Protocol API

Web interface

Packet interface

V1.5.0

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC181004API03EN | Revision 3 | English | 2022-10 | Released | Public

Table of contents

1 Introduction

1.1 About this document

This manual describes the API of the integrated web server. The name of this interface is WebIf (“Web interface”).

1.2 List of revisions

Rev	Date	Name	Chapter	Revision
1	2019-03-28	ATI, HHE	All	Document created.
2	2022-02-22	AIV, RGO	3	Documented the use of ulSrcId in IND packets and ulDestId in REQ packets for unique HTTP request identification.
			3.1	Added descriptions for new commands: Set Dispatch Entry URL, Set Dispatch Entry Enabled, Set TCP Port, Start Web Server, Stop Web Server
			3.2.1	Updated the description of the tClientAddr and tAuth fields.
			4	Added a new chapter with examples.
			5	Updated error codes.

Table 1: List of revisions

1.3 System requirements

The software package has the following system requirements to its environment:

- netX chip as CPU hardware platform
- operating system for task scheduling required

1.4 Intended audience

This manual is suitable for software developers with the following background:

- Knowledge of the programming language C
- Knowledge of the cifXTools
- Knowledge of the HTTP protocol

1.5 Technical data

The data below applies to the web server V1.5.0 featured with the WebIf component.

Stack available for netX

netX	Available
netX 10	No
netX 50	No
netX 51	No
netX 90	Yes
netX 100, netX 500	No

Table 2: Technical data – Available for netX

1.6 Terms, abbreviations and definitions

Term	Description
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP over SSL
WebIf	Web Interface (component)

Table 3: Terms, abbreviations and definitions

1.7 References to documents

This document refers to the following documents:

[1] IETF: Hypertext Transfer Protocol -- HTTP/1.1 (https://tools.ietf.org/html/rfc2616).

[2] Hilscher Gesellschaft für Systemautomation mbH: API, Web server API netX 90/4000, Revision 2, Eglisch, 2021.

Table 4: References to documents

2 Functions of the web server

The web server is a component and (can be) integrated in a firmware.

The web server communicates using the HTTP protocol. Requests of a client, e.g. from a web browser, are forwarded to the application. The application using this interface to the WebIf component will be able to receive HTTP requests and has to generate the corresponding HTTP responses.

In general, the integrated web server forwards HTTP requests to the application, though answers specific HTTP requests (e.g. URL to /netX) directly.

3 The Application Interface

3.1 Overview of packets

The following table provides an overview on the packets available for your application:

Overview over packets of the WebIf component			
No. of section	**Packet**	**Command code (REQ/CNF or IND/RSP)**	**Page**
Management packets			
3.3.1.1	WEBIF_SET_DISPATCH_ENTRY_URL_REQ/CNF – Set dispatch entry URL request/confirmation	0xAF2A/ 0xAF2B	10
3.3.1.2	WEBIF_SET_DISPATCH_ENTRY_ENABLED_REQ/CNF – Set dispatch entry enabled request/confirmation	0xAF28/ 0xAF29	11
3.3.2	WEBIF_SET_TCP_PORTS_REQ/CNF – Set TCP ports request/confirmation	0xAF26/ 0xAF27	12
3.3.3	WEBIF_START_REQ/CNF – Start the web server request/confirmation	0xAF22/ 0xAF23	13
3.3.4	WEBIF_STOP_REQ/CNF – Stop the Web Server request/confirmation	0xAF24/ 0xAF25	13
3.3.5	WEBIF_ENABLE_REQUEST_HANDLING_REQ – Enable HTTP request handling request/confirmation	0xAF10/ 0xAF11	14
3.3.6	WEBIF_DISABLE_REQUEST_HANDLING_REQ – Disable HTTP request handling request/confirmation	0xAF20/ 0xAF21	14
Packets for HTTP Request handling			
3.4.1	WEBIF_HANDLE_HTTP_REQUEST_IND/RSP – HTTP Request indication/response	0xAF00/ 0xAF01	15
3.4.2	WEBIF_HANDLE_HTTP_REQUEST_CONTENT_IND/RSP – HTTP Request Content indication/response	0xAF02/ 0xAF03	17
3.4.3	WEBIF_FINISH_HANDLING_HTTP_REQUEST_IND – End of HTTP Request indication/response	0xAF04/ 0xAF05	18
3.4.5	WEBIF_GET_HTTP_REQUEST_FIELD_REQ – Get HTTP Request Fields request/ confirmation	0xAF06/ 0xAF07	19
Packets for HTTP Response generation			
3.5.1	WEBIF_GENERATE_HTTP_RESPONSE_REQ – Generate HTTP Response request/confirmation	0xAF08/ 0xAF09	21
3.5.2	WEBIF_GENERATE_HTTP_RESPONSE_FIELD_REQ – Set HTTP Response Fields request/confirmation	0xAF0A/ 0xAF0B	22
3.5.3	WEBIF_GENERATE_HTTP_RESPONSE_CONTENT_REQ - HTTP Response Content request/confirmation	0xAF0C/ 0xAF0D	23
3.5.4	WEBIF_FINISH_GENERATION_HTTP_RESPONSE_REQ – End of HTTP Response request/confirmation	0xAF0E/ 0xAF0F	24

Table 5: Overview over packets of the WebIf component

3.2 HTTP processing in the WebIf component

In order to receive HTTP requests, the application must first enable HTTP request handling within the WebIf component. Afterwards, this handling can be disabled or re-enabled.

The HTTP processing is separated into two phases that each have to be performed sequentially.

The application has to perform the following steps for each phase (the numbering of steps relates to *Figure 1: Packet sequence* on page 8):

1. HTTP request handling: reception of the **HTTP request**.
   The WebIf component uses indication packets to inform the application and communicate to it. The application has to use response packets to communicate to the WebIf component.
   - Wait for a new HTTP request (steps 1 and 2)
   - Wait for HTTP request body until HTTP request end (steps 3, 4, 5 and 6)
   - Get HTTP request fields (steps 7 and 8)
2. HTTP response generation: generation of the **HTTP response**.
   The application has to use request packets to communicate to the WebIf component. The WebIf component replies confirmation packets to the application.
   - Begin a new HTTP response (steps 9 and 10)
   - Set HTTP response fields (steps 11 and 12)
   - Set HTTP response body (steps 13 and 14)
   - End of the HTTP response (steps 15 and 16)

In case the application does not transmit the response packet or the request packet corresponding to the current phase and current step within the timeout interval, the stack will generate a `500 Server Error` HTTP response directed to the client. After timeout expiration and the transmission of the `500 Server Error` HTTP response, the stack again indicates a new HTTP request reception with the corresponding indication.

