05 (optical sensor) 04.03.15.08 (line sensor)

Retrieve a fingerprint template from the device's sensor	DFCFingerprintGetRecord / DFCFingerprintBackup	df_send_file=<PATH ON SRV>,finger,<PID>,<FID> or
		df_send_file=<PATH ON SRV>,finger,<PID>,all or
		df_send_file=<PATH ON SRV>,finger,all
		Creates a transfer file of type 0xDF18
			04.03.12.05 (optical sensor)
			04.03.15.08 (line sensor

Clear the fingerprint templates from the device's sensor.	DFCFingerprintDeleteRecord	df_remove_finger=<PID>,<FID> or
		df_remove_finger=<PID>,all or
		df_remove_finger=all
			04.03.12.05 (optical sensor)
			04.03.15.08 (line sensor

Create a list of PID,FID pairs contained in the device's sensor	DFCFingerprintList	Not necessary.	---
Import raw data for a timeboy's list.	DFCMakeTimeboyList	Not scheduled yet.	04.03.xx.xx
Upload imported list data to a timeboy.	DFCLoadTimeboyList	Not scheduled yet.	04.03.xx.xx

Clear the timeboy list data.	DFCClearTimeboyListBuffer	Not scheduled yet.	04.03.xx.xx
Start active mode server.	DFCStartActiveConnection	Realized by http server.	---
Shut the active mode server down.	DFCStopActiveConnection	Realized by http server.	---
Compute the ID of the first device connected to the active mode server.	DFCGetFirstActiveChannelID	Not necessary – the device establishes the connection.	---
Compute the ID of the next device connected to the active mode server.	DFCGetNextActiveChannelID	Not necessary – the device establishes the connection.	---
Collect information associated to an active mode channel.	DFCGetInfoActiveChannel	Not necessary – the device establishes the connection.	---
Activate or deactivate the information on „available datasets"	DFCSetRecordAvailable	Not necessary – the device establishes the connection.	---
Access the queue for „available datasets".	DFCRecordAvailable	Not necessary – the device establishes the connection.	---
Associate a device to a channel number.	DFCBindDeviceToChannel	Nicht erforderlich – das Gerät baut die Verbindung auf.	---
Configure block types where the transfer may be interrupted.	DFCBlockTransferSetDuration	Not reasonable for http data transfer.	---
Resume the block transfer.	DFCBlockTransferResume	Not reasonable for http data transfer.	---
Compute the current type and state of block transfer.	DFCBlockTransferGetState	Not reasonable for http data transfer.	---
Discard the currently pending block transfer.	DFCBlockTransferDiscard	Not reasonable for http data transfer.	---

Information on success or failure of an http action is supplied through system messages.

A.2: Comparison of Datafox Studio and http Level 1

Function in the Datafox Studio	**Direction**	**Function in http Level 1**	**State / Plan**
Upload of device firmware	-> Device	- Web-Server sends request for the firmware update to the device - The device sends its firmware version to the web server. - Webserver checks, which firmware file to provide and sends the download URL to the device - The device downloads a firmware transfer file type 0xDF01	04.03.20.01
Upload of device setup	-> Device	Device loads transfer file type 0xDF02	04.03.12.01
Download of device setup	-> Studio	df_send_file=<PATH ON SRV>,setup, transfer file type 0xDF0C (2.2.3.2.18)	04.03.12.01

Upload of setup list data	-> Device	df_setup_list (2.2.3.2.21) [0xDF03]	04.03.10.01
Upload of access control lists data	-> Device	df_ac2_list (2.2.3.2.21) [0xDF04]	04.03.10.01
Upload of Timeboy list data	-> Device	Transfer file type 0xDF05	04.03.xx.xx
Retrieve and delete datasets	-> Studio	Datasets are actively sent by the device.	04.03.10.00
Read camera pictures	-> Studio	Device sends new images using a standard web upload form using transfer file type 0xDF06. The upload path is specified by the system variable COM.HTTP_MODE[.].SEND_IFF.	04.03.18.04
Read serial number	-> Studio	df_kvp=serialnumber (2.2.3.2.14)	04.03.12.01
Set device clock	-> Device	df_time (2.2.3.2)	04.03.10.00
Send a message	-> Device	df_msg (2.2.3.2.6)	04.03.10.01
Read a global variable	-> Studio	df_kvp=var,<VAR_NAME> (2.2.3.2.14)	04.03.12.01
Upload a language file to the device	-> Device	Transfer file type 0xDF07	04.03.18.04
Upload a color map to the TimeboyIV	-> Device	Transfer file type 0xDF08	04.03.xx.xx
Upload a LAN/WLAN configuration to the device	-> Device	Transfer file type 0xDF09	04.03.xx.xx
Upload a touch configuration (EVO 4.3) to the device	-> Device	Transfer file type 0xDF0F	04.03.xx.xx
Upload a display design to the device	-> Device	Transfer file with images (BMP) and mainmenu.bin, each represented by a 0xDF00 chunk. A file may be exported by the Display Designer.	04.03.21.01
Upload an U&Z configuration to the device	-> Device	Transfer file type 0xDF0B	04.03.xx.xx
Read system variables	-> Studio	df_kvp=var,<VAR_NAME> (same as global variables) (2.2.3.2.14)	04.03.12.01
Write system variables	-> Device	df_var (2.2.3.2.4)	04.03.10.01
Retrieve the device's system log	-> Studio	df_send_file=<PATH ON SRV>,syslog, Transfer file type 0xDF0C (2.2.3.2.18)	04.03.12.01
Update for the BioKey Modul	-> Device	Transfer file types 0xDF12 to 0xDF15	04.03.xx.xx
Retrieve fingerprint templates from the device	-> Studio	df_send_file=<PATH ON SRV>,finger,..., Transfer file type 0xDF18 (2.2.3.2.18)	04.03.12.05 (optical sensor) 04.03.15.08 (line sensor
Delete fingerprint templates from the device	-> Device	df_remove_finger=<PID>,<FID> (2.2.3.2.20)	04.03.12.05 (optical sensor) 04.03.15.08 (line sensor

Append / Write fingerprint templates to the device.	-> Device	Transfer file type 0xDF18	04.03.12.05 (optical sensor) 04.03.15.08 (line sensor

A.3: Structure of a transfer file

The transfer file is structured according to the IFF file format. This file format allows representing different data types by “chunks”. Chunks the can be interpreted according to their type and relative location inside the structure, unknown chunks may be skipped while evaluating the IFF file.

Setup list data and access control list data may be represented within a single IFF container file or distributed individually using `df_setup_list` and `df_ac2_list` among the devices.

The general structure of an IFF file is shown in the following table:

	Offset	Length	Name	Description	Example
DFIF-Header	0	4	Header	The chunk is of type „FORM“	„FORM“
	4	4	File length	Length of the form minus 8 (this is the amount of data left to read!)	16542
	8	4	Form Type Information	Id of the form	„DFIF“
Version data	12	4	Header	The chunk is of type „DFFV“	„DFFV“
	16	4	Length of data	Length of the form minus 8	28
	20	12	FW version	Firmware version string	„04.03.11.00\0“
	32	16	FW suffix	Identification for Beta- or RC versions	„https.1\0“
File 1	48	4	Header	The chunk is of type „FORM“	„FORM“
	52	4	Length of data	Length of the form (-8)	16366
	56	4	Form Type Information	Id of the form	„DFF0“
File type	60	4	Chunk-Header	Chunk Header “Filetype”	„FTYP“
	64	4	Length of data	Length of the form (-8)	4
	68	2	File type	Id of the file type as detailed below	0xDF02 (Setup)
	70	2	CRC (0xA001) of DATA segment content	CRC checksum computed for the following data chunk’s content (without its chunk header)	0xF00D
Filename	72	4	Chunk-Header	Chunk Header “Filename”	„FNAM“
	76	4	Length of data	Length des Chunk (-8)	9
	80	10	File name	Name of the file	„setup.aes“
	89	1	Padding-Byte	Padding-Byte, since the length of the filename is odd.	0

Data	90	4	Chunk-Header	Chunk Header “Data”	„DATA“
	94	4	Length of data	Length of Chunk (-8)	sz
	98	16315	Daten	The file’s content	
	16413	1	Padding-Byte	Padding-Byte, since sz (16315) ís odd.	0
File 2	16414	4	Form-Id	Second form with another file Additional forms and chunks of file 2	„FORM“
	16418	4	Length of data		127
	16422	4	Form Type Information		„DFF0“
	16426	123			
	16549	1	Padding-Byte	Padding-Byte, since sz (16315) ís odd.	0
Signatur	16550	4	Chunk-Header	Chunk-Header Signature – this chunk contains the signature of the following form	„SIGN“
	16554	4	Length	Length of Chunk (-8)	18
	16558	2	Algorithm	Signature algorithm 0=MD5+AES, 1=SHA1+AES, 27=SHA256+PubKey	0x0001
	16560	16	Signature Daten	Signature of the content of the following DFF0 form (without optional padding byte on DFF0-Level)	
File 3	16578	4	Form-Id	Third, signed form with another file	„FORM“
	16582	4	Length	Length of Form (-8)	
	16586	4	Form Type		„DFF0“
	16590	773	Data		
	17367	1	Padding-Byte	Padding byte, since 773 is odd (this last padding byte is not part of the signature!)	

A file may contain an arbitrary number of chunks. The information on the file being from the Datafox context is derived from the top level FORM type „DFIF“.

☞ **Note:**
The IFF format was specified by Electronic Arts in 1985.
At this point in time, big-endian encoding of data was dominant. This, the **chunk length** is encoded in **big-endian** despite little endian being the predominant encoding today.

Example: A chunk has a size of 123456 Bytes = 0x1E240 Bytes. In Big-Endian this is encoded by the byte sequence 0x00 0x01 0xE2 0x40.

See https://en.wikipedia.org/wiki/Interchange_File_Format or https://en.wikipedia.org/wiki/Endianness.
Standard: http://wiki.amigaos.net/wiki/EA_IFF_85_Standard_for_Interchange_Format_Files

!

Attention:
When embedding data into chunks that have odd (un-even) length (e.g. file content or file names), it is required that the chunk is padded using one additional Zero-Byte. This padding byte is not included into the chunk's length and is mandatory.

Concrete:

- A chunk always occupies and even number of bytes – even if its content has an odd length.
- The padding byte is **not** included into the CRC computation

A.3.1: Forms and Chunks contained in the transfer file

The transfer file's topmost node is a FORM. The form's type is "DFIF" as shown in the previous chapter. The form is followed by chunks containing version information ("DFFV"), an optional description form "DESC" explaining the file's content in a human readable way and a list of files – each encoded into a "DFF0" form.

A.3.1.1: Version information [Chunk „DFFV"]

Version information chunk are associated to the IFF file's creator:

	Offset	#	Name	Description	Example
Version	+0	4	Header	Header of the version chunk	„DFFV"
	+4	4	Length	Length of the Chunk (-8 bytes header)	28
	+8	12	FW version	Version of the firmware executed on the device.	„04.03.11.00\0"
	+20	16	FW suffix	Beta- or RC-Tag, if the device firmware is such.	„https.1\0"

A.3.1.2: Description of the file's content [FORM „DESC"]

Currently, IFF files are typically sent to the device using Datafox Studio. To present information on the file's content and intention, a DESC form can be added.

The description form contains information on the hierarchy ("HIER") where to place the information as well as descriptive text ("HTML").

	Offset	#	Name	Description	Example
D	+0	4	Header	Form-ID	„FORM"

	+4	4	Length	Length of the form (-8)	4
	+8	4	Form Type	Description	„DESC“

A.3.1.2.1: Hierarchy tag for the description [Chunk „HIER“]

With one or more hierarchy tags, the content of the file may be presented with the right context. By using two hierarchy tags, you may put an update file for the WIFI module into the category “module updates” / “WIFI”.

	Offset	#	Name	Description	Example
Hierarchy	+0	4	Header	Header of the hierarchy chunk	„HIER“
	+4	4	Length	Length of the chunk (-8)	Sz + 6
	+8	4	Language	Language that applied to the hierarchy tag (if there is more than one)	„DE\0“
	+12	2	Level	Level inside the hierarchy, starting at 0	0
	+14	sz	Text	Description of the hierarchy level	„Module“ or „WLAN“
	+14 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

A.3.1.2.1: Description text [Chunk „HTML“]

An html chunk contains a description of the file’s content in human readable language. This description is associated to a language as the hierarchy tags are. The text is encoded as HTML – focus should be content not stylesheet.

	Offset	#	Name	Description	Example
Beschreibungstext	+0	4	Header	Header of the description chunk	„HTML“
	+4	4	Length	Length of the chunk (-8)	Sz + 4
	+8	4	Language	Language of the description	„DE\0“
	+12	sz	Text	Description of the transfer file’s content	
	+12 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

A.3.1.3: File content [FORM „DFF0“]

Header form used to transfer a file’s content. Inside of this form the following chunks are typically used:

- File name,
- Encoding of the file data [Optional, Default: binary content]
- Auxiliary information [depending on the file type]
- Informationen on (HW-) Compatibility
- And the content of the file

	Offset	#	Name	Description	Example
File Header	+0	4	Header		„FORM“
	+4	4	Length	Total length of form and contained chunks (-8)	4 + Sz
	+8	4	Form Type	Datafox File Version 0	„DFF0“

Sz is the amount of data contained inside the FROM.

A.3.1.3.1: Type of data [CHUNK „FTYP“]

The „FTYP“ chunk inside of the transfer file form (“DFF0”) specifies how the data may be verified as well as what kind of data is contained inside the file.

	Offset	#	Name	Description	Example
FTYP Chunk	+0	4	Chunk-Header	Header of the file type chunk	„FTYP“
	+4	4	Length	Length of the chunk (-8)	4
	+8	2	Type of data	Content type according to specification	0xDF01
	+10	2	CRC (0xA001) of data Initial: 0xffff XorOut: 0	Standard CRC checksum computed across the content of DATA or DATE chunk contained in this DFF0 form. If we have encrypted data, the checksum is computed for the encrypted representation of the data. We do not give away a hint on the encryption in this way and allow integrity verification even without the correct key.	0xF00D

We use the following C-Code to compute the CRC Checksum on a data block (`data` and `size` define the raw block of data, `crc` is set to `0xffff` when calling the function):

```
// CRC 16bit berechnen
unsigned short GetCRC16(const unsigned char *data, unsigned int size, unsigned short crc)
{
    unsigned int i;
    unsigned char j;

    for ( i = 0; i < size; i++ )
    {
        crc = crc ^ data[i];
        for ( j = 0; j < 8; j++ )
        {
            if ( crc & 0x1 )
            {
                crc >>= 1;
                crc = crc ^ 0xA001;
            }
            else
            {
                crc >>= 1;
            }
        }
    }
    return crc;
}
```

A.3.1.3.2: Auxiliary parameters [CHUNK „FAUX“]

This chunk contains additional data which – depending in the file type – are necessary to evaluate the content. Please refer to the appendix defining the file types and their need for auxiliary data. If a file type requires some additional data and it is omitted, the file’s content most likely will be rejected by the device.

	Offset	#	Name	Description	Example
Aux	+0	4	Header	Header of the aux data chunk	„FAUX“
	+4	4	Length	Chunk length (-8)	sz
	+8	2	Version	Version number for this chunk’s data – if versioning is required in the future	1
	+10	sz	Auxdata	Additional data	e.g. PID and FID of a finger template
	+10 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

A.3.1.3.3: File name [CHUNK „FNAM“]

This chunk contains the name of the file being transferred.

	Offset	#	Name	Description	Example
Filename	+0	4	Header	Header of the file name chunk	„FNAM“
	+4	4	Length	Chunk length (-8)	sz
	+8	sz	File name	Name of the file	„my.cert“
	+8 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

Please note: The version numbering of the FAUX chunk starts at 1. Please consult Appendix A 3.2 on content of the FAUX chunk.

A.3.1.3.4: Encoding-Informationen des Datenblocks [CHUNK „ENC “]

Chunk with content encoding information:

	Offset	#	Name	Description	Example
Zeichen-Encoding	+0	4	Header	Header of the encoding chunk	„ENC “
	+4	4	Length	Chunk length (-8)	sz
	+8	sz	Encoding	Mime type associated to the data encoded in this file form. If this chunk is omitted, the data is treated as binary data.	ISO-8859-1
	+8 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

A.3.1.3.5: Compatibility information [CHUNK „COMP“]

This chunk applies to update file chunks, as used e.g. for module updates. The chunk contains information on the hardware being compatible with the update. The internal structure is not disclosed here.

	Offset	#	Name	Description	Example
Compatibility	+0	4	Header		„COMP“
	+4	4	Length	Chunk length (-8)	2 + sz
	+8	2	Version	Version of this chunk	1
	+10	sz	Data	Data on compatibility	
	+10 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

A.3.1.3.6: Datei-Inhalt [CHUNK „DATA“]

Note: Alternatively to the DATA chunk there is chunk containing internally encrypted data. The encrypted version is used e.g. for firmware updates or module updates and not meant to be created outside Datafox (see DATE chunk for details)

Chunk containing the raw file content:

	Offset	#	Name	Description	Example
File data	+0	4	Header		„DATA“
	+4	4	Length	Chunk length (-8)	sz

	+8	sz	Data	Content of the file	
	+8 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

A.3.1.3.7: Encrypted data chunk (replaces DATA chunk, if used) [CHUNK „DATE“]

The DATE chunk offers additional information on the contents intended placement. Additionally the content is encrypted, so that the device can detect if it was manipulated. Chunks of this type are not intended to be created by anyone outside Datafox.

	Offset	#	Name	Description	Example
Encrypted data segment	+0	4	Header	Chunk header for encrypted data	„DATE“
	+4	4	Length	Chunk length (-8)	20 + sz
	+8	2	Version	Internal chunk revision information	0
	+10	2	Internal Type	Internal type (Flash, Font, Text, Icon, …)	
	+12	4	Start offset	Start offset e.g. of a firmware update segment	0x00208000
	+16	4	End offset	End offset e.g. of a firmware update segment	0x003fffff
	+20	4	CRC	CRC computed for the unencrypted data	
	+24	4	Seed	Seed value to be for decryption	0x12345678
	+28	sz	Data	Encrypted content of the file	

A.3.1.3.8: Signature-Chunk [CHUNK „SIGN“]

The signature chunk stores the signature of the next chunk within the IFF file.

	Offset	#	Name	Description	Example
Signa-	+0	4	Header	Chunk containing a signature	„SIGN“
	+4	4	Length	Chunk length (-8)	2 + sz
	+8	2	Method	0 – MD5	0
	+10	sz	Signature data		

A.3.1.3.9: Signed data chunk [CHUNK „DATS“]

Within a firmware update file (0xDF01) one or more data chunks as described within this chapter are contained. They may contain text, program or other data. These chunks are meant to be created by Datafox and will not be processed if the signature chunk SIGN does not contain the correct signature data.

	Offset	#	Name	Description	Example
Data segment	+0	4	Header	Chunk containing signed data	„DATS“
	+4	4	Length	Chunk length (-8)	12 + sz
	+8	2	Version	Version assigned to this chunk	
	+10	2	Internal Type	Internal data type (Flash, Fonts, Text, Icons, …)	
	+12	4	Start offset	Start-Offset	`0x00208000`
	+16	4	End offset	End-Offset	`0x003fffff`
	+20	sz	Binary data	Data	

A.3.1.4: Record / List data description [FORM „DFDS“]

This form contains header data for record or list data descriptions. With the form the following chunks may be contained:

- Name of the dataset definition (as defined by the setup)
- Index of the dataset definition (as defined by the setup)
- Index of the priority field (optional)
- Index of the selection key field (optional)

Additionally the form may contains DCOL forms that will detail the individual fields of record of list data field (see A.3.1.5)

	Offset	#	Name	Description	Example
Dataset	+0	4	Header		„FORM“
	+4	4	Length	Total length of form and contained chunks (-8)	4 + Sz
	+8	4	Form Type ID	Datafox Data Structure	„DFDS“

A.3.1.4.1: Data record name [„DNAM“]

This chunk contains the name of the data record / list. The name is composed out of the type and the name as defined by the setup – separated by a single dot. The setup list “personal” is thus represented by “list.personal”.

The following prefixes are being used:

- “list” is used for setup defined list data
- “access” is used for access control lists data
- “record” is used for dataset record structure

	Offset	#	Name	Description	Example

Dataset-	+0	4	Header		„DNAM“
	+4	4	Length	Total length of chunk (-8)	sz
	+8	sz	Filename	Name of the record	„record.booking“
	+8 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

A.3.1.4.2: Index of the data record with the setup [„DIDX“]

This chunk contains a data record’s index within the setup structures. The index starts at 0.

	Offset	#	Name	Description	Example
Data record	+0	4	Header		„DIDX“
	+4	4	Length	Total length of chunk (-8)	2
	+8	2	Index	Index of the data record description within the setup.	1

A.3.1.4.3: index of the priority field [„DPRI“]

This chunk contains the index of the field that contains the priority information within a data record. If there is no priority defined for a data record, this chunk is omitted.

	Offset	#	Name	Description	Example
Prio index	+0	4	Header		„DPRI“
	+4	4	Length	Total length of chunk (-8)	2
	+8	2	Index	Index of the record’s field containing the priority data.	1

A.3.1.4.4: Index of the key field [„DKEY“]

This chunk contains the index of the column according to that the (list) data is sorted.

	Offset	#	Name	Description	Example
Key in-	+0	4	Header		„DKEY“
	+4	4	Length	Total length of chunk (-8)	2
	+8	2	Index	Index of the key column	1

A.3.1.5: Information on columns of lists or data records [FORM „DCOL“]

This form is used as a header containing information on a single data record or list column. The form contains the following chunks:

- Field/column name chunk

- Field/column content information chunk

	Offset	#	Name	Description	Example
Column	+0	4	Header		„FORM“
	+4	4	Length	Total length of form and contained chunks (-8)	4 + Sz
	+8	4	Form Type Id	Datafox Column Data	„DCOL“

A.3.1.5.1: Column content information chunk [„CINF“]

This chunk contains the data type definition for a field/column.

	Offset	#	Name	Description	Example
Field/Column type	+0	4	Header		„CINF“
	+4	4	Length	Total length of chunk (-8)	6
	+8	2	Type	Type of the field 2 – Date and Time 3 – Numerical value 4 – Alphanumerical value 7 – Fingerprint Template DIN V44600 (161 Byte) 8 – Fingerprint Template Idencom Compact (216 Byte) 9 – Binary data (max. 220 Byte)	4
	+10	2	Size	Size of the field/column in bytes	35
	+12	2	Index	Index of the field/column	5

A.3.1.5.2: Column name chunk [„CNAM“]

This chunk contains the name of a field/column.

	Offset	#	Name	Description	Example
Field/Column	+0	4	Header		„CNAM“
	+4	4	Length	Total length of chunk (-8)	sz
	+8	sz	Name	Name of the field/column	„Personal Id“
	+8 + sz	0/1	Opt. Padding-Byte	Optional padding byte if sz is odd.	0

A.3.2: File types

Each transfer file exchanged between device and server contains a type identifying the file's content – the first two bytes denoted as "Version" represent the version from section A.3.1.3.2: Auxiliary parameters [CHUNK „FAUX"] – thus this parameter is shown in italics font int the following table.

File type	Content of the file	Device -> Server	Server -> Device
0xDF00	File from or for the device's file system. The FNAM chunk is meant to contain the entire file path, if the file is not to be located in the device's top-most directory. (please check 2.2.3.2.18)		
0xDF01	Firmware file *	No	
0xDF02	Setup file		
0xDF03	List data (setup) Auxiliary data: [*Version* >> *8, Version*, List name]		
0xDF04	List data (access control 2) Auxiliary data: [*Version* >> *8, Version*, List name]		
0xDF05	List data (for Timeboy, will be sent to a MasterIV device and then transferred to the connected Timeboy(s)). Auxiliary data: [*Version* >> *8, Version*, GroupID >> 8, GroupID, List name]	No	
0xDF06	Image (Camera or Signature) or large Barcode data (general: File data created by the device)		No
0xDF07	Language file		
0xDF08	Color definition file for the Timeboy		
0xDF09	LAN / WIFI configuration file		
~~0xDF0A~~	~~Display design file~~ The display design consists out of a set of files, which are transferred as 0xDF00 files. Please use the Display Designer from the Datafox Studio to export your Design in an appropriate format.		
0xDF0B	U&Z configuration file		
0xDF0C	Systemlog		No
0xDF0D	Bootloader *	No	
0xDF0E	Fingerprint template data (Saturn 01) Auxiliary data: [*Version* >> *8, Version*, PID >> 24, PID >> 16, PID >> 8, PID, GID >> 8, GID, FID] Please use 1 for GID parameter (Group ID). GID 0 is invalid, GIDs > 1 are used for verification with transponders.	No -> 0xDF18	
0xDF0F	Touch configuration file		

0xDF10	Hardware info file (HIP) *		
0xDF11	Update for WIFI module RS9110 *	No	
0xDF12	Update for fingerprint sensor Biokey 3000 modul *	No	
0xDF13	Update for fingerprint sensor Biokey 4000 modul *	No	
0xDF14	Update for fingerprint sensor Biokey 4020 modul *	No	
0xDF15	Update for fingerprint sensor Saturn 01 *	No	
0xDF16	Update for the U&Z radio base station (FSM) *	No	
0xDF17	Update for the proximity and ambient light sensor	No	
0xDF18	Fingerprint data (Saturn 01 or Idencom) Aux-Parameter: [*Version* >> *8, Version*, Template-Type, PID >> 24, PID >> 16, PID >> 8, PID, GID >> 8, GID, FID] Template-Type: 0 – DIN V66400 format (161 Byte) 1 – Idencom-Compact format (216 Byte) 2 – Idencom-Standard format (561 Byte) 3 – Saturn 01 binary template format Please use 1 for GID parameter (Group ID). GID 0 is invalid, GIDs > 1 are used for verification with transponders		
0xDF19	Directory listing, see df_send_file (2.2.3.2.18) with parameter „dir“.		No
0xDF19	Configuration package for an TWN4 reader, that allows rea-ding different RFID transponders (BIX / AppBlaster)	No	
TBA	Update to be sent across the AC2 bus * Auxiliary data: [*Version* >> *8, Version*, Modul-ID >> 8, Modul-ID]	No	

* **Please Note:** File of this type will be created by Datafox only. The DATE chunk is encrypted using a private Datafox key which allows the device receiving it to verify the authenticity and integrity of the file.

Appendix B: HTTPS Communication

The communication described in this manual can use the encrypted https protocol from firmware version 04.03.11.01 on.

B.1: Elements of the https infrastructure

HTTPS is – as is HTTP – a client/server protocol. The client initiates a network connection to a target port on the HTTPS server using TCP/IP, the data exchange is using encryption against eavesdropping attacks.

HTTPS uses asymmetric encryption (for negotiating during the HTTPS handshake) in form of a server certificate as well as symmetric encryption for later data exchange.

B.2: Establishing the connection

The HTTPS communication is performed as implemented in current web browsers.

- The connection is initiated by the terminal by connecting to a port on the server
- Handshake between client and server
  - Negotiation of the encryption algorithm
- Exchange of the certificates
  - Validation of the server's certificate. If the certificate is validated successfully, the client has access to the server's public key. Using this public key the client now can securely encode messages that only the server can decrypt with its private key.
  - Optional: If the server requests a client certificate, it will be provided if configured at the client.
- Exchange of keys
  - The key used for symmetric encryption is exchanged between client and server

B.3: Validation of the server certificate

The firmware requires that the certificate provide by the server during the HTTPS handshake is valid. The validity is check classically – the certificate is checked against a list of certificates stored on the device (the so-called CA-bundle that is maintained by every browser or OS nowadays). The CA-bundle contains the certificates that will be accepted by a device.

Use cases:

- Your server ("your-company.de") provides a certificate signed by a Verisign-certificate:

  The server provides the client with its certificate.
  The client determines, that the server certificate "your-company.de" was generate using a Verisign certificate. The client checks its local CA bundle for the Verisign certificate and checks the parentage of the "your-company.de" certificate.

  The following possibilities exist:

  - The client does not have the Verisign certificate in its CA bundle: The certificate "your-company.de" is considered to be **invalid**.
  - The parentage cannot be verified: The certificate "your-company.de" is considered to be **invalid**.

- The provided certificate is derived from the Verisign certificate stored on the device: The certificate “your-company.de” is considered to be **valid**.

- Your server (“your-company.de”) provides a self-signed certificate:

  In this case the validation of the server certificate is only possible if the device contains a copy of the server certificate in its CA bundle.

- Your server (“your-company.de”) provides a certificate signed by Datafox:

  This case is pretty much alike the first case presented in this section. However, Datafox uses a GlobalSign certificate. The certificate issued for „your-company.de“ is derived from the Datafox certificate, which is derived from the GlobalSign certificate.

  When providing its certificate the server “your-company.de” provides the certificate chain as well (excluding the GlobalSign certificate).

  The client checks the server’s certificate chain. If the parentage from the Datafox certificate does not verify, the certificate is considered to be **invalid**.

  If the parentage against the Datafox certificate is valid, the device searches from the GlobalSign certificate in its CA bundle. If the GlobalSign certificate is not stored on the device, the certificate is considered to be **invalid**.

  If the GlobalSign certificate is stored within the CA bundle, the parentage test provides the **validity** of **invalidity** of the “your-company.de” certificate.

B.4: Communication

After successful validation of the server’s certificate, the communication partners negotiate the key to be used for encrypted data exchange (“session key”). This communication uses a symmetrical cipher mechanism. The exchange the session key, the client encrypts the session key using the server’s public key (that was contained inside the server’s certificate). The server then decodes the client’s message using the server’s private key.

The device firmware uses cipher suites TLS 1.1 and newer. The suites TLS 1.0 and even older implementation are considered obsolete for not providing the required security levels any more.

B.5: Using a self-signed (server-) certificate

You may use the OpenSSL Implementation (that typically is installed already on a Linux system and may be installed e.g., using Cygwin on Windows) to create a key pair for HTTPS communication. The private key (my.key) has to be assigned only to the web server, the public key (my.cert) is required by the server as well as by the client.

You may create the new 2048 bit RSA key pair using the following command:

```
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout my.key -out my.cert
```

If you wish to integrate a password into the key file, you may pass it using the additional parameter “-passout file:key.txt” to the call to openssl. The key has to be stored in the local file “key.txt” for this on the machine, running the openssl programm.

You may create different key pairs, e.g. an RSA key pair with 3072 bit key length:

```
openssl req -x509 -nodes -days 365 -newkey rsa:3072 -keyout my.key -out my.cert
```

Or a key pair using elliptic curve cryptography (ECC):

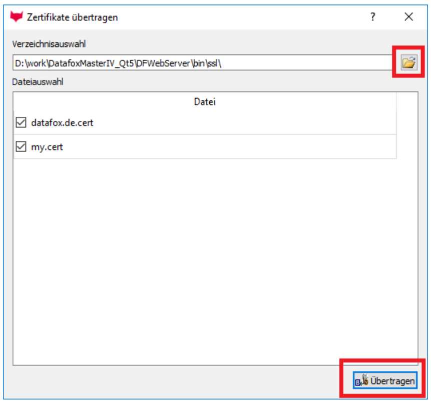
```
openssl ecparam -genkey -name prime256v1 -out key.pem
openssl req -new -sha256 -key key.pem -out csr.csr
openssl req -x509 -sha256 -days 365 -key key.pem -in csr.csr -out certificate.pem
```

The process for creating an ECC key pair uses a CSR (certificate signing request) – which is the normal way when asking a certificate authority to officially sign the certificate – however CSR is the self-signed to create the certificate. To use the created files on a device, you have to rename the files, e.g. key.pem -> ecc.key and certificate.pem -> ecc.cert before uploading them to the device.

B.5.1: Device configuration – Deploying a server certificate

If a device shall talk to an https server, it is mandatory, that the device is capable of verifying the server's certificate. To perform this verification, it is necessary that the device can check the server's chain of certificated from which the server's certificate is derived or knows that the self signed certificated may be trusted.

You may use Datafox Studio from Version 04.03.11.02 to maintain the certificated used by a client. Please use the menu entry "Configuration" -> "Transfer certificates" to upload the desired certificates as a device's trustworthy certificates (CA bundle).



Zertifikate übertragen

Verzeichnisauswahl

D:\work\DatafoxMasterIV_Qt5\DFWebServer\bin\ssl\

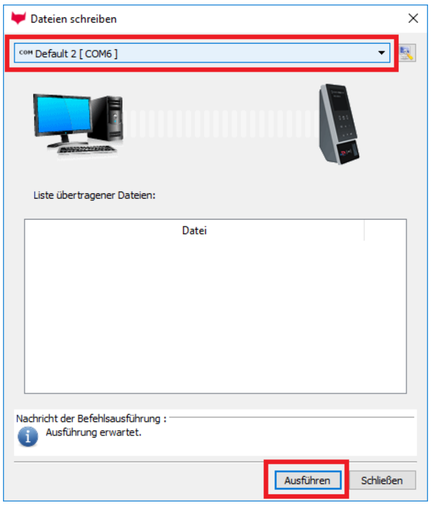
Dateiauswahl

Datei

☑ datafox.de.cert

☑ my.cert

Choose the directory containing the *.cert files to be uploaded to the device. You may check / uncheck the files individually. By selecting "Transfer" the next dialog is displayed.



Dateien schreiben

Default 2 [COM6]

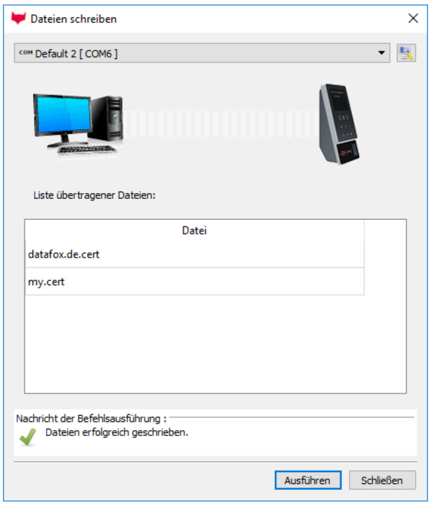
Liste übertragener Dateien:

Datei

Nachricht der Befehlsausführung :

Ausführung erwartet.

In this dialog you my select the device to upload the files to. By selecting „Execute“ the transfer process is started.



Dateien schreiben

Default 2 [COM6]

Liste übertragener Dateien:

Datei

datafox.de.cert

my.cert

Nachricht der Befehlsausführung :

Dateien erfolgreich geschrieben.

After successful operation the transferred certificates are displayed.

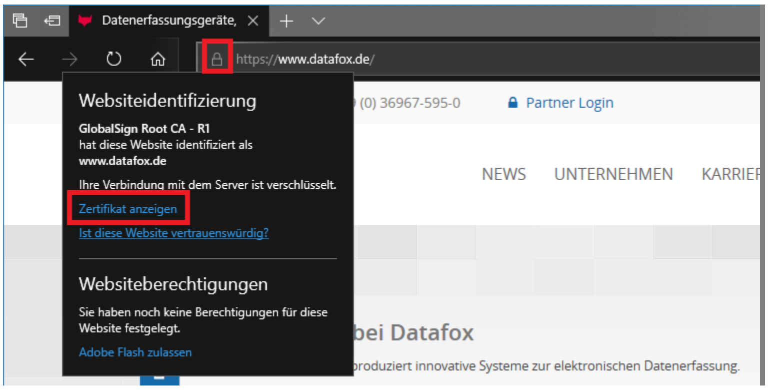
! **Attention:**
Please do **only** transfer the **necessary certificates** to the device when starting your https integration phase (and never transfer a key to the device). The storage area provided by the devices is way smaller than provided by a current web browser of OS.

Attention:
The current firmware implementation reads the certificates only at startup of the device. Currently, you have to **reboot the device** as soon as you **change the certificates** stored on the device.

B.5.2: Which certificate is used by the web server? (“old” Edge Browser)

If unclear, which certificate is used by your web server, you can collect all necessary data from your web browser using the following steps:

Click on the lock symbol of the Microsoft Edge browser. A popup will appear below the lock symbol inside the browser window showing the “Website Identification” data. Activate “Show certificate” to display the certificate chain at the right-hand side of the browser window as shown below:



Datenerfassungsgeräte,

https://www.datafox.de/

Websiteidentifizierung

GlobalSign Root CA - R1
hat diese Website identifiziert als
www.datafox.de

Ihre Verbindung mit dem Server ist verschlüsselt.

Zertifikat anzeigen

Ist diese Website vertrauenswürdig?

Websiteberechtigungen

Sie haben noch keine Berechtigungen für diese Website festgelegt.

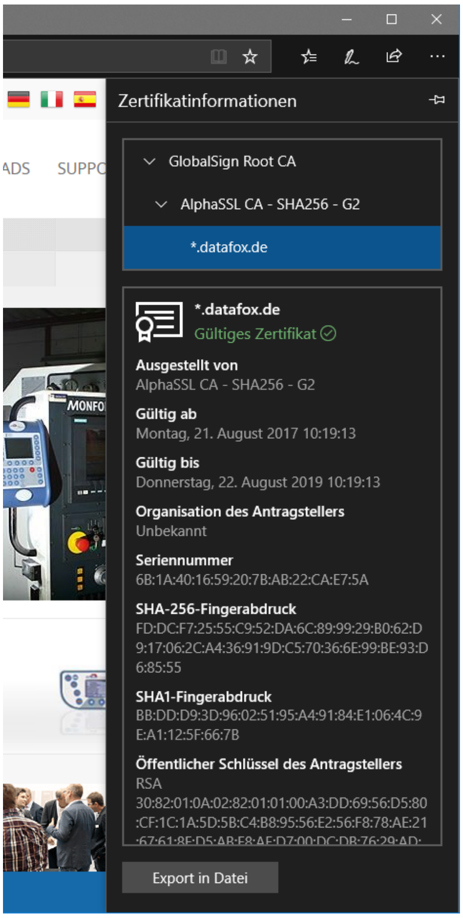
Adobe Flash zulassen

(0) 36967-595-0 Partner Login

NEWS UNTERNEHMEN KARRIER

bei Datafox

produziert innovative Systeme zur elektronischen Datenerfassung.