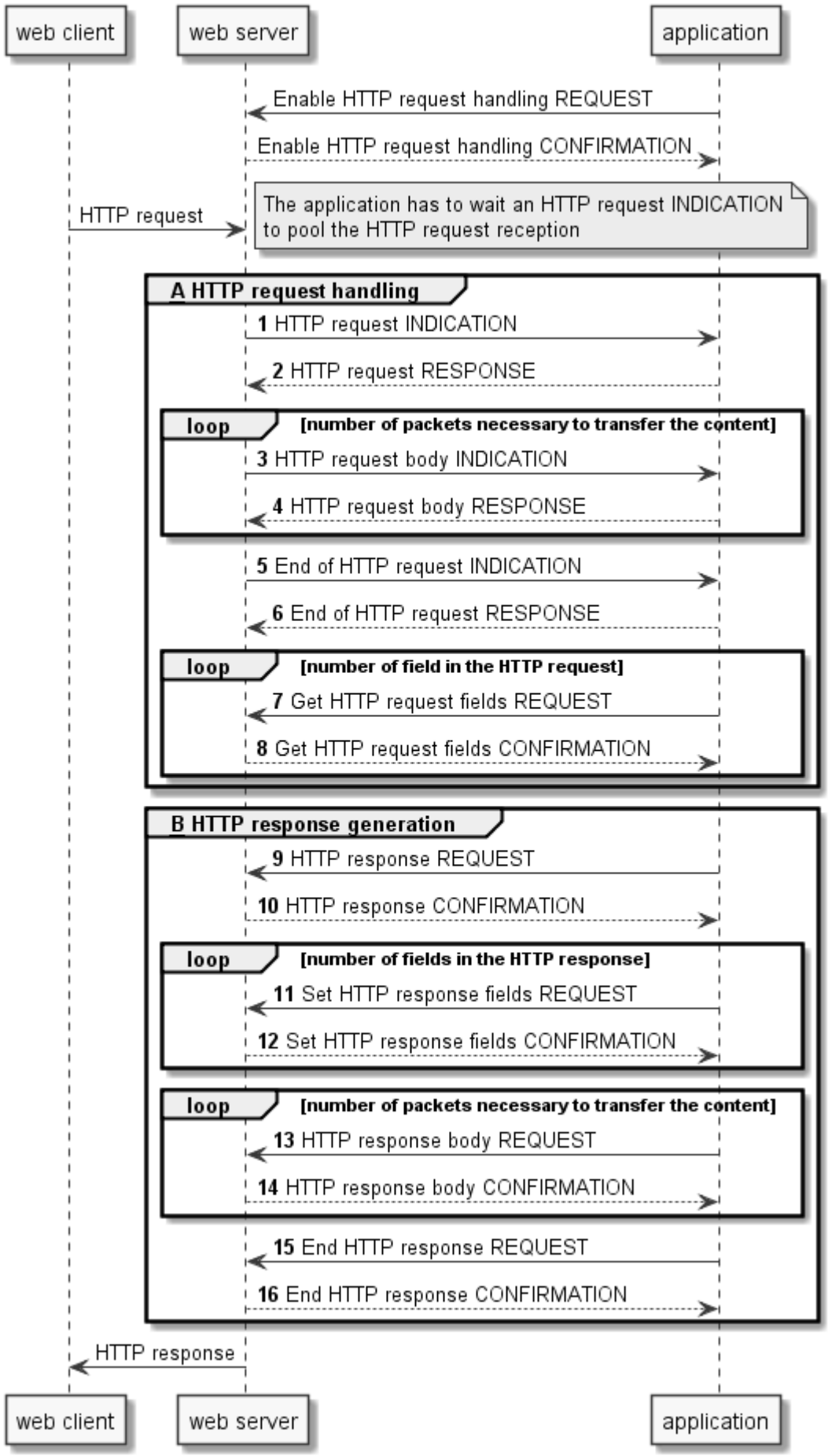


Figure 1: Packet sequence

3.3 Management

3.3.1 Dispatch Entry Customization

The web server manages modules within a dispatch table. Each web server module is represented there in an individual dispatch entry. At system startup, a default dispatch table containing all configured web server modules is predefined. A certain degree of dispatch table customization is allowed at runtime.

For each dispatch entry, a unique identifier is defined, so that a specific web-server module can later be customized using this dispatch entry ID. These entries correspond to the features described in [2], Table 2: Feature overview.

The following table describes the dispatch entries available for customization and the URLs that they are by default mapped to:

Feature	Dispatch ID (decimal)	Default URL
Reset	5	/netx/reset
Diagnostic	6	/netx/diag
WebIf	3	/webif
File server	8	/files
File manager	9	/netx/filemanager
Firmware upload	4	/netx/firmware
netProxy object access	17	/netx/npx
Authentication	12	/netx/login
User Manager	11	/netx/usermanager
GUI	7	/netx

Table 6: Dispatch ID per feature and corresponding default URLs.

The IDs above are defined in the `WebServer_Configurations_DispatchIds.h` header file.

3.3.1.1 Set Dispatch Entry URL

This service allows to customize the URL, at which a webserver module is configured to respond.

Note: This service can only be processed successfully, if the web-server is not running, e.g. before sending the WEBIF_START_REQ command.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	1 … 129	Packet data length in bytes
ulCmd	uint32_t	0xAF2A	WEBIF_SET_DISPATCH_ENTRY_URL_REQ
Data			
bDispatchId	uint8_t		ID of the specific dispatch entry
szUrl	CHAR[]	0 …128 characters	New URL string. The actual string size is determined by the packet length. It does not need to be zero-terminated.

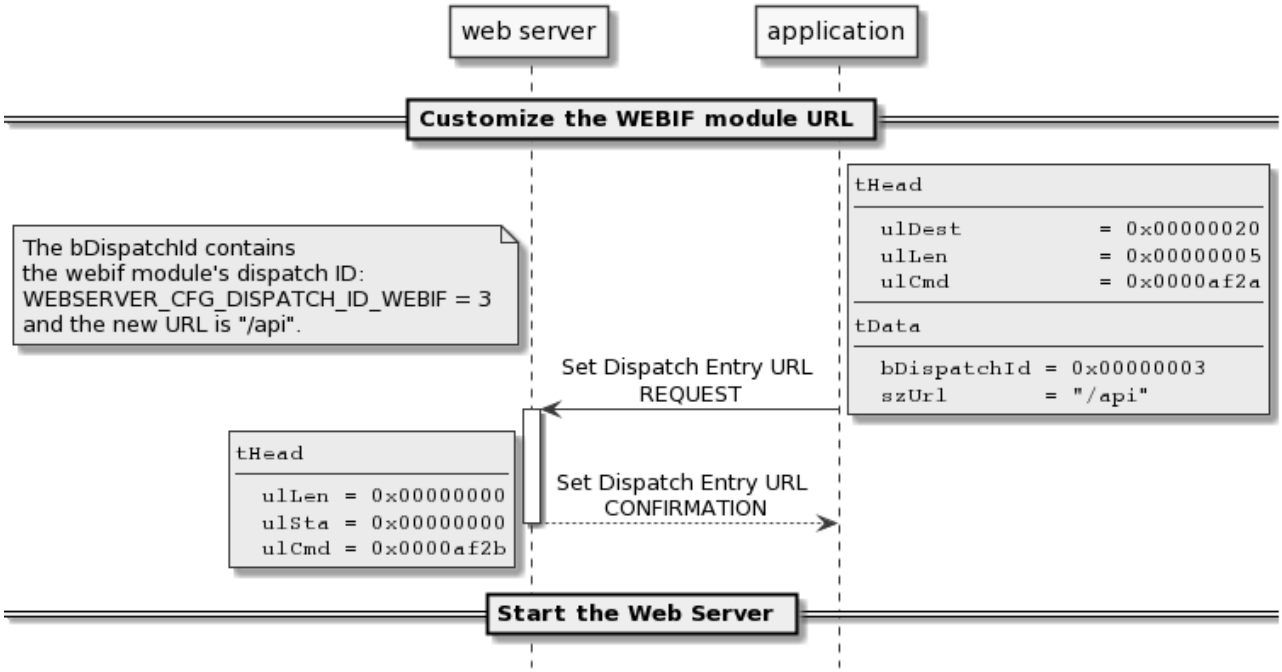
Table 7: WEBIF_SET_DISPATCH_ENTRY_URL_REQ – Set dispatch entry URL request.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t	0	See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF2B	WEBIF_SET_DISPATCH_ENTRY_URL_CNF

Table 8: WEBIF_SET_DISPATCH_ENTRY_URL_CNF – Set dispatch entry URL confirmation

Example



3.3.1.2 Set Dispatch Entry Enabled

This service allows you to enable or disable a dispatch entry within the dispatch table. By default all dispatch entries are enabled.

Note: This service can only be processed successfully if the web-server is not running, e.g. before sending the WEBIF_START_REQ command.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	2	Packet data length in bytes
ulCmd	uint32_t	0xAF28	WEBIF_SET_DISPATCH_ENTRY_ENABLED_REQ
Data			
bDispatchId	uint8_t		ID of the specific dispatch entry
bEnabled	uint8_t		Non-zero to enable, zero to disable.

Table 9: WEBIF_SET_DISPATCH_ENTRY_ENABLED_REQ – Set dispatch entry enabled request.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t	0	See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF29	WEBIF_SET_DISPATCH_ENTRY_ENABLED_CNF

Table 10: WEBIF_SET_DISPATCH_ENTRY_ENABLED_CNF – Set dispatch entry enabled confirmation

3.3.2 Set TCP Ports

This service allows changing the TCP ports that the HTTP and HTTPS (if available) servers are listening on.