Zertifikatinformationen

- GlobalSign Root CA
  - AlphaSSL CA - SHA256 - G2
    - *.datafox.de

*.datafox.de
Gültiges Zertifikat ✓

Ausgestellt von
AlphaSSL CA - SHA256 - G2

Gültig ab
Montag, 21. August 2017 10:19:13

Gültig bis
Donnerstag, 22. August 2019 10:19:13

Organisation des Antragstellers
Unbekannt

Seriennummer
6B:1A:40:16:59:20:7B:AB:22:CA:E7:5A

SHA-256-Fingerabdruck
FD:DC:F7:25:55:C9:52:DA:6C:89:99:29:B0:62:D9:17:06:2C:A4:36:91:9D:C5:70:36:6E:99:BE:93:D6:85:55

SHA1-Fingerabdruck
BB:DD:D9:3D:96:02:51:95:A4:91:84:E1:06:4C:9E:A1:12:5F:66:7B

Öffentlicher Schlüssel des Antragstellers
RSA
30:82:01:0A:02:82:01:01:00:A3:DD:69:56:D5:80:CF:1C:1A:5D:5B:C4:B8:95:56:E2:56:F8:78:AE:21:67:61:8F:D5:AB:F8:AF:D7:00:DC:DB:76:29:AD:

Export in Datei

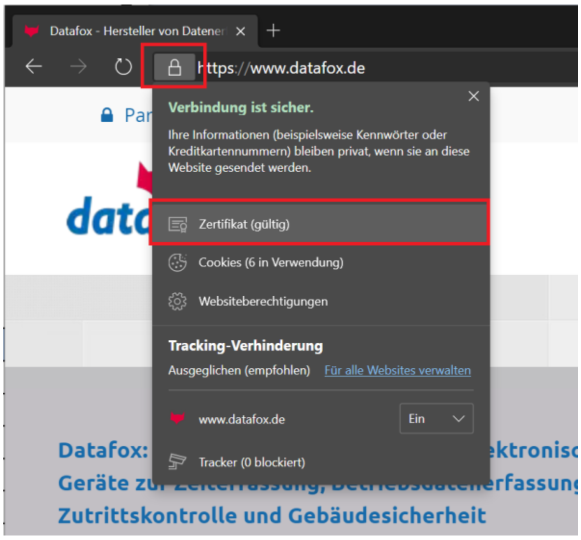
Despite the screen shots being in German language, you can see, that the certificate for “datafox.de” domain is a wildcard certificate. This certificate is derived from “AlphaSSL CA”, which is derived from “GlobalSign Root CA”.
The certificate verification using by TLS tracks the web site’s certificate up to its root certificate. For performing this check, the device needs the “GlobalSign Root CA” certificate to be stored in the device’s filesystem. Please save the certificate using “Export to file” to your Windows PC’s filesystem, e.g., as “GlobalSignRootCA.crt”.
Microsoft Edge saves the certificate as its binary representation (DER format), the device need the certificate to be PEM-encoded. You can use openssl to convert between the two representations:

```
    openssl x509 -inform DER -in GlobalSignRootCA.crt -out GlobalSignRootCA.pem -text
```

The resulting “GlobalSignRootCA.pem” then can be renamed as “GlobalSignRootCA.cert” an transfered to the device using the DatafoxStudioIV.

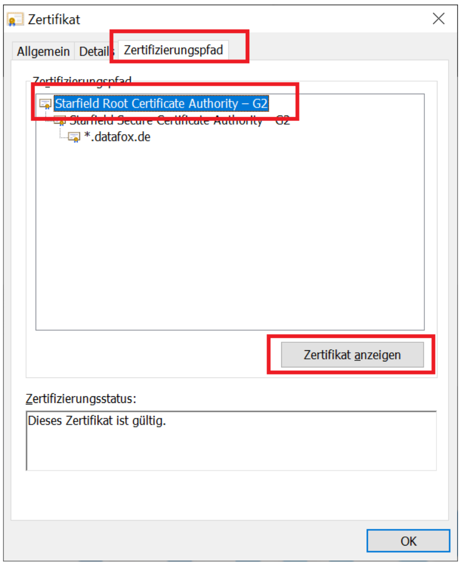
B.5.3: Which certificate is used by the web server? (“Chromium” Edge Browser)

The Chromium Edge Browser allows accessing certificate information of a website using the lock-symbol as well.



Verbindung ist sicher.

Ihre Informationen (beispielsweise Kennwörter oder Kreditkartennummern) bleiben privat, wenn sie an diese Website gesendet werden.



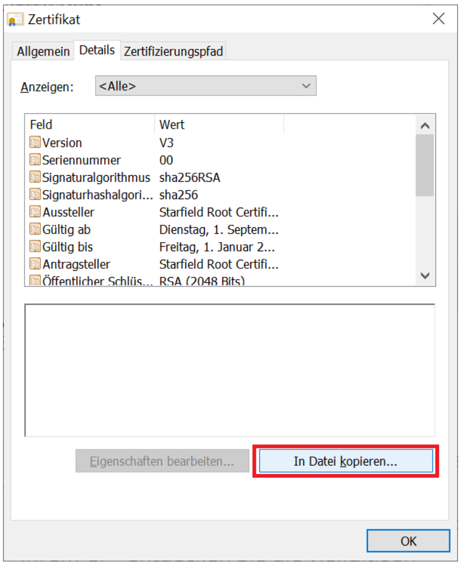
Zertifikat
Allgemein
Details
Zertifizierungspfad
Starfield Root Certificate Authority – G2
Starfield Secure Certificate Authority – G2
*.datafox.de
Zertifikat anzeigen
OK

Zertifizierungsstatus:

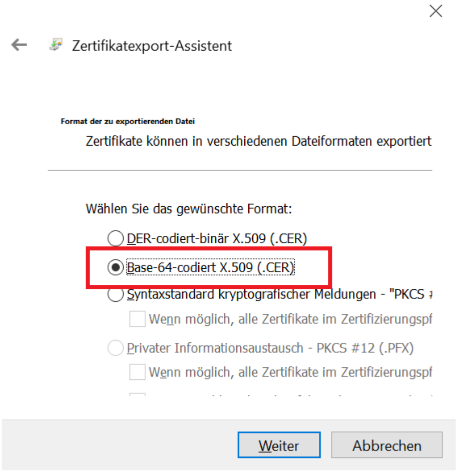
Dieses Zertifikat ist gültig.

The certificate dialog is shown. Please traverse the the certificate path tab page, select the root certificate an active the button „show certificate“

An identical dialog appears that displays only the root certificate. Traverse to the tab „Detail“ as select the button „copy to file“:



Feld	Wert
Version	V3
Seriennummer	00
Signaturalgorithmus	sha256RSA
Signaturhashalgori...	sha256
Aussteller	Starfield Root Certifi...
Gültig ab	Dienstag, 1. Septem...
Gültig bis	Freitag, 1. Januar 2...
Antragsteller	Starfield Root Certifi...
Öffentlicher Schlüs...	RSA (2048 Bits)



Zertifikatexport-Assistent

Format der zu exportierenden Datei

Zertifikate können in verschiedenen Dateiformaten exportiert

Wählen Sie das gewünschte Format:

○ DER-codiert-binär X.509 (.CER)

● Base-64-codiert X.509 (.CER)

○ Syntaxstandard kryptografischer Meldungen - "PKCS #

□ Wenn möglich, alle Zertifikate im Zertifizierungspf

○ Privater Informationsaustausch - PKCS #12 (.PFX)

□ Wenn möglich, alle Zertifikate im Zertifizierungspf

The assistent shown allows to export the certificate. Please be sure to export the certificate as a “Base 64 encoded x.509 certificate”. The so exported certificate is usable for Datafox Devices.

The exported certificate is generated using the file extension “.cer”. This file has to be renamed to end with “.cert” and may be transferred to a device then.

B.5.4: Device configuration – Deploying a client certificate

Please create a key pair for the device first. You may use a self-signed certificate/key-pair here or derive it from an official, CA-signed certificate.

Transfer certificate and key to the device. Please ensure, that the certificate is named “client.cert” and the key is named “client.key”. As soon as both files are present on the device, the TLS handshake will use them as client certificate – should the server ask for it.

> !
>
> **Attention:**
> The security of the “client.key” is of special concern. Thus, we ensure by the device firmware that the key is not read back over the network.
> So, even if the “client.key” has been transferred successfully to the device using Datafox Studio, it **cannot** be read back.

B.6: Creating a private CA

OpenSSL is a very versatile crypto tool and can not only create self-signed certificates, but also generate and sign chains of certificates. This makes it possible to derive several certificates from the same root certificate - since the root certificate is sufficient for certificate verification, you can generate a family of server or client certificates in this way.

First, please install OpenSSL on the system on which you want to create your CA. Windows or Linux systems are equally suitable here; we have not tried this process on Mac OS X.

B.6.1: Creating the root key/certificate of the CA

To do this, proceed as follows:

- First create the root key "`ca.key`". Enter your data as "subject":

  ```
  openssl req -subj "/C=<Country>/ST=<State>/L=<Stadt>/O=<Organisation>
  /OU=<Orga-Unit>/CN=<Common Name>" -new -newkey rsa:2048 -nodes -out ca.csr
  -keyout ca.key
  ```

- Now create the appropriate certificate, which then specifies the maximum validity period for all derived certificates (7300 days is about 20 years):

  ```
  openssl x509 -signkey ca.key -days 7300 -req -in ca.csr -out ca.pem
  ```

- If not explicitly configured otherwise, OpenSSL creates a "demoCA" in which the root certificate is expected and derived certificates are stored. This has the following structure:

  ```
  demoCA/
  demoCA/cacert.pem
  demoCA/index.txt
  demoCA/newcerts/
  demoCA/private/
  demoCA/private/cakey.pem
  demoCA/serial
  ```

  Create the above mentioned directories, the file index.txt should be empty, serial contain the value "01".

- Copy the created root key pair into the demoCA:

  ```
  cp ca.key ./demoCA/private/cakey.pem
  cp ca.pem ./demoCA/cacert.pem
  cp ca.pem ca.cert
  ```

Your CA is now ready for use - you must protect the cakey.pem in the private directory! This can then be used to generate new keys, the `cacert.pem` is needed later to check the derivation chain.

> !
> **Attention:**
> If you lose the key, this is equivalent to losing the complete CA.

B.6.2: Create derived key pairs

Because the `demoCA` is now initialised, "only" three steps are required to create a derived certificate:

- Create a new key
  ```
  openssl genrsa -out derived-a.key 2048
  ```

- Create a certification request
  ```
  openssl req -subj "/C=<Country>/ST=<State>/L=<Stadt>/O=<Organisation>
  /OU=<Orga-Unit>/CN=<Common Name>" -new -key derived-a.key -out derived-
  a.csr
  ```

- Generate the signed certificate (730 days => 2 years validity)
  ```
  openssl ca -in derived-a.csr -days 730 -out derived-a.cer
  ```

Thus, with `derived-a.key` and `derived-a.cer`, a key pair is now available that is derived from `cakey.pem`. To check the correctness of the derivation chain, it is necessary that `cacert.pem` is known as a trusted certificate - under Windows, it can be imported into the local keystore, for example - in the case of a Datafox device, it must be transferred to the certificate store in the device.

> ☞
> **Please note:**
> Of course, you can now create any number of certificates derived from the CA root certificate. Since OpenSSL manages these internally, it is necessary that this I differ with regard to the subject.

B.7: Analysis of certificates

Microsoft Windows is shipping a command line tool named `CertUtil`. This programme can be used to show the content of certificate files. For printing out the certificate's content, the certificate file has to be decoded first, the second instruction will then print the content:

```
> CertUtil -decode my.cert my.crt
Eingabelänge = 1440
Ausgabelänge = 1021
CertUtil: -decode-Befehl wurde erfolgreich ausgeführt.

> CertUtil my.crt
X.509-Zertifikat:
Version: 3
Seriennummer: c4124040d28438f6
Signaturalgorithmus:
    Algorithmus Objekt-ID: 1.2.840.113549.1.1.11 sha256RSA
    Algorithmusparameter:
    05 00
Aussteller:
    E=s.meyer@datafox.de
    CN=Sven Meyer
    OU=Development
    O=Datafox
    L=Geisa
```

```
    S=Thueringen
    C=DE
  Namenshash (sha1): a12670bfda2ef055e608e130abab20741390a5d5
  Namenshash (md5): acc8642302e9f97b271419d7b06149eb

 Nicht vor: 27.09.2018 14:28
 Nicht nach: 27.09.2019 14:28

Antragsteller:
    E=s.meyer@datafox.de
    CN=Sven Meyer
    OU=Development
    O=Datafox
    L=Geisa
    S=Thueringen
    C=DE
  Namenshash (sha1): a12670bfda2ef055e608e130abab20741390a5d5
  Namenshash (md5): acc8642302e9f97b271419d7b06149eb

Öffentlicher Schlüssel-Algorithmus:
    Algorithmus Objekt-ID: 1.2.840.113549.1.1.1 RSA (RSA_SIGN)
    Algorithmusparameter:
    05 00
Länge des öffentlichen Schlüssels: 2048 Bits
Öffentlicher Schlüssel: Nicht verwendete Bits = 0
    0000  30 82 01 0a 02 82 01 01  00 ce 4b 2d 46 a8 05 75
    0010  73 d8 1c 88 49 97 64 0c  09 b0 96 0b 56 49 76 f0
    0020  1d 49 63 aa 80 cf 93 23  72 88 68 d6 ab 49 ba 7e
    0030  81 56 ae 57 21 d7 39 0b  f8 a1 e0 91 88 7e 9f d1
    0040  cb 32 ce c5 02 98 e0 e3  a2 17 0f c5 c1 0e 7a 57
    0050  d7 4b 11 16 b3 8a f5 ac  f1 b0 22 9f 75 4a e5 9a
    0060  9c 51 75 72 3b ea cb f3  94 6d 7e fb b0 d5 12 2d
    0070  1e e8 76 cf 70 42 69 94  71 89 34 f3 0c d7 bf 9a
    0080  ba 11 79 85 03 1d 46 01  00 2c 1a af ba 8c 7e 91
    0090  f2 a6 a0 d4 40 4e eb c6  10 6d 7a f9 3c f4 5f 1a
    00a0  55 77 20 19 6f 5c 42 76  44 51 ad a8 16 c1 3f e9
    00b0  96 0c 20 b7 f2 9f 6c 0e  7f 68 00 64 45 da 8b d3
    00c0  5c 2e 31 ed 63 01 cf 64  ea 52 d9 aa 44 b8 e9 15
    00d0  94 ea b0 2e 3a aa 5d 68  5a 13 d8 b1 de 68 2b f1
    00e0  7a a4 b8 ad 31 a8 f4 c3  62 20 ee 32 59 6e 33 6c
    00f0  1a 28 15 e9 13 27 e9 f6  18 94 44 cd 6b 64 b9 3d
    0100  a9 2c 9b c4 d0 1c 7b 77  71 02 03 01 00 01
Zertifikaterweiterungen: 3
    2.5.29.14: Kennzeichen = 0, Länge = 16
    Schlüsselkennung des Antragstellers
        480aacdb31e748625f02ae38aaab7a228722fb93

    2.5.29.35: Kennzeichen = 0, Länge = 18
    Stellenschlüsselkennung
        Schlüssel-ID=480aacdb31e748625f02ae38aaab7a228722fb93

    2.5.29.19: Kennzeichen = 0, Länge = 5
    Basiseinschränkungen
        Typ des Antragstellers=Zertifizierungsstelle
        Einschränkung der Pfadlänge=Keine

Signaturalgorithmus:
    Algorithmus Objekt-ID: 1.2.840.113549.1.1.11 sha256RSA
    Algorithmusparameter:
    05 00
Signatur: Nicht verwendete Bits=0
    0000  e0 63 94 47 a9 c0 e5 22  e2 ba 6e 7a 81 23 1f d7
    0010  91 96 94 77 0c 3d 40 33  e1 9f 4e 35 e6 f6 76 51
    0020  9e 45 1e b9 63 01 f4 6a  c4 04 06 5d a9 5c 10 be
    0030  b5 72 6e fd 0e ed 92 e7  eb 18 50 39 32 93 e2 55
    0040  1b 1d f4 a3 dd f3 28 6f  b0 fa 7f 88 85 9f 40 e0
    0050  90 9e 56 37 93 06 a6 0f  79 5e 9f f0 ef e4 36 55
    0060  85 a5 03 de aa 00 87 2b  b3 43 d1 20 14 51 ea a6
    0070  18 d8 a0 7d 8f 19 de 51  d5 54 02 c5 7a 92 39 52
    0080  84 ab 11 df b9 2f 78 9e  1f c5 f1 d9 b7 42 a6 0e
    0090  9a 84 3c 7f 56 05 81 c5  ac 4f 2e 99 39 77 88 84
    00a0  bd f2 c9 4b f0 a8 0b 58  83 bb d0 22 d4 5f 74 67
    00b0  45 5f 35 cf 90 0a 58 00  d1 05 60 38 ab 7b 0a 56
    00c0  2e 68 1c 4f 03 f6 7a 56  51 0a 38 65 a0 f2 e3 31
    00d0  c0 71 86 2e 06 d9 b0 a3  da 9a 23 45 5b 61 9e 1d
    00e0  7d 92 b0 1c b4 32 6d 80  e5 08 1e 14 05 9d 0d 40
    00f0  c7 c2 69 1b e8 81 d4 db  12 f5 36 77 e6 8e 27 80
Signatur stimmt mit dem öffentlichen Schlüssel überein.
Stammzertifikat: Antragsteller stimmt mit Aussteller überein
Schlüssel-ID-Hash(rfc-sha1): 480aacdb31e748625f02ae38aaab7a228722fb93
Schlüssel-ID-Hash(sha1): 09d0b7592a814746a0763cd728dadd7a63a6f3c7
Schlüssel-ID-Hash(bcrypt-sha1): 052b05f39aae0645b961e889c512889baa633aa0
Schlüssel-ID-Hash(bcrypt-sha256):
08beebf2f0d0b0cf4a857388dfb6a9ecda6f073665d7f44804e9552e16d469f1
Schlüssel-ID-Hash(md5): e3606281f405dc9bc9185609a419d76d
Schlüssel-ID-Hash(sha256):
29d0ee85e0290b1f3b13deda03a67bb824cc598c7e1ba4fbf5035f2290b8ecf1
Schlüssel-ID-Hash(pin-sha256): LSyEenteNDnDtS6o/57zWVbDOpCaOIOoyaNpcNfFuNQ=
Schlüssel-ID-Hash(pin-sha256-hex):
2d2c847a7b5e3439c3b52ea8ff9ef35956c33a909a3883a8c9a36970d7c5b8d4
Zertifikathash(md5): ce42b996362553fa26c594ad1ee81a24
Zertifikathash(sha1): fae481cd6b0e846ff5df4360844a013cac36d36c
Zertifikathash(sha256):
ea9dd651d5be4918bef5c699a444fca600129290ddc36def717b76004f0c2762
Signaturhash: 2f818c6fc579789dd464c4205eb8ceedf5568b483406fab4b9cae8b80e450684
CertUtil: -dump-Befehl wurde erfolgreich ausgeführt.
```

B.8: Limitations of the Implementation

The current implementation of HTTPS in Datafox Devices has some limitations. These are:

- TLS 1.1 and TLS 1.2 are supported.
- The current implementation does not support TLS 1.3.
- The length of an RSA key may not be bigger than 2048 bits.
- If you need stronger security, consider using ECC with key length of 256 bit.

B.9: Additional Information

There are lots of additional documentation available. Please consider the following pages to dig deeper into the matter of https communication:

- https://tools.ietf.org/html/rfc2818
- https://en.wikipedia.org/wiki/HTTPS
- https://robertheaton.com/2014/03/27/how-does-https-actually-work/

Appendix C: Initial device configuration using http

C.1: Sending info telegrams with configuration data cyclically

Using the extended download methods described in Appendix A, the http protocol provides all functions needed to set up a MasterIV device.

The concept requires a minimal device configuration that allows sending an info telegram (see df_kvp=info in section 2.2.3.2.14). The configuration is done using system variables – the device will send an info request to the configured server after being physically installed.

The following system variables control the info telegram behaviour:

System Variable Name	Description
http.config.mode	Number defining the period of sending the info telegram 1 = daily, 2 = weekly 4 = monthly (28 days) The first telegram is sent about 30 minutes after system start.
http.config.host	Host which receives the telegram
http.config.port	Port at host to which the telegram is sent
http.config.send	URL at the webserver receiving the info telegram

The timing of sending the info telegram depends on the device being configured by a setup at system start time:

- With setup: approximately 30 minutes after start up, the according to http.config.mode.
- Without setup: approximately 30 seconds after start up (“emergency mode”), then approximately every 10 minutes.

In this mode the device will try to create an active mode connection to establish a service connection – provided an active mode server is configured. This may lead to some jitter in the timing, so that the info telegram is not sent exactly 30 seconds after device start-up.

C.2: CRC Implementation of the info telegram

The info telegram uses a 32bit checksum that is computed using the CRC algorithm. The algorithm uses

- Initial: `0xffffffff`
- Polynomial: `0x04c11db7`

As parameters, reflect is not applied.

Example: The CRC computed for the string “123456789” is `0x340BC6D9`.

C.3: Use Case: Monitoring and Updating device certificates

The cyclic info telegram may be used to key an eye on the certificates deployed to devices. In addition to the device type and the serial number a list of certificate file names and their checksums is sent with the info telegram – this allows identifying the certificates on the device.

The following cases may occur:

- You are not planning a change of your server’s certificate chain and the root certificate from which your server’s chain of certificates is derived is valid “long enough”
  - No action is required
- You plan a change to your server’s certificate chain
  - Send the new root certificate to the device

As result from previous considerations, you derive if you want to remove an existing certificate or deploy a new certificate to the device. However, the scenario of exchanging a root certificate is expected to occur only rarely since top level certificates of a CA typically have a very long validity period.

Removing a certificate may be triggered directly by sending a `df_remove_file` (see 2.2.3.2.19) command. The deleted certificate will be available to the device until the communication is restarted (e.g. by using a service mode command (see 2.2.3.2.3) or by restarting the device).

In order to transfer a certificate to a device, wrap it within a transfer file (file type `0xDF00`, see Appending A.3.2), assign a name ending with “`.cert`” and send the file to the device (either as protocol level answer (see 2.2.3.1.1) or have it downloaded by the device by sending a `df_load_file`.

Appendix D: Test server application with http integration

The ease integration of the http/https interface, Datafox provides its internal test server application. This application may be used to test-drive the communication between a Datafox Device and a web server and the test out the terminal’s behaviour to commands sent by the server.

Attention: The application is provided AS-IS. It does not come with any support or warranty.

> ☞ **Please note:**
> Datafox provides – as an alternative to the test server described in this chapter – on online test environment. This environment is described at
>
> https://www.datafox.de/support/testumgebungen
>
> and is accessible from the internet.

The test server is continuously being expanded. The reference version is available at

https://www.datafox.de/download/dist-DFWebServer-current.zip

ready for download. Please note that the application is a test application intended for internal purposes and is provided without warranty claims.

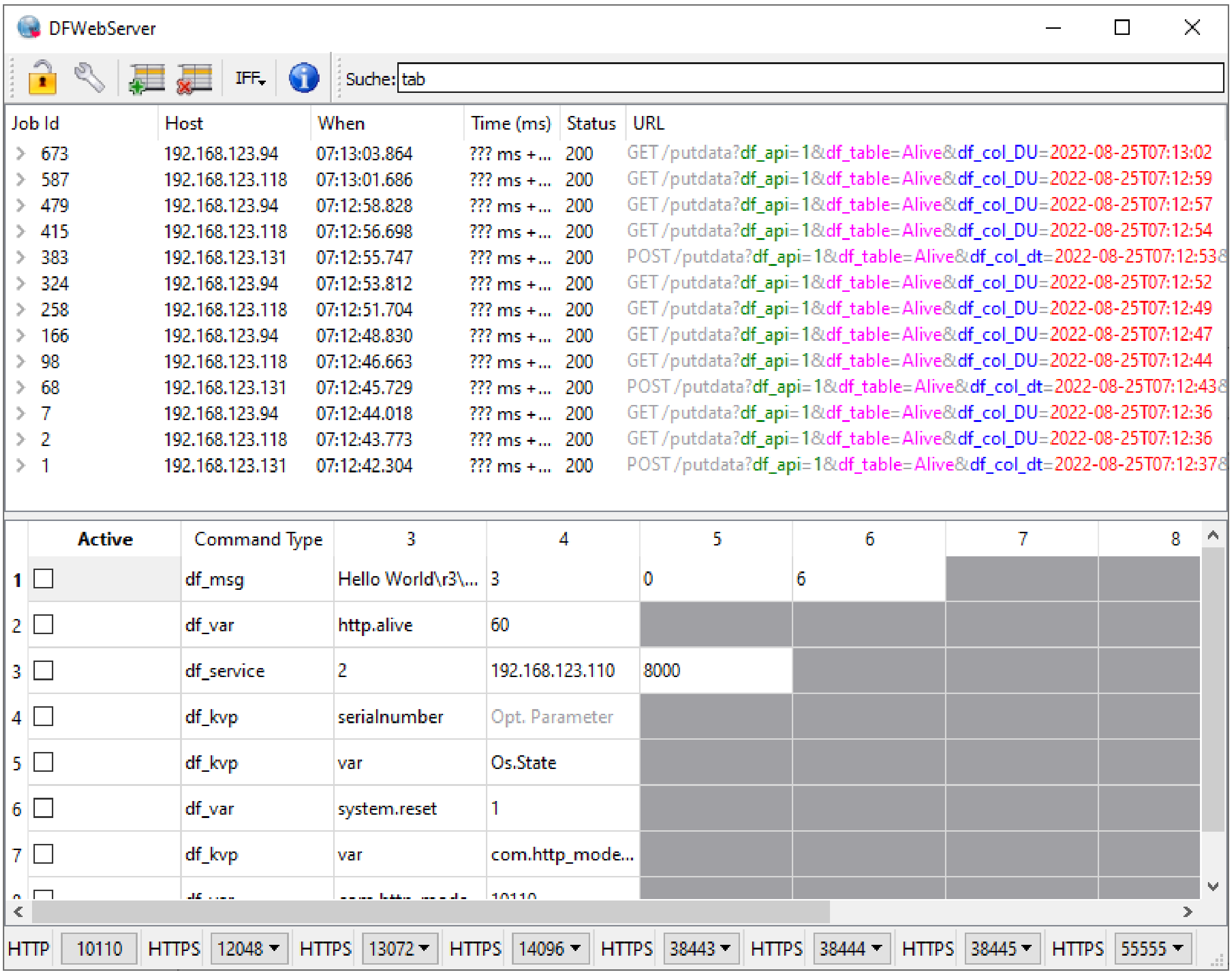
D.1: The User interface

After starting the test servers displays the following window:

Configuration

Add / Remove instructions

Sequence of requests are responds



DFWebServer

Job Id	Host	When	Time (ms)	Status	URL
673	192.168.123.94	07:13:03.864	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:13:02
587	192.168.123.118	07:13:01.686	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:59
479	192.168.123.94	07:12:58.828	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:57
415	192.168.123.118	07:12:56.698	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:54
383	192.168.123.131	07:12:55.747	??? ms +...	200	POST /putdata?df_api=1&df_table=Alive&df_col_dt=2022-08-25T07:12:53&
324	192.168.123.94	07:12:53.812	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:52
258	192.168.123.118	07:12:51.704	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:49
166	192.168.123.94	07:12:48.830	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:47
98	192.168.123.118	07:12:46.663	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:44
68	192.168.123.131	07:12:45.729	??? ms +...	200	POST /putdata?df_api=1&df_table=Alive&df_col_dt=2022-08-25T07:12:43&
7	192.168.123.94	07:12:44.018	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:36
2	192.168.123.118	07:12:43.773	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:12:36
1	192.168.123.131	07:12:42.304	??? ms +...	200	POST /putdata?df_api=1&df_table=Alive&df_col_dt=2022-08-25T07:12:37&

	Active	Command Type	3	4	5	6	7	8
1	□	df_msg	Hello World\r3\...	3	0	6		
2	□	df_var	http.alive	60				
3	□	df_service	2	192.168.123.110	8000			
4	□	df_kvp	serialnumber	Opt. Parameter				
5	□	df_kvp	var	Os.State				
6	□	df_var	system.reset	1				
7	□	df_kvp	var	com.http_mode...				
8	□	df_var	com.http_mode	10110				

HTTP 10110 HTTPS 12048 ▾ HTTPS 13072 ▾ HTTPS 14096 ▾ HTTPS 38443 ▾ HTTPS 38444 ▾ HTTPS 38445 ▾ HTTPS 55555 ▾

List of command that can be sent to the terminal.

The server's current list of ports

10110 is a non-https server, the other ports are server by HTTPS servers.

If a port configuration has an error, there is a tooltip shown at the red cross indicating the error.

The web server accepts data records on all properly configured server ports. You can adjust the configuration of the server ports using the configuration files settings-<*n*>.ini (n=0,…,9).

The settings-file has the following structure:

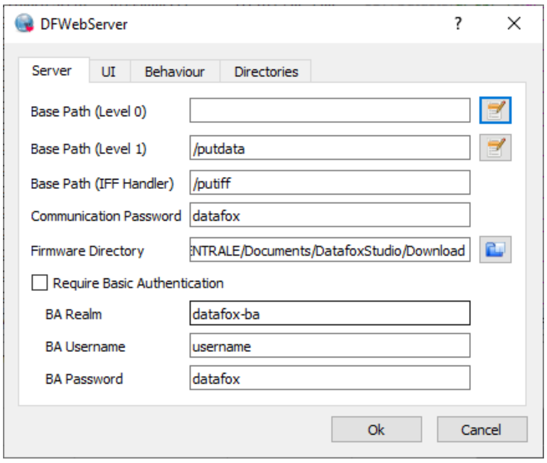
```
[General]
port=8443
minThreads=1
maxThreads=10
cleanupInterval=1000
readTimeout=60000
sslKeyFile=ssl/my.key
sslCertFile=ssl/my.cert
maxRequestSize=16000
maxMultiPartSize=1000000
```

Removing the lines `sslKeyFile` and `sslCertFile` from the configuration files will result in a non-encrypted, plain http server.

D.2: Webserver configuration

The configuration dialog allows adjusting settings concerning the server, the UI, the behaviour or the directories.

D.2.1: Server



Base Path (Level 0)
Base Path (Level 1) /putdata
Base Path (IFF Handler) /putiff
Communication Password datafox
Firmware Directory :NTRALE/Documents/DatafoxStudio/Download
☐ Require Basic Authentication
BA Realm datafox-ba
BA Username username
BA Password datafox

To adjust the server the following settings are available:

You may assign different application-level server paths for processing API-Level 0 (see 2.1) and API-Level 1 (see 2.2) requests.

The **base path** is part of the URL which has to send by the device. Above screenshot indicates that API-Level 1 request will have to be sent using “/putdata” as request path in order to be processed by the server.

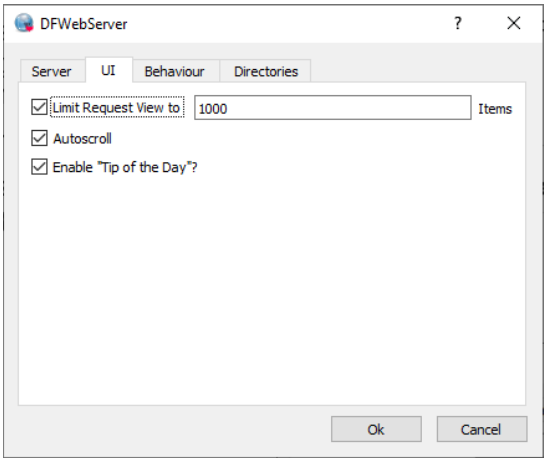
The base path (IFF handler) defines an endpoint which processes IFF content sent from a device automatically. The IFF structure is analysed and shown.

The **communication password** for RC4 based encryption of field content may be set here. Please consider using TLS as channel encryption (“HTTPS”) – RC4 is present here only for backward compatibility reasons.

Firmware-Directory points to a directory in the filesystem containing DFZ files. You can use e.g., the download-directory from Datafox Studio.

Basic authentication may be configured through these options. The server then requires the credentials for accessing its content.

D.2.2: User Interface (UI)



☑ Limit Request View to 1000 Items

☑ Autoscroll

☑ Enable "Tip of the Day"?

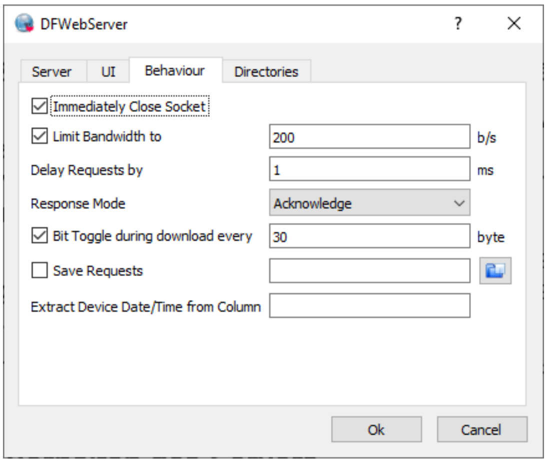
For the UI section you can adjust the following settings.

You may – due to performance considerations – **limit** the number of **requests shown** in the user interface. A reasonable setting will depend on the host's computational power as well as on the use case. During development it is typically not required to retain data that has been produced may hours ago.

Using the **autoscroll** option, you can define if a request currently selected shall be visible when newer data arrives – or may be scrolled out of the visible area.

The **Tip of the day** may be deactivated or activated here.

D.2.3: Behaviour



☑ Immediately Close Socket

☑ Limit Bandwidth to 200 b/s

Delay Requests by 1 ms

Response Mode Acknowledge

☑ Bit Toggle during download every 30 byte

☐ Save Requests

Extract Device Date/Time from Column

The server's behaviour may be adjusted using the following settings:

The server may include the request to **close the connection** directly with the answer to the client. For HTTP/1.1 this is no typical behaviour – and is intended for development purposes. When activating this setting, the connection on socket basis is not closed by the server directly, the server sends the “Connection: Close” Header to the client instead.

If the **bandwidth limitation** is activated, the server sends its answer packages slower that the physical bandwidth would permit.

The **delay** between receiving a **request** and sending the **response** may be specified in milliseconds. You may use this setting to simulate system load on the server side.
Please keep in mind that typically the TCP timeout will tear down a connection after 20 seconds automatically if idle. Then the server cannot send an answer to the device any more.

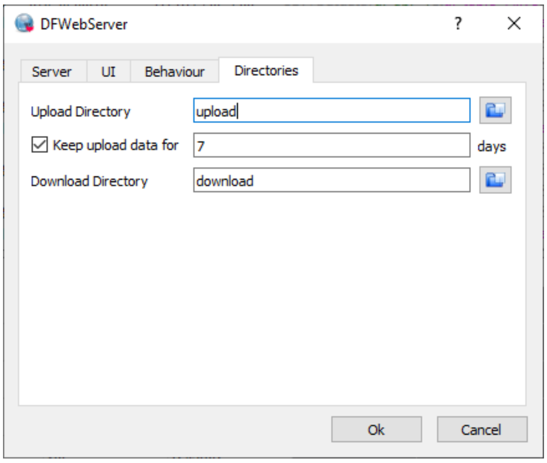
Additionally, you may specify how the server **responds to a request**. Typically, the server sends an acknowledge as response, but you can select that the server does not answer at all (so that the connection remains open until the TCP timeout tears it down) or deliberately sends no acknowledge.

In order to simulate download errors, you may choose to have the server to **toggle a bit every** now and then within the download content. Don’t use this function.

Should you need to record requests, you may choose to have the server **save** these requests. Please keep in mind, that the request URL is stored only – the body of a POST request is not stored in this way.

With the option to **Extract Date/Time** from a data record, the time delay of a request may be examined. The server determines the creation timestamp from the content of the designated column (the device’s clock and the PC clock should be aligned properly)

D.2.4: Directories



Upload Directory: upload
☑ Keep upload data for: 7 days
Download Directory: download

Using the directory settings, you may specify which folders on the server’s hard disk are used as up- and download directories. At the http side these directories may be accessed as “/upload-area/” or “/download-area/”.

The **upload directory** is used to store files the device sends to the server. If this path does not contain a drive letter, the path is considered to be relative to the current working directory. The content of the directory is addressed as “/upload-area/<filename>” on the http side.
Remark: The “/upload-area/” is a definition of the DFWebServer application – your application may use different paths here.

The Webserver may cyclically **clean up** the upload directory and remove old files. You may specify the minimal age of a file to be cleaned.

The **download directory** is a directory where the device may download data from (analogue to the upload directory). On http level the directory is addressed by “/download-area/”.

Remark:

- The test server does not implement reliable communication cipher using `df_cb` and `df_ce`.