Note: This service can only be processed successfully if the web-server is not running, e.g. before sending the WEBIF_START_REQ command.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	8	Packet data length in bytes
ulCmd	uint32_t	0xAF26	WEBIF_SET_TCP_PORTS_REQ
Data			
usHttpPort	uint16_t	1 … 65535	TCP listening port for non-encrypted transfers.
usHttpsPort	uint16_t	1 … 65535	TCP listening port for encrypted transfers.
ulReserved	uint32_t		Reserved

Table 11: WEBIF_SET_TCP_PORTS_REQ – Set TCP ports request.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t	0	See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF27	WEBIF_SET_TCP_PORTS_CNF

Table 12: WEBIF_SET_TCP_PORTS_CNF – Set TCP ports confirmation

3.3.3 Start the Web Server

The web server may have been configured not to start explicitly at system startup. This command gives the control to the application to start the web server only if it is needed and choose the most appropriate time to do so.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulCmd	uint32_t	0xAF22	WEBIF_START_REQ

Table 13: WEBIF_START_REQ – Start the web server request.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t	0	See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF23	WEBIF_START_CNF

Table 14: WEBIF_START_CNF– Start the web server confirmation

After the successful confirmation of the command, the web server starts to listen for client connections on the configured TCP port(s).

3.3.4 Stop the Web Server

Once the web server is started, it may be stopped again. This service gives the control to the application to stop the web server in order to disable it or to modify some of its configuration, before starting it again. It may take up to 4 seconds for the web server to stop and the confirmation packet to be received.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulCmd	uint32_t	0xAF24	WEBIF_STOP_REQ

Table 15: WEBIF_STOP_REQ – Stop the web server request.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t	0	See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF25	WEBIF_STOP_CNF

Table 16: WEBIF_STOP_CNF – Stop the web server confirmation

3.3.5 Enable HTTP Request handling service

As soon as the application is ready to receive and process the HTTP requests, the application can enable the handling of the incoming HTTP requests.

If the handling of the incoming HTTP requests is not enabled, the HTTP Request indications are not transmitted to the application; and the web server will respond“404 Not Found” to all requests on /webif URL

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulCmd	uint32_t	0xAF10	WEBIF_ENABLE_REQUEST_HANDLING_REQ

Table 17: WEBIF_ENABLE_REQUEST_HANDLING_REQ – Enable HTTP request handling request

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t	0	See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF11	WEBIF_ENABLE_REQUEST_HANDLING_CNF

Table 18: WEBIF_ENABLE_REQUEST_HANDLING_CNF – Enable HTTP request handling confirmation

In order to assure the retro-compatibility, the symbols WEBIF_REGISTER_REQ and WEBIF_REGISTER_CNF are still defined.

3.3.6 Disable the HTTP Request handling service

In order to stop the reception of the HTTP request indication, the application can use the disable request.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulCmd	uint32_t	0xAF20	WEBIF_DISABLE_REQUEST_HANDLING_REQ

Table 19: WEBIF_DISABLE_REQUEST_HANDLING_REQ – Disable HTTP request handling request

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t	0	See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF21	WEBIF_DISABLE_REQUEST_HANDLING_CNF

Table 20: WEBIF_DISABLE_REQUEST_HANDLING_CNF – Disable HTTP request handling confirmation

In order to assure the retro-compatibility, the symbols WEBIF_UNREGISTER_REQ and WEBIF_UNREGISTER_CNF are still defined.

3.4 HTTP Request handling

3.4.1 HTTP Request service

As soon as the WebIf stack receives a new HTTP request, it transmits this indication to the application. This indication marks the beginning of the HTTP request handling.

Figure 2 GET request reception and *Figure 5 POST request reception* show typical application cases of the HTTP Request service.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulSrcId	uint32_t	1 – 0xFFFFFFFF	A unique number identifying the HTTP request. It must be stored and used in all following request packets to identify the same HTTP request.
ulLen	uint32_t	136 (V1) 180 (V2)	Packet data length in bytes
ulCmd	uint32_t	0xAF00	WEBIF_HANDLE_HTTP_REQUEST_IND
Data			
ulHttpMethod	uint32_t	1 … 7	HTTP method to be used: ▪ 1: GET ▪ 2: HEAD ▪ 3: POST ▪ 4: PUT ▪ 5: DELETE ▪ 6: TRACE ▪ 7: CONNECT
ulContentLength	uint32_t	0 ... 0xFFFFFFFE, 0xFFFFFFFF	Body size, 0xFFFFFFFF enables reception by chunks.
aPathName	CHAR[128]	[0 … 255] * 128	Null-terminated string of maximal 128 characters containing the path name.
tClientAddr	WEBIF_IP_ADDR_T	N/A	Only valid for the V2 packet (see the ulLen field in the header). The address of the client which transmitted the request.
tAuth	WEBIF_AUTH_T	N/A	This field is filled-out only on IoT firmware with HTTPS enabled. This is the information about the authenticated user using the COM-CPU authentication.

Table 21: WEBIF_HANDLE_HTTP_REQUEST_IND – HTTP Request indication

The “tClientAddr” field contains IP address information about the client from which the request originated. Here are its sub-fields:

Variable	Type	Value / range	Description
ulIpAddr	Uint32_t		Client IP Address (v4)
usPort	Uint16_t	0 … 65535	Client TCP Port
usReserved	Uint16_t	0	Reserved

Table 22: WEBIF_IP_ADDR_T – Client IP Address information

The tAuth field could be used in some firmwares (IoT with HTTPS enabled) to retrieve the user previously authenticated via the HTTP basic authentication in the COM-CPU user database. If the feature is not enabled, the whole field is zeroed. Here are its sub-fields:

Variable	Type	Value / range	Description
ulGroupBitfield	Uint32_t	Bit field	0 = user has no rights, For more information, see Authentication Manager's structure AUTH_USRDB_USER_GROUP_BF_T type
szUserName	CHAR[32]	""	Username (UTF-8)

Table 23: WEBIF_AUTH_T – Authentication information

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t		See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF01	WEBIF_HANDLE_HTTP_REQUEST_RSP

Table 24: WEBIF_HANDLE_HTTP_REQUEST_RSP - HTTP Request response

3.4.2 HTTP Request Body service

The application has to wait for the HTTP request body indications until the end of the HTTP request handling. If there is no data to receive (body is empty; ulContentLength field on the HTTP request indication is equal to zero), the stack will not transmit any HTTP request body indications. With the chunked encoding enabled, the decoding is performed by the stack, data here are without chunk headers.

Figure 5 POST request reception shows a typical application case of the HTTP Request Body service.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulSrcId	uint32_t	1 … 0xFFFFFFFF	Unique number identifying the HTTP request.
ulLen	uint32_t	1040 (=8 + 8 + 1024)	Packet data length in bytes
ulCmd	uint32_t	0xAF02	WEBIF_HANDLE_HTTP_REQUEST_CONTENT_IND
Data			
ulOffset	uint32_t	0 …	Offset of the following data in the body.
ulDataSize	uint32_t	0 … 1024	Size of the following data
aData	uint8_t * 1024	[0 … 255] * 1024	Array of bytes

Table 25: WEBIF_HANDLE_HTTP_REQUEST_CONTENT_IND – HTTP Request Content indication

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t		See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF03	WEBIF_HANDLE_HTTP_REQUEST_CONTENT_RSP

Table 26: WEBIF_HANDLE_HTTP_REQUEST_CONTENT_RSP – HTTP Request Content response

3.4.3 End of HTTP Request service

As soon as the entire body of the HTTP request has been transmitted to the application, the WebIf transmits this indication. This indication marks the end of the HTTP request handling.

Figure 2 GET request reception and *Figure 5 POST request reception* show typical application cases of the End of HTTP Request service.

As displayed there, the application should reply a response packet (see *Table 28*) each time it receives an End of HTTP Request indication.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulSrcId	uint32_t	1 … 0xFFFFFFFF	Unique number identifying the HTTP request.
ulLen	uint32_t	0	Packet data length in bytes
ulCmd	uint32_t	0xAF04	WEBIF_FINISH_HANDLING_HTTP_REQUEST_IND

Table 27: WEBIF_FINISH_HANDLING_HTTP_REQUEST_IND – End of HTTP Request indication

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t	0	See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF05	WEBIF_FINISH_HANDLING_HTTP_REQUEST_RSP

Table 28: WEBIF_FINISH_HANDLING_HTTP_REQUEST_RSP - End of HTTP Request response

3.4.4 Consideration about the request reception with body

In the application, the body reception handling should be accomplished with care regarding the security. If the ContentLength is given (a value between 1 and 0xFFFFFFFE), the stack itself will perform the verification of the body size; in this case (for example, if the reception buffer in the application is too small) the application can reject the request reception at the transmission of a confirmation packet. In case of chunk encoding (ContentLength is equal to 0xFFFFFFFF), the stack cannot perform any size verification.

If an attacker manages to transmit a continued stream of chunks to the WebIf module, the request content indication packets will be transmitted without end; the application has to interrupt the reception by transmitting a confirmation packet with a status not equal to 0.