D.3: Processing of requests

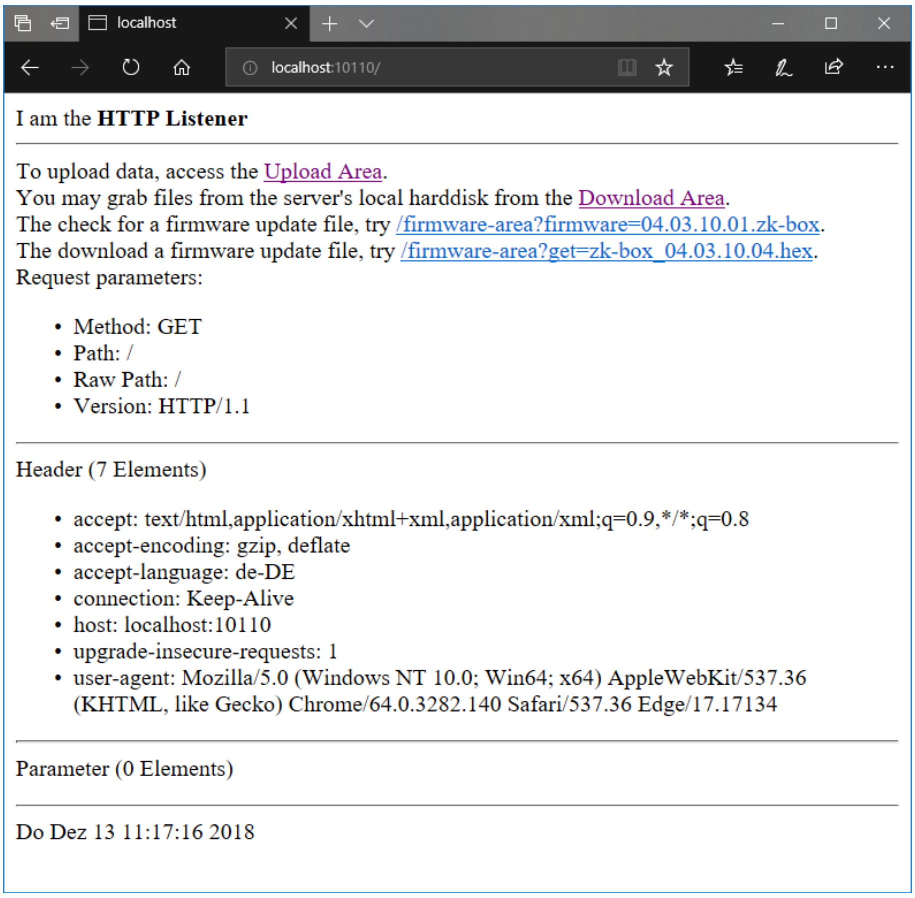
The http and https server port listeners will react to an incoming http request. As soon as a Datafox Device is configured with the server’s communication endpoint and a data record is created, it will be shown in the main windows upper section.

The server will process the settings from the main window’s lower section and create the answer sent to the device. Above screenshot will contain – additionally to the acknowledgement of the server receiving the dataset (`df_api=1`) and beep instruction (`df_beep=1`) – please see 2.2.3.2 for details.

D.4: IFF files inside the web server

D.4.1: Analysis of IFF files

The web server is prepared for processing IFF files. These are typically send by the device due to `df_send_file` instructions (see 2.2.3.2.18). To experiment with the upload processing, please direct your web browser directly to any of the server’s listener ports:



I am the **HTTP Listener**

To upload data, access the Upload Area.
You may grab files from the server's local harddisk from the Download Area.
The check for a firmware update file, try /firmware-area?firmware=04.03.10.01.zk-box.
The download a firmware update file, try /firmware-area?get=zk-box_04.03.10.04.hex.
Request parameters:

- Method: GET
- Path: /
- Raw Path: /
- Version: HTTP/1.1

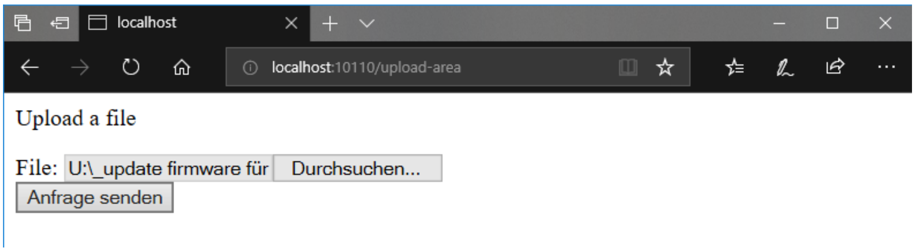
Header (7 Elements)

- accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
- accept-encoding: gzip, deflate
- accept-language: de-DE
- connection: Keep-Alive
- host: localhost:10110
- upgrade-insecure-requests: 1
- user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134

Parameter (0 Elements)

Do Dez 13 11:17:16 2018

Above shown web page is an internal test page of the web server. You can use this to analyse you IFF files – not depending on whether they have been created by a device or by your service. Select the “Upload Area” hyperlink in above webpage:

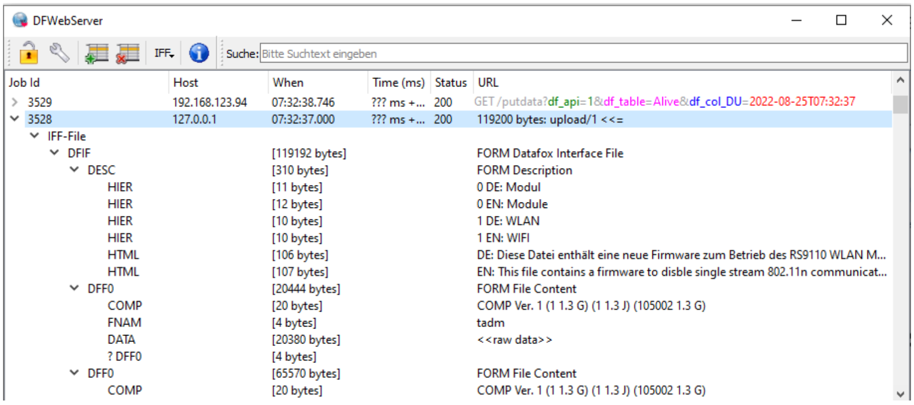


localhost:10110/upload-area

Upload a file

File: U:_update firmware für

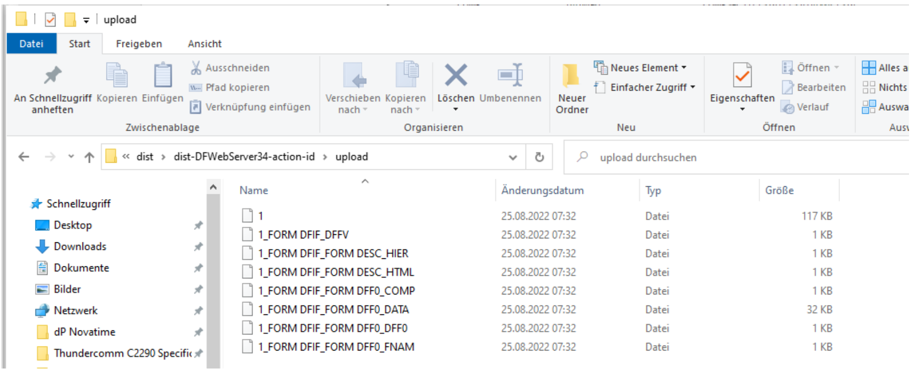
After sending the request, the web server analyses the IFF frame and displays it as following:



Suche: Bitte Suchtext eingeben

Job Id	Host	When	Time (ms)	Status	URL
3529	192.168.123.94	07:32:38.746	??? ms +...	200	GET /putdata?df_api=1&df_table=Alive&df_col_DU=2022-08-25T07:32:37
3528	127.0.0.1	07:32:37.000	??? ms +...	200	119200 bytes: upload/1 <<=
IFF-File					
DFIF		[119192 bytes]			FORM Datafox Interface File
DESC		[310 bytes]			FORM Description
HIER		[11 bytes]			0 DE: Modul
HIER		[12 bytes]			0 EN: Module
HIER		[10 bytes]			1 DE: WLAN
HIER		[10 bytes]			1 EN: WIFI
HTML		[106 bytes]			DE: Diese Datei enthält eine neue Firmware zum Betrieb des RS9110 WLAN M...
HTML		[107 bytes]			EN: This file contains a firmware to disble single stream 802.11n communicat...
DFF0		[20444 bytes]			FORM File Content
COMP		[20 bytes]			COMP Ver. 1 (1 1.3 G) (1 1.3 J) (105002 1.3 G)
FNAM		[4 bytes]			tadm
DATA		[20380 bytes]			<<raw data>>
? DFF0		[4 bytes]			
DFF0		[65570 bytes]			FORM File Content
COMP		[20 bytes]			COMP Ver. 1 (1 1.3 G) (1 1.3 J) (105002 1.3 G)

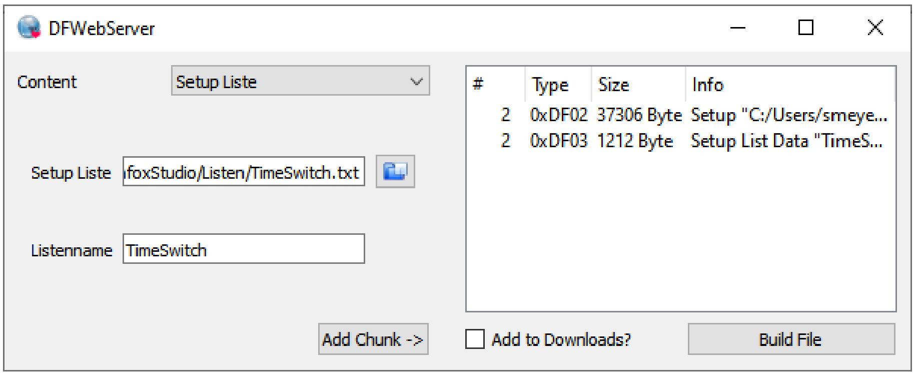
The content of the IFF file is separated according to the sections and saved – along with the IFF file – into the upload directory:



Name	Änderungsdatum	Typ	Größe
1	25.08.2022 07:32	Datei	117 KB
1_FORM DFIF_DFFV	25.08.2022 07:32	Datei	1 KB
1_FORM DFIF_FORM DESC_HIER	25.08.2022 07:32	Datei	1 KB
1_FORM DFIF_FORM DESC_HTML	25.08.2022 07:32	Datei	1 KB
1_FORM DFIF_FORM DFF0_COMP	25.08.2022 07:32	Datei	1 KB
1_FORM DFIF_FORM DFF0_DATA	25.08.2022 07:32	Datei	32 KB
1_FORM DFIF_FORM DFF0_DFF0	25.08.2022 07:32	Datei	1 KB
1_FORM DFIF_FORM DFF0_FNAM	25.08.2022 07:32	Datei	1 KB

D.4.2: Creating an IFF file

The webserver may be used to create IFF files for sending them to devices. The functionality is available using the IFF menu from the toolbar:



Content: Setup Liste

Setup Liste: foxStudio/Listen/TimeSwitch.txt

Listenname: TimeSwitch

☐ Add to Downloads?

#	Type	Size	Info
2	0xDF02	37306 Byte	Setup "C:/Users/smeye...
2	0xDF03	1212 Byte	Setup List Data "TimeS...

The process of creating an IFF file is split into two sub setups. Firstly, the content is assembled, then the file is built.

At the left-handed side, you may choose content to be stored into the IFF file. By clicking the button “Add Chunk ->” the current content is prepared for transfer and moved to the list on the right side.

If all elements to be stored in the IFF File have be collected, you may activate the button “Build File” to create the IFF file. By checking “Add to Downloads?” you may automatically create a command inside the web server’s command table to instruct the device to download the IFF file.

D.5: Working and updating server certificates

The webserver ships with a set of self-signed certificates, so that I can be used easily. These certificates are RSA as well as ECC key/certificate pairs.

You may – should the certificates have expired – recreate these using the “gen_certificate.sh” script – provided you have OpenSSL and a Unix command prompt (e.g. Cygwin) available.

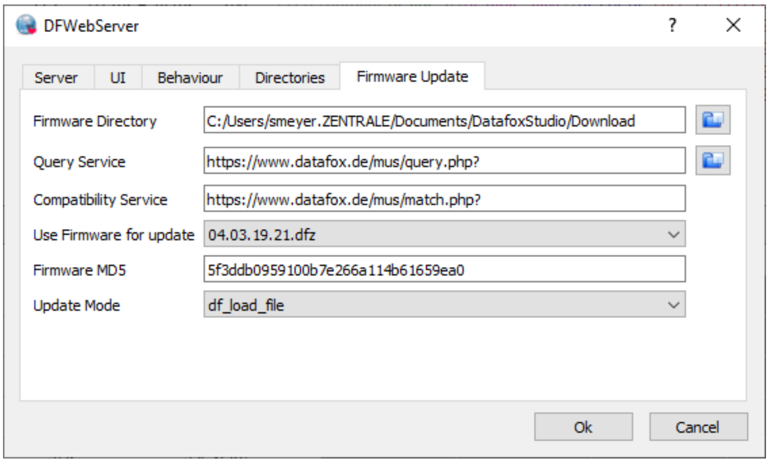
D.6: Firmware-Update using the web server

The test server implements a mechanism for experimenting with the firmware update using HTTP. For using this mechanism two things are required:

- The device has to send an “extinfo” telegram to the web server
- The web server has to be configured for firmware update.

Configuration of the web server

The update service configuration is present at a dedicated tab within the configuration dialog. A sample setup may look as following:



DFWebServer

Firmware Directory: C:/Users/smeyer.ZENTRALE/Documents/DatafoxStudio/Download

Query Service: https://www.datafox.de/mus/query.php?

Compatibility Service: https://www.datafox.de/mus/match.php?

Use Firmware for update: 04.03.19.21.dfz

Firmware MD5: 5f3ddb0959100b7e266a114b61659ea0

Update Mode: df_load_file

- The **Firmware Directory** is the directory where DFZ files are located. Using the download directory from Datafox Studio here might be a good idea.
- The **Query Service** is part of the mechanism described in Appendix E for determining the compatible firmware.

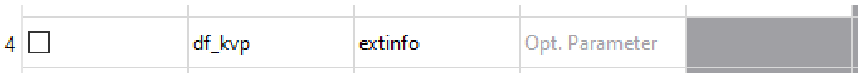
  This service accessible for own development and demonstration purposes though the Datafox webserver.

  Please **do not use** this service for development or production purposes.
- The **Match Service** is part of the mechanism from Appendix E for choosing the correct firmware as well.

  Please **do not use** this service for development or production purposes.
- As soon as quey and match services are configured, you can use the button next to the query service to fill the selection box “**Use Firmware for Update**”. Choosing a firmware here will lead to the server looking for the appropriate file in the “Firmware directory” and filling out the “**Firmware MD5**” field accordingly.
- The final configuration parameter allows specifying the **Update Mode**. The following settings are available:
  - „Direct IFF Data provisioning“:
    The firmware as IFF is sent directly as reponse.
  - „df_load_firmware“:
    The update service sends a “df_load_firmware” command to the device. The device the downloads the specified firmware directly from the configured firmware update server.
  - „df_load_file“:
    The update service sends a “df_load_file” command. This leads to a download request sent from the device to the web server. The web server then delivers the firmware connect.

Requesting an “extinfo” telegram

If above mentioned settings are done, you may request an “extinfo” from a device using the following entry to the command list.



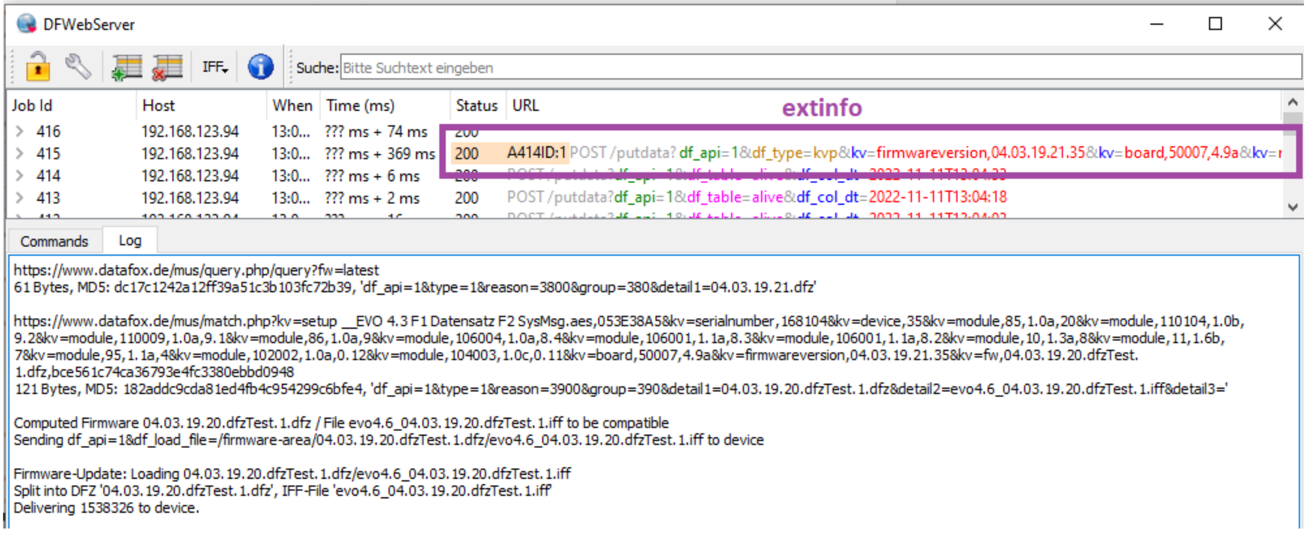
4	☐	df_kvp	extinfo	Opt. Parameter	

The arrival of the extinfo response will trigger the firmware update workflow:

- Checking of the device and the chosen firmware are compatible.
- Delivering the firmware according to the update settings.

Protocol

The steps of the update process are shown in the “Log” tab:



Job Id	Host	When	Time (ms)	Status	URL
416	192.168.123.94	13:0...	??? ms + 74 ms	200	
415	192.168.123.94	13:0...	??? ms + 369 ms	200	A414ID:1 POST /putdata? df_api=1&df_type=kvp&kv=firmwareversion,04.03.19.21.35&kv=board,50007,4.9a&kv=r
414	192.168.123.94	13:0...	??? ms + 6 ms	200	POST /putdata?df_api=1&df_table=alive&df_col_dt=2022-11-11T13:04:33
413	192.168.123.94	13:0...	??? ms + 2 ms	200	POST /putdata?df_api=1&df_table=alive&df_col_dt=2022-11-11T13:04:18
412	192.168.123.94	13:0...	??? ... 16 ...	200	POST /putdata?df_api=1&df_table=alive&df_col_dt=2022-11-11T13:04:03

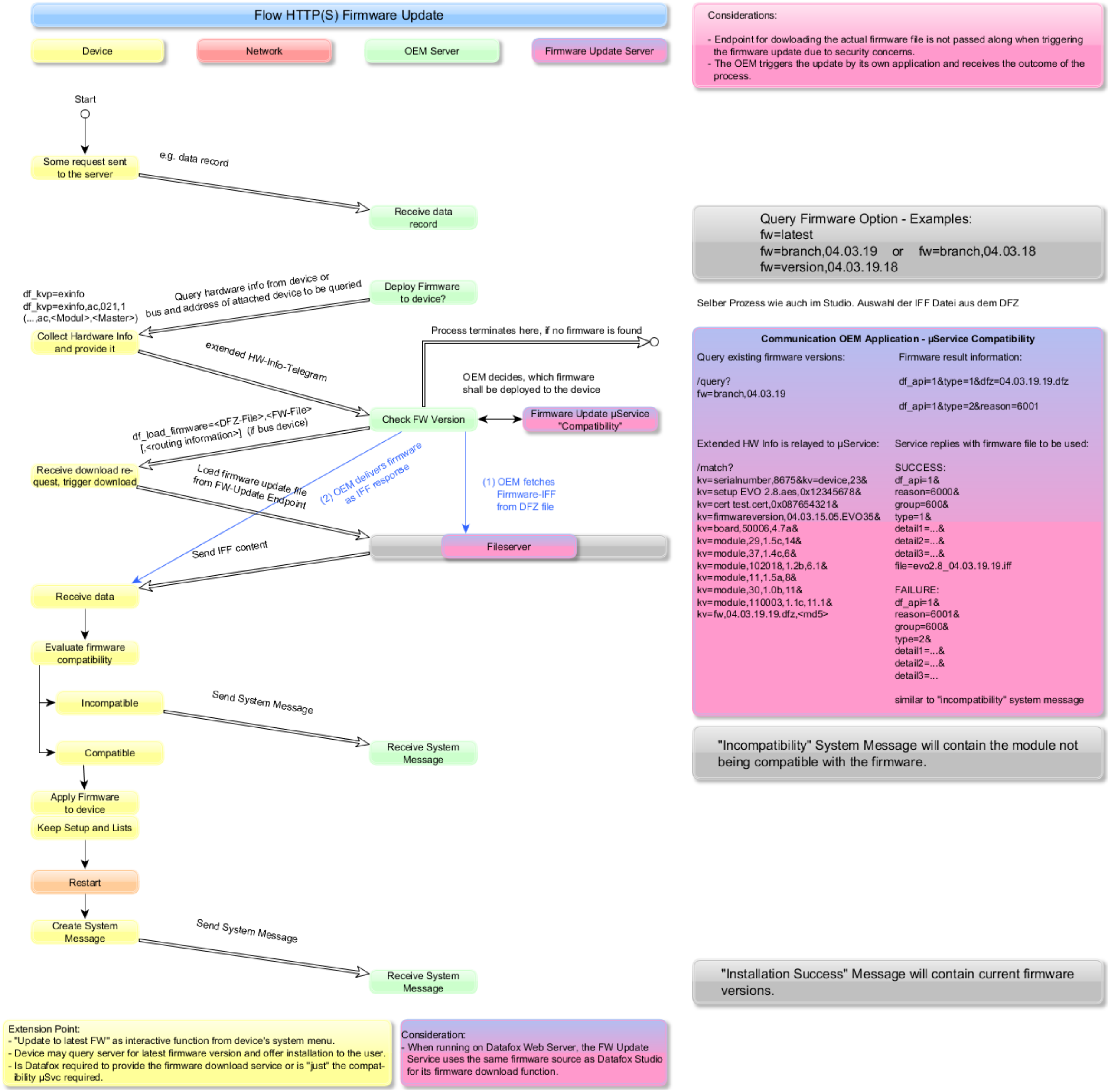
```
https://www.datafox.de/mus/query.php/query?fw=latest
61 Bytes, MD5: dc17c1242a12ff39a51c3b103fc72b39, 'df_api=1&type=1&reason=3800&group=380&detail1=04.03.19.21.dfz'

https://www.datafox.de/mus/match.php?kv=setup __EVO 4.3 F1 Datensatz F2 SysMsg.aes,053E38A5&kv=serialnumber,168104&kv=device,35&kv=module,85,1.0a,20&kv=module,110104,1.0b,
9.2&kv=module,110009,1.0a,9.1&kv=module,86,1.0a,9&kv=module,106004,1.0a,8.4&kv=module,106001,1.1a,8.3&kv=module,106001,1.1a,8.2&kv=module,10,1.3a,8&kv=module,11,1.6b,
7&kv=module,95,1.1a,4&kv=module,102002,1.0a,0.12&kv=module,104003,1.0c,0.11&kv=board,50007,4.9a&kv=firmwareversion,04.03.19.21.35&kv=fw,04.03.19.20.dfzTest.
1.dfz,bce561c74ca36793e4fc3380ebbd0948
121 Bytes, MD5: 182addc9cda81ed4fb4c954299c6bfe4, 'df_api=1&type=1&reason=3900&group=390&detail1=04.03.19.20.dfzTest.1.dfz&detail2=evo4.6_04.03.19.20.dfzTest.1.iff&detail3='

Computed Firmware 04.03.19.20.dfzTest.1.dfz / File evo4.6_04.03.19.20.dfzTest.1.iff to be compatible
Sending df_api=1&df_load_file=/firmware-area/04.03.19.20.dfzTest.1.dfz/evo4.6_04.03.19.20.dfzTest.1.iff to device

Firmware-Update: Loading 04.03.19.20.dfzTest.1.dfz/evo4.6_04.03.19.20.dfzTest.1.iff
Split into DFZ '04.03.19.20.dfzTest.1.dfz', IFF-File 'evo4.6_04.03.19.20.dfzTest.1.iff'
Delivering 1538326 to device.
```

Appendix E: Firmware Update using HTTP(S)

Datafox Devices implement a firmware update mechanism using HTTP(S) starting from firmware release 04.03.20. The device downloads the “correct” firmware file, checks it and – if supporting the device – installs it.

Determining the “correct” firmware is subject of this appendix.

For your orientation:


Flow HTTP(S) Firmware Update
Device
Network
OEM Server
Firmware Update Server
Start
Some request sent to the server
e.g. data record
Receive data record
Deploy Firmware to device?
Query hardware info from device or bus and address of attached device to be queried
df_kvp=exinfo
df_kvp=exinfo,ac,021,1
(...,ac,<Modul>,<Master>)
Collect Hardware Info and provide it
extended HW-Info-Telegram
Process terminates here, if no firmware is found
OEM decides, which firmware shall be deployed to the device
Check FW Version
Firmware Update µService "Compatibility"
df_load_firmware=<DFZ-File>,<FW-File> [,<routing information>] (if bus device)
Receive download request, trigger download
Load firmware update file from FW-Update Endpoint
(2) OEM delivers firmware as IFF response
(1) OEM fetches Firmware-IFF from DFZ file
Fileserver
Send IFF content
Receive data
Evaluate firmware compatibility
Incompatible
Send System Message
Receive System Message
Compatible
Apply Firmware to device
Keep Setup and Lists
Restart
Create System Message
Send System Message
Receive System Message

Considerations:

- Endpoint for dowloading the actual firmware file is not passed along when triggering the firmware update due to security concerns.
- The OEM triggers the update by its own application and receives the outcome of the process.

Query Firmware Option - Examples:

```
fw=latest
fw=branch,04.03.19    or    fw=branch,04.03.18
fw=version,04.03.19.18
```

Selber Prozess wie auch im Studio. Auswahl der IFF Datei aus dem DFZ

Communication OEM Application - µService Compatibility

Query existing firmware versions:

```
/query?
fw=branch,04.03.19
```

Firmware result information:

```
df_api=1&type=1&dfz=04.03.19.19.dfz

df_api=1&type=2&reason=6001
```

Extended HW Info is relayed to µService:

```
/match?
kv=serialnumber,8675&kv=device,23&
kv=setup EVO 2.8.aes,0x12345678&
kv=cert test.cert,0x087654321&
kv=firmwareversion,04.03.15.05.EVO35&
kv=board,50006,4.7a&
kv=module,29,1.5c,14&
kv=module,37,1.4c,6&
kv=module,102018,1.2b,6.1&
kv=module,11,1.5a,8&
kv=module,30,1.0b,11&
kv=module,110003,1.1c,11.1&
kv=fw,04.03.19.19.dfz,<md5>
```

Service replies with firmware file to be used:

```
SUCCESS:
df_api=1&
reason=6000&
group=600&
type=1&
detail1=...&
detail2=...&
detail3=...&
file=evo2.8_04.03.19.19.iff

FAILURE:
df_api=1&
reason=6001&
group=600&
type=2&
detail1=...&
detail2=...&
detail3=...
```

similar to "incompatibility" system message

"Incompatibility" System Message will contain the module not being compatible with the firmware.

"Installation Success" Message will contain current firmware versions.

Extension Point:

- "Update to latest FW" as interactive function from device's system menu.
- Device may query server for latest firmware version and offer installation to the user.
- Is Datafox required to provide the firmware download service or is "just" the compatibility µSvc required.

Consideration:

- When running on Datafox Web Server, the FW Update Service uses the same firmware source as Datafox Studio for its firmware download function.

Diagram showing the process of the firmware update

The process of the firmware update requires interaction between the Datafox device, the OEM application, the scripts “query.php” and/or “match.php” and a source for firmware files.

The device cyclically sends alive messages. These messages may be used to ask the device for its hardware details (extended info, “extinfo”). The “extinfo” contains information on the device’s hardware components, which are required in order to run the compatibility tests.

“match.php” may be used to determine if a device’s hardware information is compatible with a desired firmware version and to select the correct IFF file to be sent to the device.

Sending the firmware to the device can be accomplished in three ways:

- The OEM application sends the firmware directly as response to the device (see section 2.2.1.2)
- The OEM application tells the device to download the firmware file from a dedicated firmware update server (df_load_firmware, see section 2.2.3.2.27)
- The OEM application tells the device to download an IFF file containing a firmware (df_load_file, see section 2.2.3.2.17)

Both scripts require the parameters to be passed in classical URL encoding along with the GET request. The answer is sent URL encoded as well and resembles the structure of a system message.

Example:

Request:

```
…/query.php?fw=latest
```

Response:

```
df_api=1&type=1&reason=3800&group=380&detail1=04.03.19.20.dfz
```

☞ **Please note:**
The structure of the response is part of the PHP scripts. These scripts are not influenced by the device update.
The structure of a system messages is described in section 2.3.

E.1: Prerequisites for using “query.php” and/or “match.php”

“query.php” as well as “match.php” require a source for device firmware files accessible in the server’ local file system. In order to check MD5 fingerprint efficiently, they have to be precomputed as well.

The firmware has to be extracted into a folder having the same name as the DFZ archive. Additionally, a file named “<dfz-name>.md5” containing the hexadecimal representation of the MD5 hash is expected.

Starting from a root node $ROOT, at least the following files have to be made available when deploying firmware version 04.03.19.19:

```
$ROOT/04.03.19.19.dfz/*.iff
$ROOT/04.03.19.19.dfz.md5
```

Other files from the DFZ archive, e.g. the “*.hex” files, are not used by this firmware method.

E.1.1: Sample script for deploying a firmware version at a server

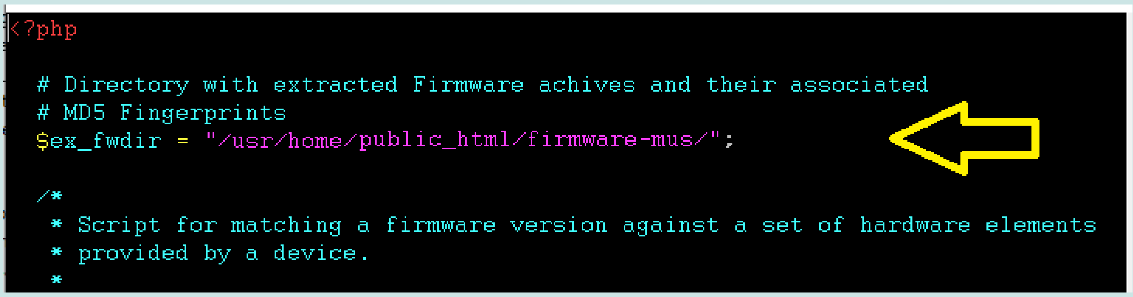
In order to deploy a firmware version, the following script (Linux) may be used. Please note that the script is rather a sample for your convenience that is provided AS-IS without warranty of any kind:

```
#!/bin/bash
```

```
input_fn=$1
base_fn=`basename $input_fn`

rm -rf $base_fn $base_fn.md5
mkdir $base_fn
( cd $base_fn && unzip ../$input_fn > /dev/null )

md5sum $input_fn | cut -b 1-32 > $base_fn.md5

echo IFN $input_fn
echo BFN $base_fn
```

It is expected to be called from the $ROOT directory with the path to DFZ-file. It extracts the DFZ file’s content into a corresponding sub directory and creates a file containing the MD5 hash.

Please note:
The path to the $ROOT directory is part of both scripts. Please insert the location of the firmware source repository on your server into the scripts when deploying them:



```
<?php

  # Directory with extracted Firmware achives and their associated
  # MD5 Fingerprints
  $ex_fwdir = "/usr/home/public_html/firmware-mus/";

  /*
   * Script for matching a firmware version against a set of hardware elements
   * provided by a device.
   *
```

The PHP scripts are available at:

https://datafox.de/download/musvc_firmware_update.zip

E.2: Using “query.php”

The script “query.php” offers information on the firmware versions available on the server. The parameters are expected to passed along with the GET request, the answer is URL encoded in the style of system message.

E.2.1: Determining the latest firmware available

Check for the latest firmware version available on the server.

Request:

```
…/query.php?fw=latest
```

Response:

```
df_api=1&type=1&reason=3800&group=380&detail1=04.03.19.20.dfz
```

E.2.2: Determining the latest firmware from a release branch

Check for the latest firmware version on the server from a certain release branch.

Request:

```
…/query.php?fw=branch,04.03.19
```

Response:

```
df_api=1&type=1&reason=3800&group=380&detail1=04.03.19.20.dfz
```

Request:

```
…/query.php?fw=branch,04.03.16
```

Response:

```
df_api=1&type=1&reason=3800&group=380&detail1=04.03.16.06.dfz
```

E.2.3: Checking if a specific firmware version is available at the server

Check, if a specific firmware version is available.

Request:

```
…/query.php?fw=version,04.03.19.20
```

Response:

```
df_api=1&type=1&reason=3800&group=380&detail1=04.03.19.20.dfz
```

Request:

```
…/query.php?fw=version,04.03.19.21
```

Response:

```
df_api=1&type=2&reason=3802&group=380&detail1=no match
```

E.2.4: Listing all firmware files available on the server

List all firmware versions available on the serve.

Request:

```
…/query.php?fw=list
```

Response:

```
df_api=1&dfz=04.03.19.20.dfz&dfz=04.03.19.19.dfz&…&dfz=04.03.19.01.dfz&dfz=
04.03.18.08.dfz&dfz=04.03.16.06.dfz&…&dfz=04.03.09.20.dfz&dfz=04.02.05.60.d
fz
```

E.3: Using “match.php”

The script “match.php” evaluates, if a specified firmware version is compatible with a certain device. In addition to this, the script identifies the firmware file to be delivered to the device in order to apply the update.

The relevant information for sending a request to “match.php” is provided by the device itself from the “extinfo” command. This command (see 2.2.3.2.14) sends information on the device, its operational state, the main board as well as the built-in modules. In order to complete the query to “match.php” you will have to add the firmware version as well as its MD5 fingerprint (shown on yellow background below)

The checksum is computed on the DFZ-File you are using to provide the update - “match.php” compares you checksum to the version it has access to.

```
.../match.php?
      kv=firmwareversion,04.03.20.03.Evo43&
      kv=board,50007,4.4a&
      kv=module,102026,1.0a,0.11&
      kv=module,104003,1.0a,0.12&
      kv=module,12,1.2a,1&
      kv=module,35,1.3a,2&
      kv=module,1,1.3j,6&
      kv=module,11,1.6b,7&
      kv=module,10,1.1c,8&
      kv=module,106002,1.0a,8.1&
      kv=module,106001,1.0a,8.2&
      kv=module,106001,1.0a,8.3&
      kv=module,106004,1.0a,8.4&
      kv=module,19,1.3a,9&
      kv=module,110004,1.0a,9.1&
      kv=module,110101,1.0c,9.2&
      kv=module,20,1.3a,18&
      kv=device,11&
      kv=serialnumber,1234&
      kv=setup EVO 4.3 F1 Datensatz F2 SysMsg.aes,0AF95295&
      kv=fw,04.03.20.03.dfz,8d3343de9a00cb36e7617e66ace126d8
```

The answer will either be an error message (if it contains …&type=2&…) or the name of the firmware to be delivered from the DFZ container.

An error will be reported similar to

```
      df_api=1&type=2&reason=3911&group=390&detail1=no acceptable compatibility
      info
```

Information on a suitable firmware will be reported as:

```
df_api=1&type=1&reason=3900&group=390&detail1=04.03.19.20.dfz&de-
tail2=evo_intera_II_04.03.19.20.iff
```

E.4: Delivering firmware content

Providing the firmware content is done by the OEM service application, which is the one maintaining the connection to the device. In above diagram the blue arrow (2) shows a shortcut providing the firmware where it is possible to send the firmware with the communication session initiated by the device.

If you are planning to use a standard webserver for offering the firmware files, you may instruct the device by using `df_load_firmware` to send a download request to a preconfigured firmware update server. This server may then be a “normal” standard web server being optimized for content delivery.

Appendix F: Description of API-Level 0

Danger:
“Level 0” protocol only offers basic functions for controlling a device. It is available for Hardware 3 devices in the described way.

If you plan to create an http interface for current (Hardware 4) devices, please implement (parts of) “Level 1” (see 2.2) protocol

In this level generated client records are sent to the Web application. Through the response of Web application actions can be performed.

Plaintext request
getdatagv.php?table=BB&bTYP=Manu&bLOG=Log&bDAT=2011-05-24_08:30:12&bPER=Per&checksum=2120
Plaintext Reply
status=ok&checksum=2120 (checksum of request) (always specify the end: \ r \ n (carriage return line feed))

F.1 Request

Request from the client (device) to the server.

F.1.1 Method: GET

Please note:
If you need a fixed parameter e.g. a client ID, which is sent with every request, then you can set this in the URI of the system variables MOBILE.HTTPSEND.

Example: GET /path/to/script.php?clientid=1234&

Please pay attention to the trailing "&" and that the string of MOBILE.HTTPSEND variable has a length limit of 63 characters.

Parameter name	Meaning
table	Name of record description used to generate the data from.
...	Between *table* and *checksum* are the fields of the record description. The parameter names are corresponding to the field names.
checksum	Checksum for the values of the different data field values. Refer to the section "Parameter checksum".

F.2 Response

Response from the server to the client (device).

Content-Type: application/x-www-form-urlencoded; charset: iso-8859-1

F.2.1 Required parameters details

<table>
<tr><th>Parameter name</th><th>Meaning</th></tr>
<tr><td>status</td><td>Status of process.
<table>
<tr><td>OK</td><td>Signals that the processing was done successfully and the next data set may be sent. In this case the checksum supplied with the server’s response has to be identical to the checksum received.
If status=ok and the correct checksum are sent, the device will send next record afterwards.</td></tr>
<tr><td>error</td><td>The data needs to be retransmitted.</td></tr>
</table>
! **Danger:**
Please make sure that there is no endless loop by constantly sending status=error arises. The client sends the same data record until it is acknowledged with OK status.</td></tr>
<tr><td>checksum</td><td>Checksum for the values of the different data field values. Refer to the section "Parameter checksum".</td></tr>
</table>

F.2.1.1 Parameter "checksum"

The checksum is an additional protection, which is to ensure data consistency of the sent field values between client and server. The data integrity itself is ensured by the checksums used by TCP / IP. It's up to you whether you validate the checksum or simply in response to return the value delivered again.