3.4.5 Get HTTP Request Fields service

In case, the HTTP request contains GET-using fields, the application has to use a request packet (see *Table 29: WEBIF_GET_HTTP_REQUEST_FIELD_REQ – Get HTTP Request Fields request*) to read the parameters and values of a GET request.

Note: The application has to "know" the fields. As a result, fields have to be defined before starting the implementation of the application.

3.4.5.1 „Is In URI“ Boolean

The following text uses the command-line tool “curl” for “Client URL”. This tool is convenient to easily transmit HTTP requests with different HTTP methods or special fields, unlike end-user web browsers.

Fields encoded in the URL:

```
> curl http://webserver.local/service?field1=1&field2=2
```

This results in the following HTTP request:

```
GET /get?field1=1&field2=2 HTTP/1.1\r\n
Host: webserver.local\r\n
```

On the other hand, some fields can be also encoded in the HTTP request header:

```
> curl -H “field1: 1” -H “field2: 2” http://webserver.local/service
```

This results in the following HTTP request:

```
GET /get HTTP/1.1\r\n
Host: webserver.local\r\n
field1: 1
field2: 2
```

If uiIsInUri is set to zero in the request, the requested field is in the HTTP header. Otherwise, the requested field is in the URL.

3.4.5.2 Packets

The following request packet allows to retrieve the content of a single field from an HTTP request that uses the GET method. You have to specify the name of this field in the null-terminated string aName. If the content of the specified field could successfully be retrieved, it is delivered in variable aContent of the confirmation packet.

Figure 6 POST request header field extraction shows a typical application case of the Get HTTP Request Fields service.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulDestId	uint32_t		The unique number identifying the HTTP request retrieved from the ulSrcId of a previous WEBIF_HANDLE_HTTP_REQUEST_IND packet
ulLen	uint32_t	132 (= 4 + 128)	Packet data length in bytes
ulCmd	uint32_t	0xAF06	WEBIF_GET_HTTP_REQUEST_FIELD_REQ
Data			
uiIsInUri	uint32_t	0, 1	0 if the field is in HTTP header, 1 if the field is in URL.
aName	CHAR[128]	[0 … 255] * 128	Null-terminated string with the field name

Table 29: WEBIF_GET_HTTP_REQUEST_FIELD_REQ – Get HTTP Request Fields request

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	128	Packet data length in bytes
ulSta	uint32_t		See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF07	WEBIF_GET_HTTP_REQUEST_FIELD_CNF
Data			
aContent	CHAR[128]	[0 … 255] * 128	Null-terminated string with the field content.

Table 30: WEBIF_GET_HTTP_REQUEST_FIELD_CNF - Get HTTP Request Fields confirmation

3.5 HTTP Response generation

After receiving all packets of a HTTP request, the application has to generate the HTTP response.

3.5.1 HTTP Response service

The application transmits this packet in order to begin to generate the HTTP response. This request marks the beginning of the HTTP response generation.

Figure 3 GET request response header generation and *Figure 7 POST request response generation* show typical application cases of the HTTP Response service.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulDestId	uint32_t		The unique number identifying the HTTP request retrieved from the ulSrcId of a previous WEBIF_HANDLE_HTTP_REQUEST_IND packet
ulLen	uint32_t	8 (V1) 136 (V2)	Packet data length in bytes
ulCmd	uint32_t	0xAF08	WEBIF_GENERATE_HTTP_RESPONSE_REQ
Data			
ulStatusCode	uint32_t	1xx, 2xx, 4xx, 5xx,	HTTP status code (200 OK)
ulContentLength	uint32_t	0 … 0xFFFFFFFE 0XFFFFFFFF	Size of the HTTP response body. Chunk Encoding.
aReasonPhrase	Char	[0 … 127] * 128	Only valid for the V2 packet (see the ulLen field in the header). The reason phrase.

Table 31: WEBIF_GENERATE_HTTP_RESPONSE_REQ – Generate HTTP Response request

Note: The ulContentLength value shall be given here to know, when the body is completely received from the application side. Unlike the ulStatusCode value, the ulContentValue value will not be used to generate the content of the HTTP response header. It would be probably necessary to the send the “Content-Length” field explicitly: see the remark in section 3.5.2.1. If the V1 packet is used (with a data size (the ulLen field) of 8 bytes) or the reason phrase (the aReasonPhrase field in the V2 packet) is not given (the first character is 0), the integrated default reason phrase will be transmitted in the HTTP response header. There is a dedicated default reason phrase for the following status codes: 301, 304, 400, 401, 403, 404, 500, 501 and 503. For the other status codes, the integrated reason phrase is “Unknown”.
Otherwise, the reason phrase given in this packet is transmitted in the HTTP response header. Do not forget to use the V2 packet by setting the data size (the ulLen field) to the correct (V2) length above.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t		See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF09	WEBIF_GENERATE_HTTP_RESPONSE_CNF

Table 32: WEBIF_GENERATE_HTTP_RESPONSE_CNF - Generate HTTP Response confirmation

3.5.2 Set HTTP Response Fields service

The application uses this request to transmit HTTP response fields. Use this service during the HTTP response generation only! If ulSta in the confirmation packet is equal to zero, the HTTP response field specified in variable aName of the request packet could be set to the content specified within variable aContent.

Figure 3 GET request response header generation shows a typical application case of the Set HTTP Response Fields service.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulDestId	uint32_t		The unique number identifying the HTTP request retrieved from the ulSrcId of a previous WEBIF_HANDLE_HTTP_REQUEST_IND packet
ulLen	uint32_t	2 * 128	Packet data length in bytes
ulCmd	uint32_t	0xAF0A	WEBIF_GENERATE_HTTP_RESPONSE_FIELD_REQ
Data			
aName	CHAR[128]	[0 … 255] * 128	Null-terminated string with the name of the field
aContent	CHAR[128]	[0 … 255] * 128	Null-terminated string with the field content

Table 33: WEBIF_GENERATE_HTTP_RESPONSE_FIELD_REQ – Set HTTP Response Fields request

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t		See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF0B	WEBIF_GENERATE_HTTP_RESPONSE_FIELD_CNF

Table 34: WEBIF_GENERATE_HTTP_RESPONSE_FIELD_CNF - Set HTTP Response Fields confirmation

3.5.2.1 Remark about the necessary response fields

Except on some cases (described in section 3.3.2 of the RFC7230 https://tools.ietf.org/html/rfc7230#section-3.3.2), the “Content-Length” field is needed by the client in order to know the size of the transmitted bodyIn normal case, the application shall explicitly (via the Set HTTP Response Fields service) define the “Content-Length” field, it would not be defined by the WebIf interface automatically.

The application should also define the “Content-Type” header field, which helps the client to recognize the data format.

3.5.3 HTTP Response Body service

The application uses this request to transmit the body of the HTTP response. This request shall be only transmitted during the HTTP response generation.

Figure 4 GET request response body generation shows a typical application case of the HTTP Response Body service.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulDestId	uint32_t		The unique number identifying the HTTP request retrieved from the ulSrcId of a previous WEBIF_HANDLE_HTTP_REQUEST_IND packet
ulLen	uint32_t	1028 (=4 + 1024)	Packet data length in bytes
ulCmd	uint32_t	0xAF0C	WEBIF_GENERATE_HTTP_RESPONSE_CONTENT_REQ
Data			
ulDataSize	uint32_t	0 … 1024	Size of the following data
aData	uint8_t * 1024	[0 … 255] * 1024	Array of bytes

Table 35: WEBIF_GENERATE_HTTP_RESPONSE_CONTENT_REQ - HTTP Response Content request

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t		See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF0D	WEBIF_GENERATE_HTTP_RESPONSE_CONTENT_CNF

Table 36: WEBIF_GENERATE_HTTP_RESPONSE_CONTENT_CNF - HTTP Response Content confirmation

3.5.4 End of HTTP Response service

As soon as all HTTP response fields and the body are transmitted, the application transmits this request to trigger the transmission of the HTTP response. This request marks the end of the HTTP response generation.

Figure 4 GET request response body generation and *Figure 7 POST request response generation* show typical application cases of the End of HTTP Response service.

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulDestId	uint32_t		The unique number identifying the HTTP request retrieved from the ulSrcId of a previous WEBIF_HANDLE_HTTP_REQUEST_IND packet
ulLen	uint32_t	0	Packet data length in bytes
ulCmd	uint32_t	0xAF0E	WEBIF_FINISH_GENERATION_HTTP_RESPONSE_REQ

Table 37: WEBIF_FINISH_GENERATION_HTTP_RESPONSE_REQ – End of HTTP Response request

Packet description

Variable	Type	Value / range	Description
ulDest	uint32_t		Destination
ulLen	uint32_t	0	Packet data length in bytes
ulSta	uint32_t		See section Status codes / Error codes on page 35
ulCmd	uint32_t	0xAF0F	WEBIF_FINISH_GENERATION_HTTP_RESPONSE_CNF

Table 38: WEBIF_FINISH_GENERATION_HTTP_RESPONSE_CNF - End of HTTP Response confirmation

4 Examples

4.1 GET Request Example

4.1.1 Request Generation

An example GET request can be generated using the „curl“ tool. Apart from the requested file name it also contains user authentication information.

```
> curl --basic -u root:password -i http://{ip}/webif/index.html
```

Here is the actual HTTP GET request sent to the web server

```
GET /webif/index.html HTTP/1.1
Host: 192.168.210.12
Authorization: Basic cm9vdDpwYXNzd29yZA==
User-Agent: curl/7.58.0
Accept: */*
```

The content of the requested index.html file used in the example below is a simple html file 364 bytes long:

```
<!DOCTYPE html>
<html lang="en">
<meta charset="UTF-8">
<title>Page Title</title>
<meta name="viewport" content="width=device-width,initial-scale=1">
<link rel="stylesheet" href="">
<style>
</style>
<script src=""></script>
<body>

<div class="">
 <h1>This is a Heading</h1>
 <p>This is a paragraph.</p>
 <p>This is another paragraph.</p>
</div>

</body>
</html>
```