The checksum calculation is done by adding up the individual character values of the values of GET requests. The Keys will not get into the checksum!

Example: ... &field1=4711&field2=Kommt&... (keys in blue, values in red)
4711 = 52 + 55 + 49 + 49
Kommt = 75 + 111 + 109 + 109 + 116
By summation of the ANSI values of each character results in a checksum of 725.

F.2.2 Optional parameters to include into the response

These parameters are optional, but may have dependencies among themselves. These dependencies are represented by a separate block in the table.

<table>
<tr><th>Parameter name</th><th>Meaning</th></tr>
<tr><td>time</td><td>The date and time that will be set by the device. The data and time supplied will be applied by the device when deviating more than +/- 10 seconds from the device’s clock.
Format: YYYY-MM-dd_hh:mm:ss
Example: time = 2016-11-17_12:13:14</td></tr>
<tr><td>beep</td><td>Beep signal.
The table is a '+' used to represent a long tone and, - for a short tone.
<table>
<tr><td>1</td><td>OK signal</td></tr>
<tr><td>2</td><td>ERROR signal</td></tr>
<tr><td>3</td><td>+</td></tr>
<tr><td>4</td><td>- +</td></tr>
<tr><td>5</td><td>- -</td></tr>
<tr><td>6</td><td>+ +</td></tr>
<tr><td>7</td><td>- - -</td></tr>
<tr><td>8th</td><td>+ + +</td></tr>
<tr><td>9</td><td>- + -</td></tr>
<tr><td>10</td><td>+ - +</td></tr>
<tr><td>11</td><td>SMS signal</td></tr>
</table>
</td></tr>
</table>

F.2.2.1 Service mode

Parameter name	Meaning
service	A value of 1 causes the client to enter the service mode after having transmitted all data records. ☞ **Please note:** Standard HTTP behaviour is that a connection is closed by the Web server. The client will switch to service mode only when the web server closed the connection. To end a connection, you can supply "Connection: close" in the HTTP header of the server’s response. This causes the web server to end the connection.
host	Optional and only to be specified when *service* was provided. If the parameter is omitted, the value of the system variable com.http_mode[n].host is used.
port	Optional and only to be specified when *service* was provided.

	If the parameter is omitted, the value of the system variable com.http_mode[n].port is used.
key	Optional and only to be specified when *service* was provided. This parameter defines, if the service mode connection is - Unencrypted (not key parameter set) - Encrypted with the first active mode server's key (key=key0) - Encrypted with the second active mode server's key (key=key1) ☞ **Please note:** This parameter requires at least firmware version 04.03.14.09.

F.2.2.2 Global variables

Parameter name	**Meaning**
setup.1	Setting the value of a global variable by index 1-8.
setup.id	Setting the value of a global variable by name.

F.2.2.3 Chain of events

Parameter name	**Meaning**
ek	Name of a chain of events, which is to be executed.

F.2.2.4 Message

Parameter name	**Meaning**
message	Text message to be shown on the display. A line break can be reached by specifying "\r" (0x0d). Example: message = This \r is \r pure \ r Text. ! **Danger:** The message is only displayed if the setup option "server online" is activated. This option is found on the "default settings" page.
delay	Specifies the amount of time in seconds for which the message is displayed.
size	Specifies the font size and style. 0 Standard font

<table>
<tr><td></td><td><table>
<tr><td>1</td><td>16 pixel (7 lines)</td></tr>
<tr><td>2</td><td>16 pixel, fixed width (7 lines)</td></tr>
<tr><td>3</td><td>19 pixel (6 lines)</td></tr>
<tr><td>4</td><td>19 pixel, fixed width (6 lines)</td></tr>
<tr><td>5</td><td>21 pixel (5 lines)</td></tr>
<tr><td>6</td><td>21 pixel, fixed width (5 lines)</td></tr>
</table>
The indicated pixel values and lines in the table are approximations and may vary depending on the device used.</td></tr>
</table>

F.2.2.5 Online function of the access control (AC)

<table>
<tr><th>Parameter name</th><th>Meaning</th></tr>
<tr><td>access, modules</td><td>The value of the string – see field "TM" from the "reader" list. The value must therefore consist of exactly 3 digits always.</td></tr>
<tr><td>master</td><td>Id for the RS485 bus used for AC, describes the bus strand.
RS485 bus ID 1
RS485 bus ID 2 etc.
Specify either the properties “master” and “module” or alternatively “access”.</td></tr>
<tr><td>mask</td><td><table>
<tr><td>1/0</td><td>If the bit is set, the buzzer is activated.</td></tr>
<tr><td>2/1</td><td>If the bit is set, the green LED is activated.</td></tr>
<tr><td>4/2</td><td>If the bit is set, the red LED is activated.</td></tr>
<tr><td>8/3</td><td>If the bit is set, the 1st relay is addressed.</td></tr>
<tr><td>16/4</td><td>If the bit is set, the 2nd relay is addressed.</td></tr>
<tr><td>32/5</td><td>If the bit is set, the 3rd relay is addressed.</td></tr>
<tr><td>64/6</td><td>If the bit is set, the 4th relay is addressed.</td></tr>
<tr><td>128/7</td><td>If the bit is set, the 5th relay is addressed.</td></tr>
<tr><td>256/8</td><td>If the bit is set, the 6th relay is addressed.</td></tr>
<tr><td>...,</td><td>Unused. Please set to 0.</td></tr>
</table></td></tr>
<tr><td>type</td><td><table>
<tr><td>0</td><td>Off</td></tr>
<tr><td>1</td><td>On</td></tr>
<tr><td>2</td><td>Toggle (on for 600ms, 600ms off)</td></tr>
<tr><td>3</td><td>3 times turn on for 500ms</td></tr>
</table></td></tr>
<tr><td>duration</td><td>[applies for type=1 only]: The duration, for that the mask is applied.
Interpretation:
- 0 = always on
- 1 - 40 = duration in seconds, for that the mask is active</td></tr>
</table>

Danger:
Please follow the ordering "**access -> mask -> type -> duration**" or “**master -> module -> mask -> type -> duration**” strictly.

F.2.3 Encoding

Please note:
When using HTTPS for device communication, the encoding described in this chapter does not increase communication security.

We recommend **not to use this mechanism when using HTTPS**.

The data fields of the data set can be encrypted using a stream cipher RC4. The (encrypted) field contents are transferred in their hexadecimal representation then.

Parameter name	Meaning
dfcb	The parameter specifies that all these fields up to ‘dfce’ include encrypted field contents. The value of ‘dfcb’ contains the four-digit (1000-9999) public key of the applicable password for the stream cipher.
dfce	The parameter indicates that all of the following fields have no encrypted field contents anymore. If the value is decrypted correctly it must match the value of ‘dfcb’.

F.2.3.1 Illustrate the GET request

In clear text (unencrypted) and encrypted:

Plaintext request
getdatagv.php?table=BB&bTYP=Manu&bLOG=Log&bDAT=2011-05-24_08:30:12&bPER=Per&checksum=2120
Plaintext Reply
status=ok&checksum=2120 (checksum)
Encrypted request
getdatagv.php?dfcb=1000&table=e977&bTYP=14dce883&bLOG=4d7876&…&checksum=c01de865&dfce=019c1bd2
encrypted response
dfcb=1000&status=2b97&checksum=1726950d&…&setup_2=a449fd9c&setup_blue=a9375c8d0672&dfce=b99239f3

F.2.3.2 Detection of encryption

To indicate whether the data fields are sent encryptedly, the start of the encrypted data is marked using ‚dfcb' (Datafox crypt begin), the end with ‘dfce’ (Datafox crypt end). ‘dfcb’ is the first field and ‘dfce’ the last field in the request.

The value of the field ‘dfcb’ itself is transmitted in plain text and is the public key. It is a random number between 1000 and 9999. The value must be used in conjunction with the “communication password” for the encryption and decryption.

The encryption of data is thus effected by "private key + public key" as a password key.

In the response, the field ‘dfcb’ must be returned identically. This ensures that the decryption was successful and that request and response also fit.

The value of the field ‘dfce’ is the same as ‘dfcb’ – however it has to be encrypted. While decoding can thus be ensured, if the key used is correct. The value of ‘dfce’ therefore must be identical to ‘dfcb’ after decryption.

If there are problems during deciphering you must include ‘dfc=error’ in the response. In addition, the fields ‘dfcb’ and ‘dfce’ have to be populate with information as follows:

The following *errors* are to be observed by the evaluating script:

‘dfcb’ is not a number or is outside of its value limit of 1000 - 9999

- Answer: dfc=error&dfcb=range&dfce=unknown/missing
  - Range describes a range error – the value is outside its limits.
  - Unknown describes an unchecked condition.
  - Missing identifies a missing field in the request.

‘dfcb’ without final ‘dfce’

- Answer: dfc=error&dfcb=1000&dfce=missing

‘dfce’ is not a number or is outside of its value limit of 1000 - 9999

- Answer: dfc=error&dfcb=1000&dfce=range

‘dfce’ without incipient ‘dfcb’

- Answer: dfc=error&dfcb=missing&dfce=unknown

‘dfce’ is not equal ‘dfcb’

- Answer: dfc=error&dfcb=1000&dfce=different
  - ‘dfce’ (after decoding) is not equal to ‘dfcb’.

F.2.3.3 Response of the web server

The field contents from the request are sequentially decrypted using the RC4 stream cipher. The field contents of the reply is to be seen as part of the entire data stream and thus encrypted using the same RC4 stream cipher instance used for the decryption. The only exception is the first field value ‘dfcb’, that is identical to the one supplied by the request.
The last encrypted field of the reply has to be the ‘dfce’ field. The value of ‘dfce’ must be (after decryption) equal to the value of ‘dfcb’.

☞ **Please note:**
Please ensure that the parameter values are processed in the order they are passed in request and response. The RC4 cipher realizes an internal state preventing processing data out of order.

Appendix G: Distribution Update through the Access bus (Routing)

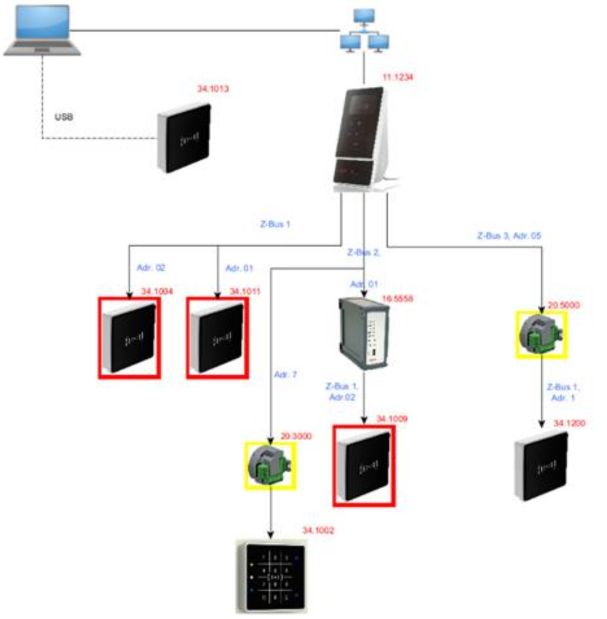
The following section is taken from the software booklet accompanying firmware version 04.03.21. It describes how updates are distributed in the access bus - both with Datafox Studio, the communication library DFCom and via HTTP(S).

G.1 Distribution of Update using the access control bus

If you want to update a device that is connected to the access bus, it is necessary to transmit the new firmware to the access controller that can be reached via the main communication (e.g. LAN). This controller then takes over the transfer to the target devices connected to the access bus.

In the past, we provided so-called "UPD" files for this purpose. We have revised this concept and now support the devices "in the bus" with the usual DFZ firmware file. The information about which devices are to be updated is added as routing information during the firmware update.

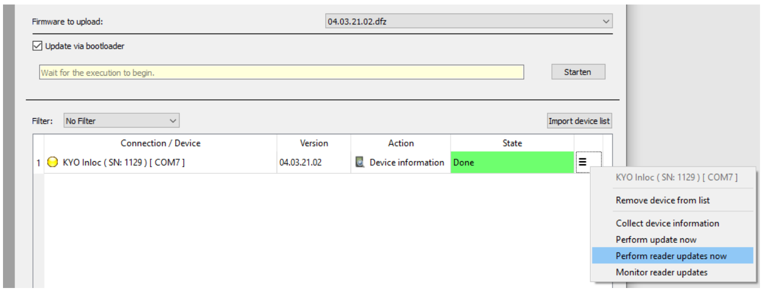
The implementation in the DFCom library uses the new function DFCWriteFile() (see df_files.h) for this purpose. In the HTTP context, the routing data is transmitted as HTTP header "df-routing:".



G.2 Update of access control devices using the Datafox Studio

The update manager in Datafox Studio has been extended with the current version 04.03.21.06 so that firmware can also be distributed via the access bus.

Please first select the firmware version to be transferred for the bus device that is to receive the update in the upper area of the update manager (here 04.03.21.02.dfz). Then select the action "Perform reader update now" from the context menu of the higher-level access controller in the update manager.



Firmware to upload: 04.03.21.02.dfz

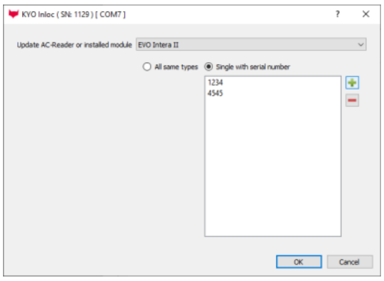
☑ Update via bootloader

Wait for the execution to begin.

Filter: No Filter

	Connection / Device	Version	Action	State
1	KYO Inloc (SN: 1129) [COM7]	04.03.21.02	Device information	Done

In the dialogue that now appears, select the type of device(s) to be updated. You can roll out the update to all accessible devices of the type or limit it to individual devices.



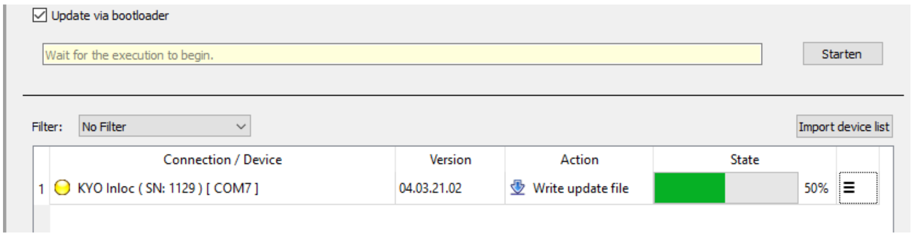
KYO Inloc (SN: 1129) [COM7]

Update AC-Reader or installed module: EVO Intera II

○ All same types ● Single with serial number

1234
4545

Pressing the "Ok" button starts the update process. The progress is monitored by Datafox Studio and displayed in the Update Manager:



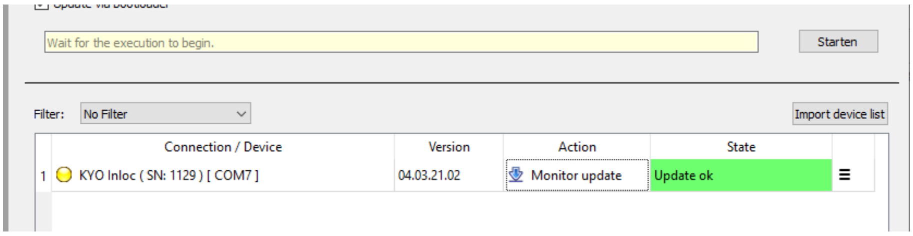
☑ Update via bootloader

Wait for the execution to begin.

Filter: No Filter

	Connection / Device	Version	Action	State
1	KYO Inloc (SN: 1129) [COM7]	04.03.21.02	Write update file	50%

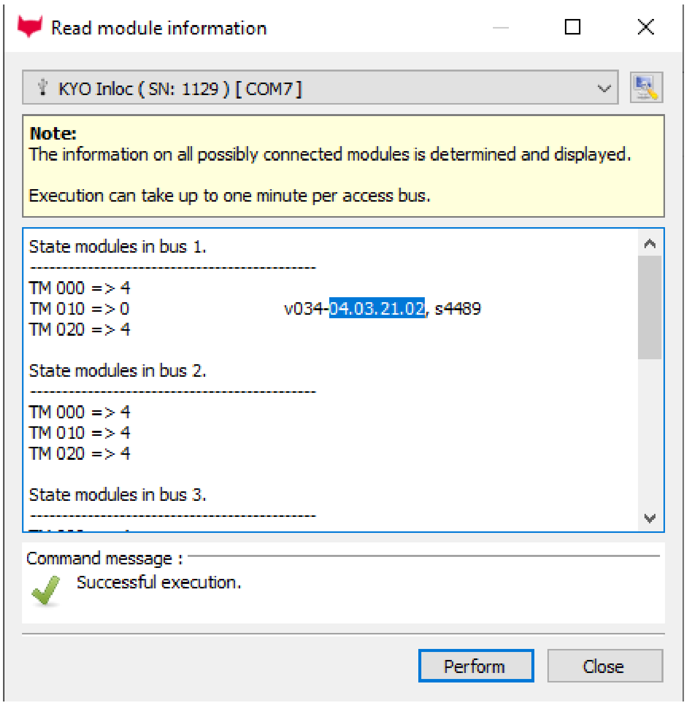
After successful transfer and execution of the update, a corresponding message is displayed in the action column.



Wait for the execution to begin.

	Connection / Device	Version	Action	State
1	KYO Inloc (SN: 1129) [COM7]	04.03.21.02	Monitor update	Update ok

You can check the firmware versions used in the system via the "Extras -> Read status messages from the access control" dialogue.



Note:
The information on all possibly connected modules is determined and displayed.

Execution can take up to one minute per access bus.

```
State modules in bus 1.
-----------------------------------------------
TM 000 => 4
TM 010 => 0                     v034-04.03.21.02, s4489
TM 020 => 4

State modules in bus 2.
-----------------------------------------------
TM 000 => 4
TM 010 => 4
TM 020 => 4

State modules in bus 3.
-----------------------------------------------
```

Command message :
Successful execution.

G.3 Creating routing information when implementing firmware update yourself

Unless you use Datafox Studio to update devices in the access bus, this chapter provides information on implementation details of routing, i.e. the accessibility of the individual devices within the system.

G.3.1 Logical structure of routing information

Routing information consists of one or more paths through a Datafox system. This essentially requires specifying whether a node is to process or forward the data and how the node is to be reached (from the predecessor node).

An example is shown on the right. The KYO Oneloc (SN 3000) highlighted there can be reached from the PC (notebook) via the EVO 4.3 (SN 1234) connected to the network in access bus 2 at address 7.

Three EVO Intera II readers are also highlighted, the two readers with serial numbers 1004 and 1011 are connected to access bus 1 of the EVO 4.3, the EVO Intera II reader (SN 1009) is connected as a detached reader to the KYO Inloc (SN 5558).