4.1.2 Receiving the Request

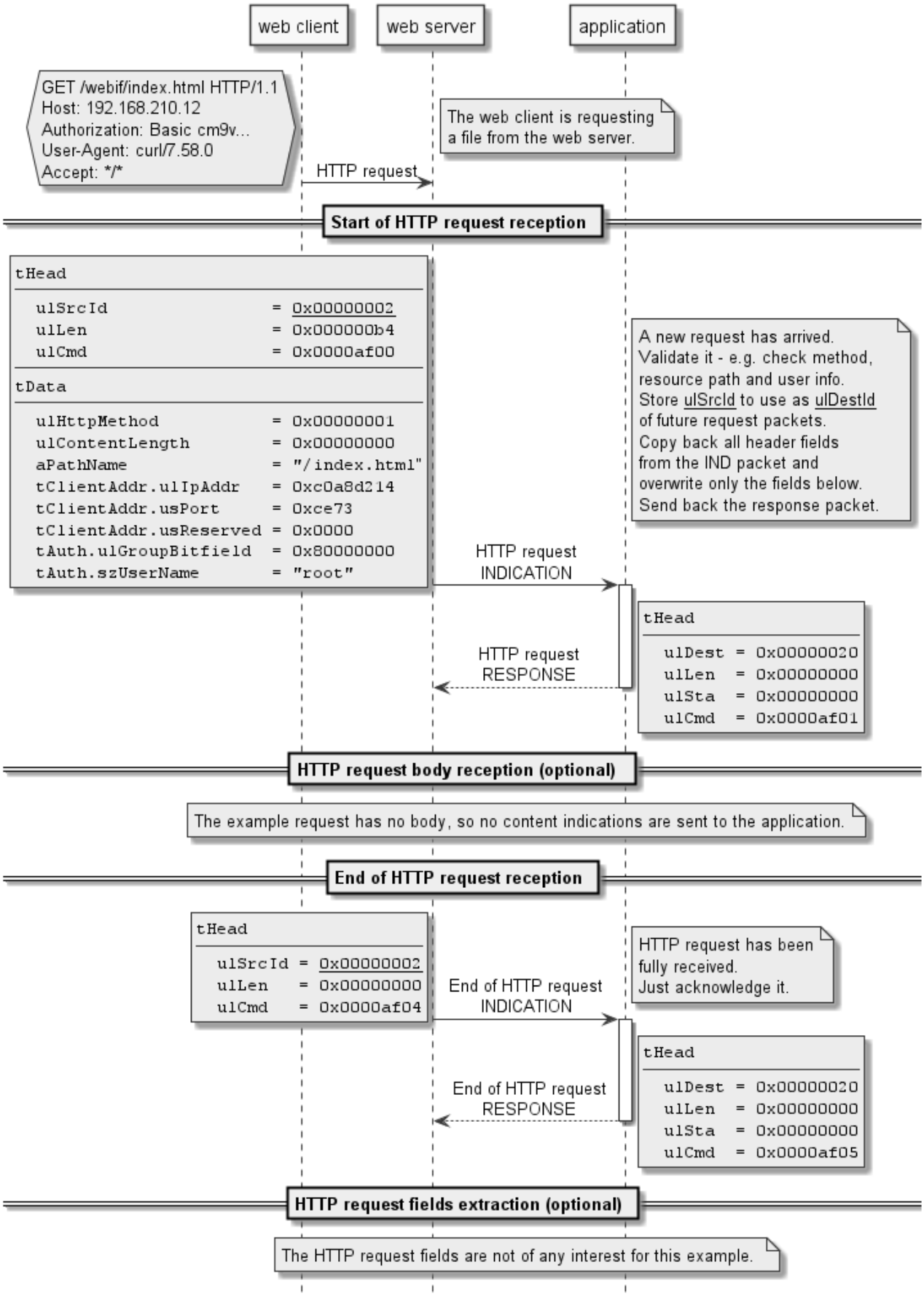


Figure 2 GET request reception

4.1.3 Responding to the Request (Header Generation)

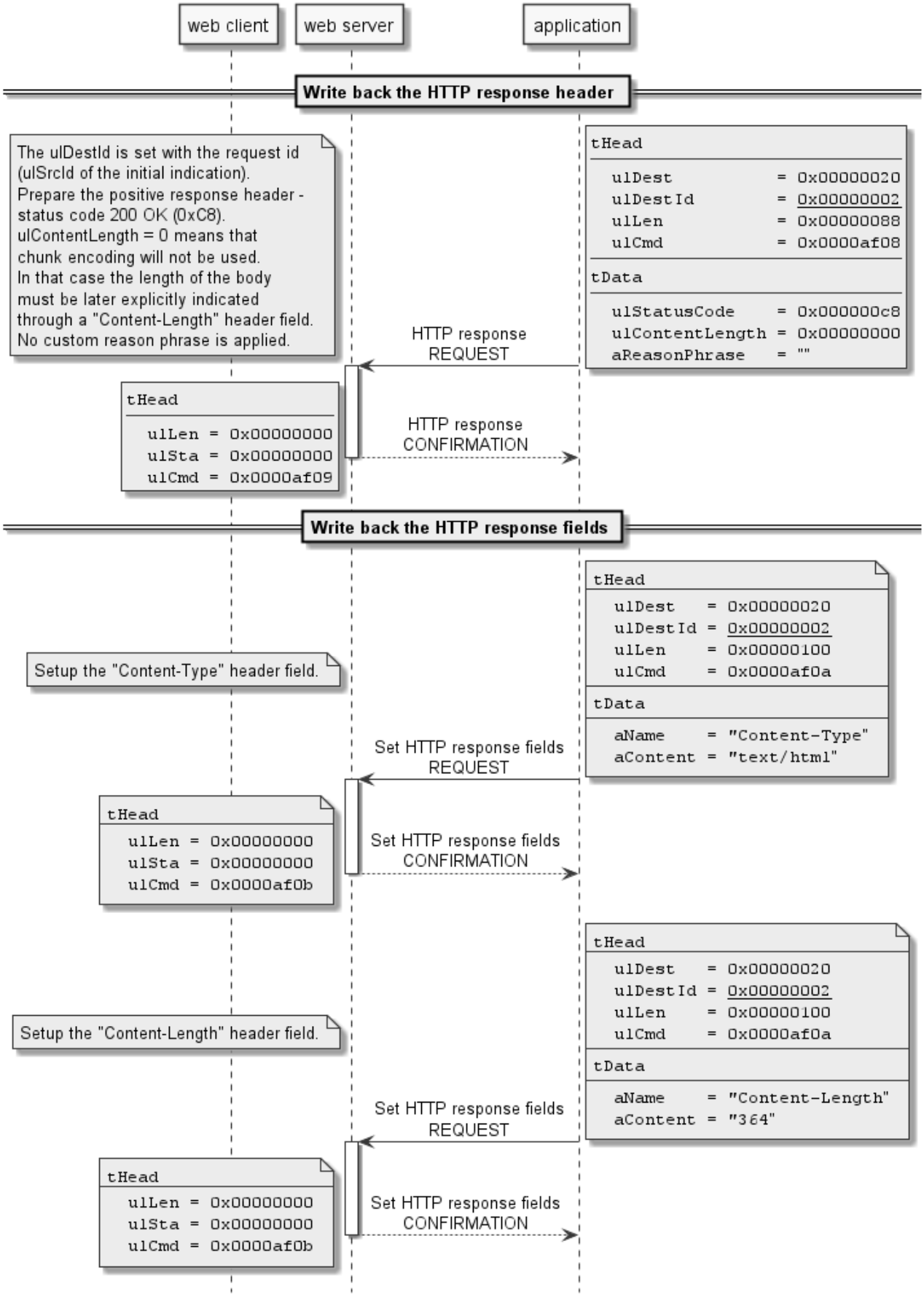


Figure 3 GET request response header generation

4.1.4 Responding to the Request (Content Body)

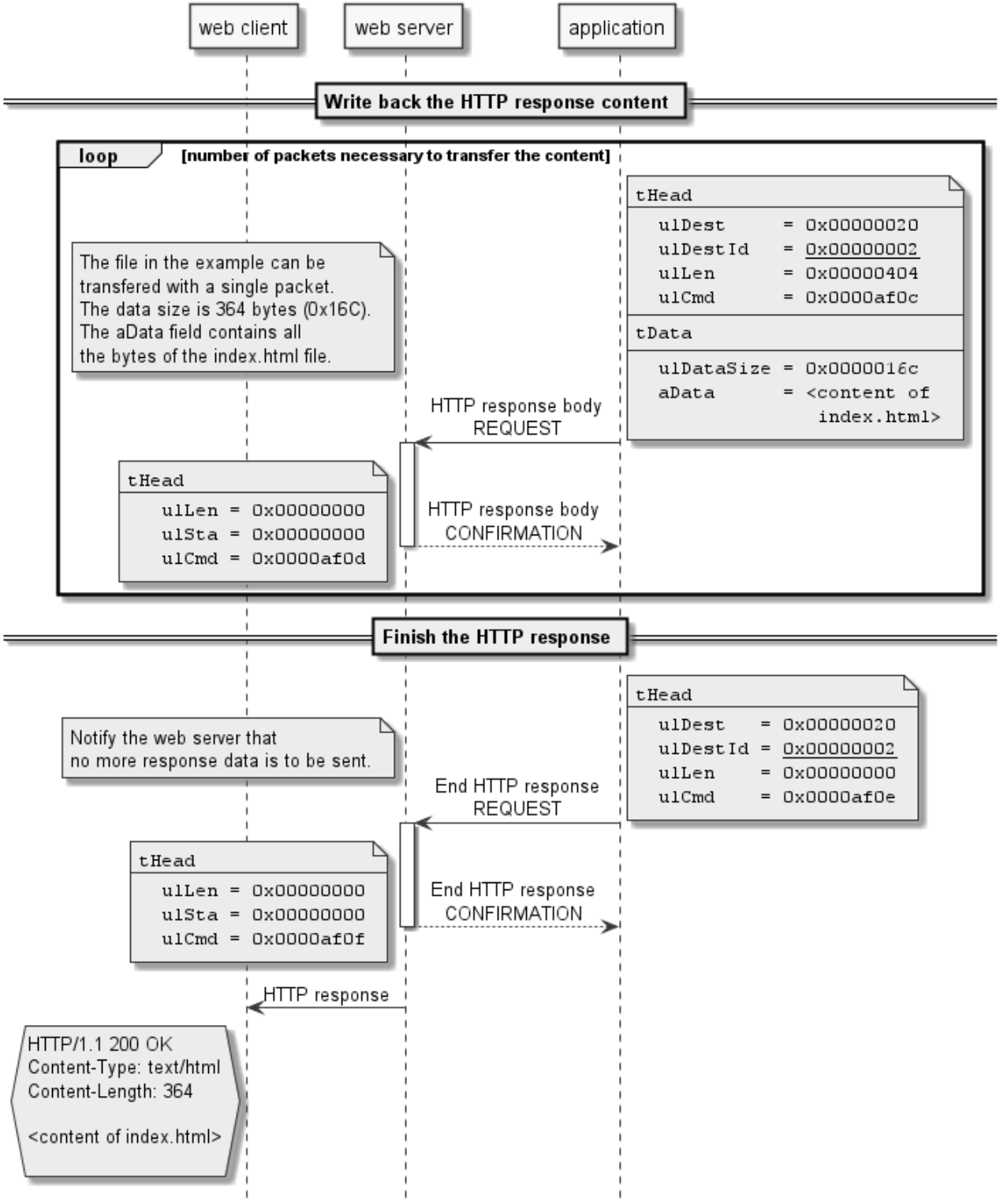


Figure 4 GET request response body generation

4.2 POST Request Example

4.2.1 Request Generation

An example POST request can be generated using the „curl“ tool. The following request is used to transfer url-encoded data to the web server. The (form) data is part of the request body.

```
> curl -X POST http://{ip}/webif/form -H "Content-Type: application/x-www-form-urlencoded" -d "param1=value1&param2=value2"
```

Here is the actual HTTP POST request sent to the web server

```
POST /webif/form HTTP/1.1
Host: 192.168.210.12
User-Agent: curl/7.58.0
Accept: */*
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

param1=value1&param2=value2
```

The web server does not need to respond with a content, so for the purpose of simplifying the example a “204 No Content” response status will be returned. Returning a content is illustrated in the GET Request Example, see 4.1.4.

4.2.2 Receiving the Request

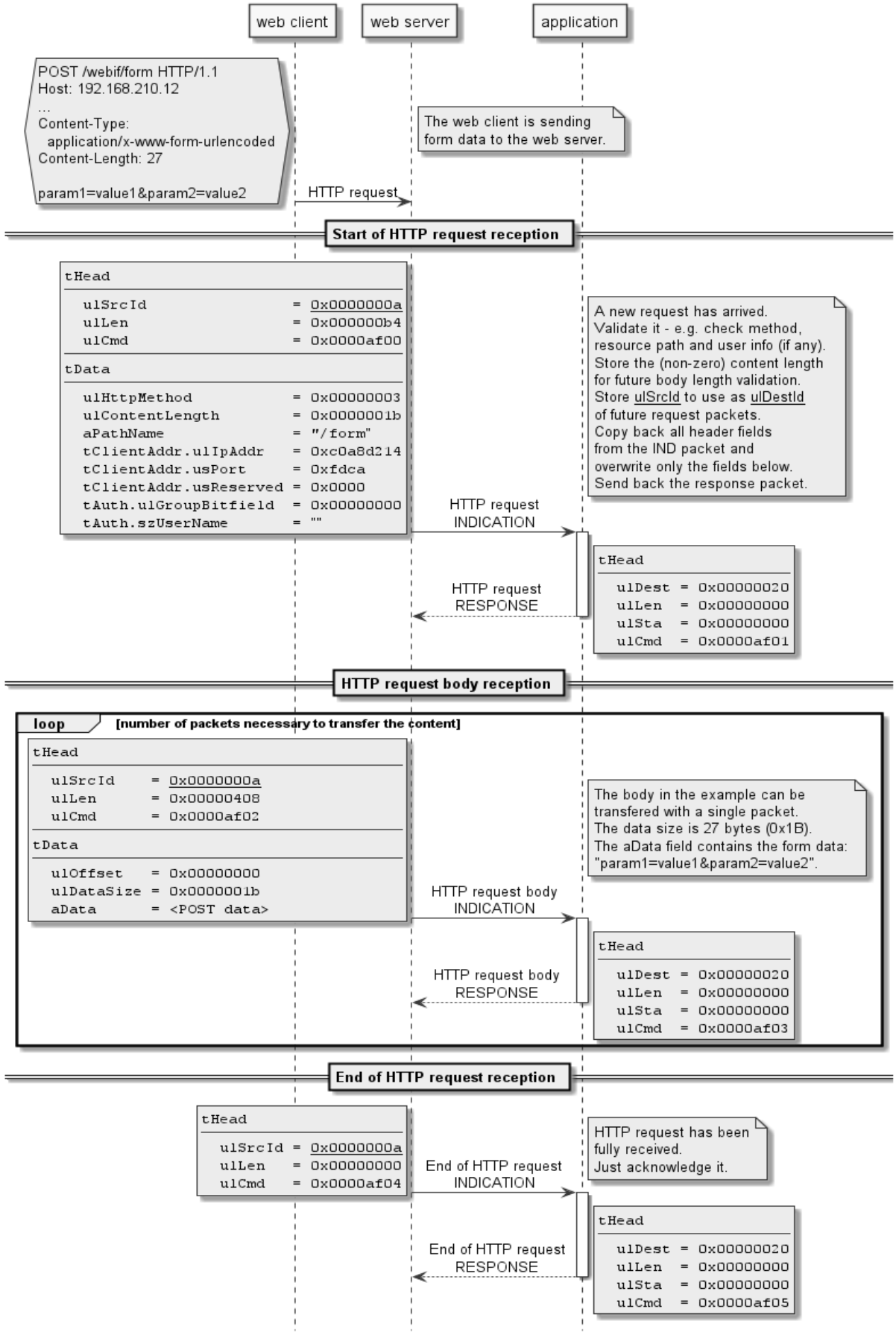


Figure 5 POST request reception

4.2.3 Extracting Request Fields

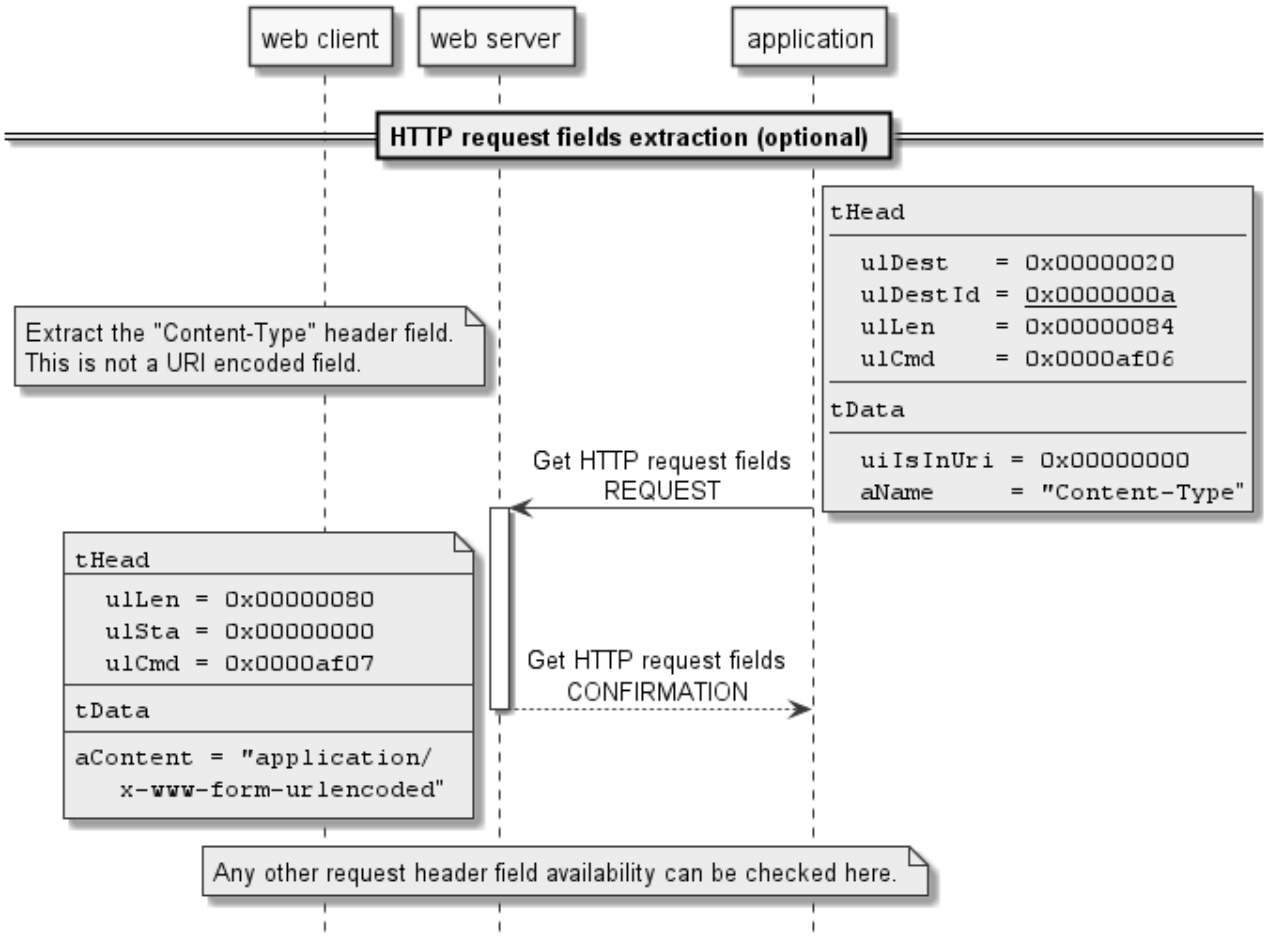


Figure 6 POST request header field extraction

4.2.4 Responding to the Request

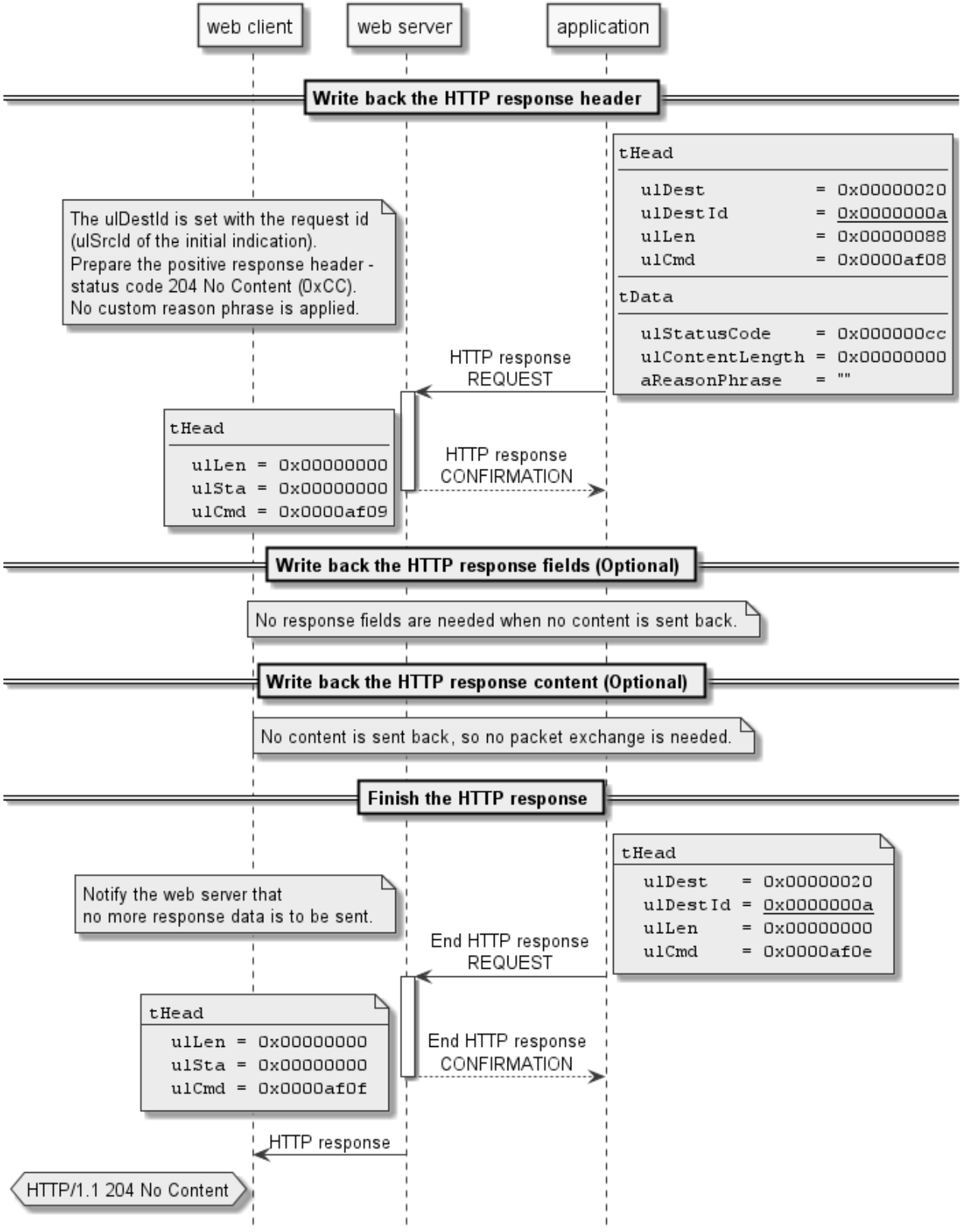


Figure 7 POST request response generation

4.3 Request termination example

4.3.1 How to react to malformed requests

A request may sometimes have an unexpected structure. A file being uploaded may be bigger than the application can handle or the web client may purposely try to flood the web server with too much data (DoS attack).

```
> curl -d "@big_file.txt" -X POST http://{ip}/webif/upload
```

Any such request that may cause an endless sequence of indications can be terminated by the application by returning a response packet with the ulSta field set to a non-zero value. This will lead to the web server generating an error response “500 Internal Server Error” and a termination of the exchange.