Routing is done on a logical level, i.e. no knowledge of bus addresses and buses is required. Only knowledge of the devices involved on the way to the target device and their device types is required (cf. **G.5 Assignment of device types**). The devices are on the same logical level on a bus. If the reader is detached or not does not affect the routing rule.

From the PC, the KYO Oneloc is connected to the EVO 4.3. The rule requires device types and serial numbers to describe the path through the system. Thus, the KYO Oneloc can be reached by the routing rule

PC -> 11.1234 -> 20.3000

> ☞ **Please note:**
> If you want to update a directly accessible device, you do not need any routing information. Simply transfer the update directly to this device as usual, without routing information.

G.3.2 Options

To address a specific subgroup of devices, such as the EVO Intera II devices highlighted in red in the picture above, the specific devices are addresses via device type and serial number.

PC -> 11.1234 -> 34.1004
PC -> 11.1234 -> 34.1011
PC -> 11.1234 -> 34.1009

If you do not want to limit a routing rule to a specific serial number, but want to use all devices of the type, you can express this with an asterisk (*) instead of the serial number. In order to address all KYO Oneloc highlighted in yellow, the following routing rule can be used.

Related to the above example, the following rules are equivalent:

PC -> 11.1234 -> 20.*

G.1.2.3 Concrete, complete routing rules

A routing information (RI) can contain several routing rules (RR), these are separated by semicolons (;). A routing rule corresponds to a path through the system to a target device.

A path begins with an identifier about the possible feedback (N for no feedback expected, A for response expected), followed by the list of devices that specify the path through the system. Optionally, the definition of the receiver can still be made, which is appended to the last device in square brackets.

Underlying grammar:

```
RI  ->  ( <RR> <APPL>? <Semicolon> )+
RR ->  <AN> ( <SPC> <Level> <Comma> <Addr> <Comma> <Mode> )+
AN ->  A | N
Level   ->  0 | 1
Addr    ->  <Number „DevTypeId“> <Dot> <Number „Serial Number“> | *
APPL    ->  [ <Addr> ]
Mode    ->  S | E
Number->  0 | [1-9][0-9]*
Semicolon ->  ;
SPC     ->  _ (Blank)
Comma -> ,
Dot->  .
```

Level describes the degree of support of routing rules by the component. Datafox devices (with firmware from 04.03.21) have a "1" here, any other devices are described here with "0". The value is designed for possible future extensions to the routing.

The address (**Addr**) designates a Datafox device and is specified as device type <dot> serial number. The serial number can also be specified as "*" - in this case, the rule applies to all devices match the specified device type.

Mode indicates what is to happen to the transmitted data. Here, "S" for self-application and "E" for forwarding to a following node are possible.

If a controller applies a firmware update itself, this is associated with a restart of the device. Forwarding within the same rule is therefore not possible.

The update rules described above thus become complete as

`N 1,11.1234,E 1,34.1004,S;N 1,11.1234,E 1,34.1011,S;N 1,11.1234,E 1,15.5558,E 1,34.1009,S;` (1)

respectively

`N 1,11.1234,S[34.*];N 1,11.1234,E 1,15.5558,E 1,34.1009,S;` (3)

or

`N 1,11.1234,S[34.*];N 1,11.1234,E 1,15.5558,E 1,34.1009,S;` (3)

specified.

> ☞ **Please note:**
> Addressing with "[]" in rule (3) leads to the evaluation of whether the firmware update is compatible being carried out by the previous device. If devices with different CPUs are connected in the access bus, this means that the update is not transmitted to a device with a CPU that is not compatible with the update - you save transmission bandwidth and time in the access bus.
>
> If modelling according to (1) or (2) is selected, the firmware is first transferred to the target device and only there checked for compatibility.

G.4 Determining access control bus participants in a live system

This section describes how to determine the devices attached to an access control bus programmatically.

G.4.1 Structure of system variable „access.readerinfo“

The system variable "access.readerinfo" contains information about participants on the access buses of an access controller. This variable is provided by the controller and can therefore only be read, not written.
The data is coded in INI format:

```
[global]
busses=4
bus_active=0,1,2

[bus_0]
readers=010,020

[bus_1]
readers=010,011,020

[bus_2]
readers=010,020

[reader_0_010]
state=0
readerId=110
[reader_0_020]
state=0
readerId=120
cpu=49004
type=34
serial=3102
vendor="datafox"
version="04.03.21.03"

[reader_1_010]
state=0
readerId=210
type=20
serial=1108
vendor="datafox"
version="04.03.21.03"
[reader_1_020]
state=0
readerId=220
type=24
serial=1115
vendor="datafox"
version="04.03.15.18"

[reader_2_010]
state=0
readerId=310
type=24
serial=321
vendor="datafox"
version="04.03.15.18"
```

```
vendor="phg"
version="3C02"

[reader_1_011]
state=0
readerId=211
cpu=49004
type=34
serial=0
vendor="datafox"
version="04.03.21.02"

[reader_2_020]
state=0
readerId=320
cpu=49004
type=34
serial=1013
vendor="datafox"
version="04.03.21.03"
```

The content of the variables is divided into three different section types:

G.4.2 Section [global]

In the [global] section, the key "busses" is stored; it specifies the number of possible buses as a numerical value and is required for finding the readers.
In addition, the key “bus_active” is provided, in which the bus numbers - separated by commas and starting with 0 - are enumerated.

G.4 3 Section [bus_<idx>]

The section [bus...] describes a bus. Here <idx> is replaced by the bus number, starting with 0. The [bus...] section contains the key "readers". This indicates which readers are present in this bus. A 3-digit number is always used (format like TM). If there are several readers in the bus, the values are separated by a comma.

G.4.4 Section [reader_<busId>_<readerId>]

In the [reader...] section, the individual values for a reader are now stored. In the section, <busId> is replaced by the number of the bus (o ... n) and <readerId> by the 3-digit numerical value of the reader.

The following keys are stored in the section:

Key	Value	Example	Description
state	Number	7	Current status of the access control module 0 = module detected and working 3 = module not present in list, nonetheless found in the bus. 4 = module present in list, not found in bus. 5 = module uses different communication key. 6 = module requires a login password. 7 = RFID reader type different from setup (Mifare, Legic, Unique, etc.) 8 = Error while configuring the RFID reader.
readerId	Number	1	Id of reader from the reader list
vendor	Text	„datafox“	Manufacturer code of the module
cpu	Number	49004	ID of the CPU used by the module. If the CPU is not detectable, this value is omitted.
type	Number	34	Numeric device type
serial	Number	1001	Device serial number
version	Text	„04.03.20.11“	Current firmware version

G.4.5 Locating the [reader] sections

To find the [reader...] sections, the number of buses is first read from the global section.

Then the bus sections are searched for all possible bus numbers, starting with 0. If no bus section for a specific number can be found, no reader is available in this bus.

The connected readers can now be read out in the sections of the buses. Together with the bus index and the participant number in the bus, the section of the reader can now be searched for.

☞ **Please note:**
An update by means of an IFF file can only be carried out if the reader is reported with status 0.
Furthermore, the device must be of the Datafox type and the key cpu must be present.

G.5 Assignment of device types

The table lists the devices currently available through Datafox and the respective device type identifier.

Device Type	**ID of Device Type**
AE-MasterIV	5
Dockingstation V2	18
EVO 2.8 Pure	10
EVO 3.5 Pure	23
EVO 3.5 Universal	21
EVO 4.3	11
EVO 4.6 Flexkey	35
EVO 5.0 Pure	36
EVO Agera	24
EVO Intera II	34
Fahrzeugdatenlogger V2	19
IPC Embedded-System/Q7	14
IO-Box V4	15
KYO Cenloc	25
KYO Fourloc	37
KYO Inloc	16
KYO Oneloc	20
Mobil-Box V4	17
PZE-MasterIV	0

Appendix T: Troubleshooting

This chapter collects aspects that have been encountered by Datafox during different deploy scenarios. Even if these scenarios don't exactly fit a problem, you are currently observing, please consider through the chapter as ideas for your investigation.

T.1: Problems with specific web servers

We observed that not all web servers do implement the requirements of RFC 2616 correctly. We use the system variable `http.flags` to adopt the client's behaviour to these idiosyncrasies.

T.1.1: Port inside the Host-Header of the http request

RFC 2616 (http/1.1) defines the host header inside the http header as

```
   Host = "Host" ":" host [ ":" port ]
```

However, we observed web servers that will reject a request having the port contained inside the host header (some servers even crashed). On the other hand there are reverse proxies that require the port inside the host header to determine the correct internal host to establish the connection to.

Should your webserver require to not get the port inside the host header. Please set bit 0 of the device's `http.flags`.

!

Attention:
A server not capable of handling requests containing Host header and ports may occur at different stages, especially with cloud solutions currently.

We observed at least one cloud solution that allowed https communication and replied an HTTP 502 Bad Gateway due to a port contained in the Host header. Since the SSL communication was correctly established at this point of time, the response must have come from a server dispatching the request to the correct host, due to the port contained inside the HOST header.

T.1.2: Proxy server/load balancer and service mode (Connection: Close)

If the device is required to establish a service mode connection (see `service` (F.2.2.1 **Service mode**) or `df_service` (2.2.3.2.3)), it requires the server to close the pending http connection. This may be done explicitly by the server or due to a timeout. An intentional close of the connection is indicated by the server through the header files "`Connection: close`".

Should there now be a proxy server (or a load balancer) between client and server, this proxy will intercept the "`Connection: close`" and close the connection to the server – however the connection to the client is kept. Thus, the device cannot establish a service mode connection.

You may have the device close its connection itself when receiving a service mode request by setting bit 1 in the `http.flags`.

T.1.3: Requests with absoluteURI or abs_path adressing

RFC 2616 (http/1.1) defines in section 5.1.2 that the http request URI may be one of the following:

```
   Request-URI    = "*" | absoluteURI | abs_path | authority
```

The first line of a http request with absoluteURI-addressing looks as following:

```
GET http://www.w3.org/pub/WWW/TheProject.html HTTP/1.1
```

An abs_path-encoded request's first line will not contain protocol, host and port – they will follow in a host-header:

```
GET /pub/WWW/TheProject.html HTTP/1.1
Host: www.w3.org
```

Datafox devices with firmware newer than 04.03.18.03 create abs_path requests, provided they are not interfacing with a proxy server (com.http_mode[.}.proxy) or when this is explicitly required by setting bit 5 of the http.flags.

T.1.4: Example: Applying the correct http.flags

As mentioned before, http.flags are a bitfield – which is represented by a decimal number in the device. Should your installation require abs_path addressing and have a problem with the port inside the host header, you will have to set bits 0 and 2 simultaneously.

For this set http.flags to 5 = $2^0 + 2^2$.

T.2: HTTPS connetion to an AWS / CloudFront service

Amazon CloudFront requires the SNI hostname to be set correctly, otherwise the HTTPS handshake will terminate with alert 40. The SNI hostname is identical to the name of your AWS instance.

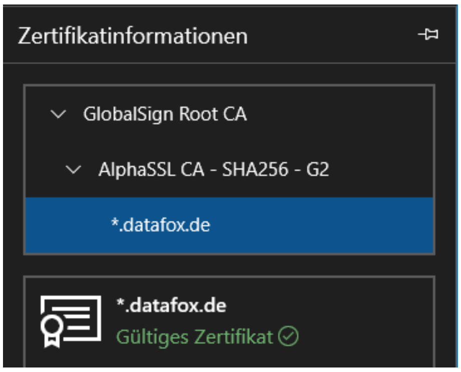
Please enter the AWS instances Hostname into the com.http_mode[.].tls.sni_host variable inside the device communication settings.

T.3: My device reports SSL-Write -9984 error – although I installed the server's certificate correctly at the device.

According to the TLS specification the server shall transfer the entire chain of certificates excluding the root certificate. The client can then test the chain against the trustworthy certificates stored directly on it.

We have observed servers that do not transmit the entire chain. In this case you will have to deploy intermediate certificates **as well** to the client.

Consider the certificate chain of "datafox.de" website. The root certificate originates as "GlobalSign Root CA", "AlphaSSL CA – SHA256 – C2" is derived from it, which then is used to create "*.datafox.de".



Zertifikatinformationen

- GlobalSign Root CA
  - AlphaSSL CA - SHA256 - G2
    - *.datafox.de

*.datafox.de
Gültiges Zertifikat ✓

You may have to export, convert and deploy the entire chain according the description in Appendix B.

T.4: Virtual Hosts and HTTPS (e.g., Microsoft IIS)

Using the HTTP protocol resolving of virtual hosts (to enable running more than one logical server on a physical server hardware) is typically done using the HOST header field from the HTTP request.

When establishing a connection using HTTPS the request's header fields are not available during the SSL handshake – and additional problem is that different virtual server instances may have different certificates.

If you are using an installation with virtual hosts, please set the SNI hostname along with the networking configuration. According to our knowledge at least the Microsoft IIS then will be able to provide the correct certificate during the SSL handshake.

T.5: Reference to http.flags

Using http flags, the following adjustments to data processing may be done:

Bit / Value	Name	Description
1 / 1	Host-Header without Port	If set, the host header is sent without the port information (see E.1.1)
2 / 2	Close for service	If set, the device actively closes a connection if a request for service has been received by the device (see E.1.2)
~~3 / 4~~	~~Using abs_path requests~~	~~If set, the device creates abs_path instread of absoluteURI requests (see E.1.3)~~ ! **Attention:** This Flag has been obsoleted as of version 04.03.18.03. The decision abs_path / absoluteURI is based on the definition of a proxy server: - If a proxy server is present, absoluteURI is used.

		- If no proxy server is configured, abs_path is used.
4 / 8	Exclude static headers	If set, the device will only create `Host:`- and `Content-Length:`-headers. Every other header will have to originate from “header.extensions” (see 1.7) or will not be sent.
5 / 16	Force absoluteURI	(04.03.18.03 an up): If this flag is set, absoluteURI is used independently of a proxy server being present always.
6 / 32	TCP Proxy	(04.03.21.21 / 04.03.22.03 and newer): Proxy mode 0: The device establishes an http or https connection to the proxy server and informs this on the connection target through the http(s) request. 1: The Proxy server is addressed using **http** protocol (ignoring the “http.mode” setting). The device sends a CONNECT-Request to establish an explicit connection to the target host. Communication to the target host is using **http or https**, depending on the “com.http” device setting.

T.6: Basic Authentication not working

We have observed cases of non-working basic authentication together with cellular radio in combination with M2M SIM cards. The root cause of failure was that the provides disconnects each TCP/IP connection directly after delivering the data package.

In this scenario it is not possible to determine the BA realm: The device associates the realm to the connection for its duration – for security reasons. When receiving the realm along with the “http-401 – unauthorized” message, the realm is being discarded when the connection is terminated by the provider – with the next connection attempt another http-401 is generated by the server, since the device does not know the authentication realm anymore and cannot provide the required credentials.

T.7: Different certificates/certificate chains on the same web server

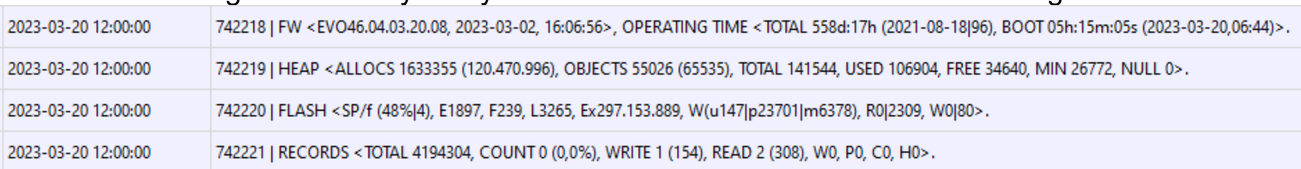
Currently, we support RSA certificates with up to 2048 bit key length with the device firmware. This restriction applies to all certificates in the certificate chain.

The usability of certificate chains with longer keys depends largely on the communication type and device equipment of the device - therefore this is not generally guaranteed.

Virtual hosts offer a possibility to specifically use different certificates on a web server - and thus also certificates of different lengths. You can specify the use of a special virtual host via the SNI_HOST if it is not present in the DNS, and thus create a dedicated endpoint on your web server that uses a certificate chain with 2048 bits.

T.8: Free Memory during TLS communication

Datafox device log free memory every 12 hours – this statistic is created at midnight and at noon.



```
2023-03-20 12:00:00    742218 | FW <EVO46.04.03.20.08, 2023-03-02, 16:06:56>, OPERATING TIME <TOTAL 558d:17h (2021-08-18|96), BOOT 05h:15m:05s (2023-03-20,06:44)>.
2023-03-20 12:00:00    742219 | HEAP <ALLOCS 1633355 (120.470.996), OBJECTS 55026 (65535), TOTAL 141544, USED 106904, FREE 34640, MIN 26772, NULL 0>.
2023-03-20 12:00:00    742220 | FLASH <SP/f (48%|4), E1897, F239, L3265, Ex297.153.889, W(u147|p23701|m6378), R0|2309, W0|80>.
2023-03-20 12:00:00    742221 | RECORDS <TOTAL 4194304, COUNT 0 (0,0%), WRITE 1 (154), READ 2 (308), W0, P0, C0, H0>.
```

The line containing „HEAP“ shows, that

- An amount of 141544 bytes is available (TOTAL)
- From this amount, 106904 bytes are in use currently (USED)
- As consequence 34640 byte are currently available (FREE)
- During the device operation at least 26772 bytes have been available (MIN) and
- The memory allocation failed 0 times (NULL).

Please ensure that at least 16 kBytes of memory are available as MIN during HTTPS communication.

☞ **Please note:**
Please keep in mind that the memory allocation takes place during HTTPS communication. As soon as you connect a USB cable the device changes its communication method – the FREE and USED values do not match the HTTPS communication scenario.

T.9: Runtime considerations on HTTPS communication

The TLS handshake required for HTTPS communication to securely negotiate the communication key is costly - both in terms of computing time and in terms of the data exchanged between server and device.
It is quite possible that 4-5 seconds are required for the handshake and about 1.5 kB of data must be exchanged in this context. The actual data record then transferred normally requires less than 1/3 of this amount of data.

T.9.1: Closing the connection from the server side

Typically, it is not necessary to negotiate a new connection for each exchanged data record. However, this becomes necessary if the device or the server actively closes the communication connection after each data exchange, for example by adding "Connection: Close" in the HTTP header or by actively closing the TCP network socket.
We therefore recommend explicitly closing the connection only when establishing a maintenance connection (service mode, cf. 2.2.3.2.3).

T.9.2: Cloud services

Even with "cloud services", where connections are even scarcer than with "on premise" services, the unconditional closing of an HTTPS connection should be reconsidered. If a device has more than one record to transmit, it will immediately initiate the next connection - and thus occupy another connection. This results in more CPU load on both sides with ultimately equal use of connections.
Allowing the device to reuse the existing connection by means of a short timeout (3-5 seconds) will result in both: connections being released quickly and less server load.

T.9.3: Support from the device

Since firmware release 04.03.21.12, the device transmits along with the HTTP header how many data records are still available for transmission. Based on this information (and the knowledge of which messages you still expect from the device due to current actions), you can decide whether it makes sense for the server to close the connection.