4.3.2 Sequence Diagram

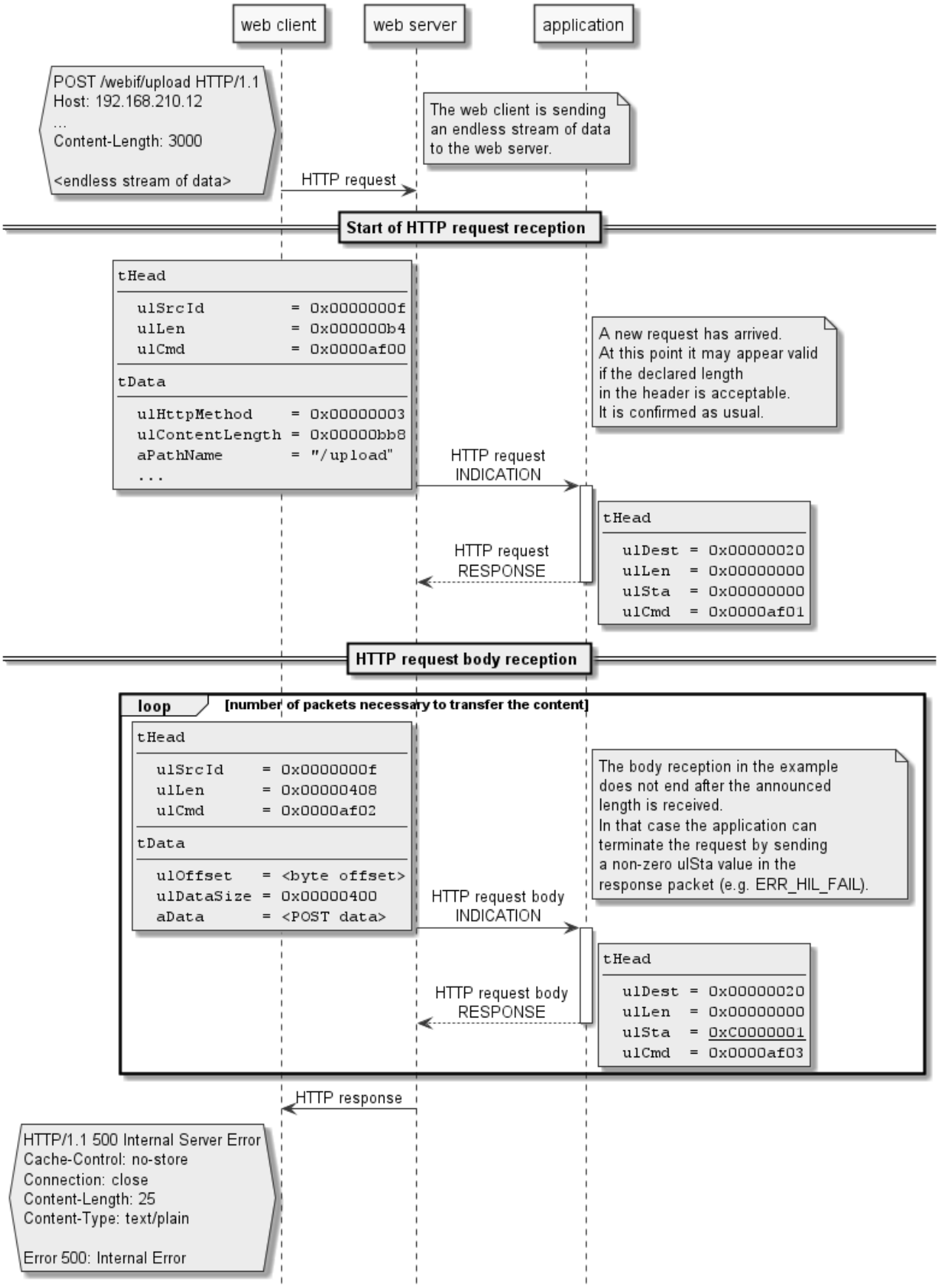


Figure 8 Malformed POST request termination

5 Status codes / Error codes

Hexadecimal value	Definition and description
0xC0000001	ERR_HIL_FAIL Common error, detailed error information optionally present in the data area of packet.
0xC0000006	ERR_HIL_UNKNOWN_DESTINATION_ID Unknown Destination Id in Packet received. The value does not match the identifier of any active HTTP request.
0xC0000119	ERR_HIL_NOT_CONFIGURED Configuration not available. The web server component is not configured to support delayed start.
0xC0000205	ERR_HIL_DATA_ALREADY_SET The data was already set. The web server is already started.

Table 39: WebIf status and error codes

6 Appendix

6.1 List of tables

6.2 List of figures

6.3 Legal notes

Copyright

Important notes

Utmost care was/is given in the preparation of the documentation at hand consisting of a user's manual, operating manual and any other document type and accompanying texts. However, errors cannot be ruled out. Therefore, we cannot assume any guarantee or legal responsibility for erroneous information or liability of any kind. You are hereby made aware that descriptions found in the user's manual, the accompanying texts and the documentation neither represent a guarantee nor any indication on proper use as stipulated in the agreement or a promised attribute. It cannot be ruled out that the user's manual, the accompanying texts and the documentation do not completely match the described attributes, standards or any other data for the delivered product. A warranty or guarantee with respect to the correctness or accuracy of the information is not assumed.

We reserve the right to modify our products and the specifications for such as well as the corresponding documentation in the form of a user's manual, operating manual and/or any other document types and accompanying texts at any time and without notice without being required to notify of said modification. Changes shall be taken into account in future manuals and do not represent an obligation of any kind, in particular there shall be no right to have delivered documents revised. The manual delivered with the product shall apply.

Under no circumstances shall Hilscher Gesellschaft für Systemautomation mbH be liable for direct, indirect, ancillary or subsequent damage, or for any loss of income, which may arise after use of the information contained herein.

Liability disclaimer

The hardware and/or software was created and tested by Hilscher Gesellschaft für Systemautomation mbH with utmost care and is made available as is. No warranty can be assumed for the performance or flawlessness of the hardware and/or software under all application conditions and scenarios and the work results achieved by the user when using the hardware and/or software. Liability for any damage that may have occurred as a result of using the hardware and/or software or the corresponding documents shall be limited to an event involving willful intent or a grossly negligent violation of a fundamental contractual obligation. However, the right to assert damages due to a violation of a fundamental contractual obligation shall be limited to contract-typical foreseeable damage.

It is hereby expressly agreed upon in particular that any use or utilization of the hardware and/or software in connection with

- Flight control systems in aviation and aerospace;
- Nuclear fission processes in nuclear power plants;
- Medical devices used for life support and
- Vehicle control systems used in passenger transport

shall be excluded. Use of the hardware and/or software in any of the following areas is strictly prohibited:

- For military purposes or in weaponry;
- For designing, engineering, maintaining or operating nuclear systems;
- In flight safety systems, aviation and flight telecommunications systems;
- In life-support systems;
- In systems in which any malfunction in the hardware and/or software may result in physical injuries or fatalities.

You are hereby made aware that the hardware and/or software was not created for use in hazardous environments, which require fail-safe control mechanisms. Use of the hardware and/or software in this kind of environment shall be at your own risk; any liability for damage or loss due to impermissible use shall be excluded.

Warranty

Hilscher Gesellschaft für Systemautomation mbH hereby guarantees that the software shall run without errors in accordance with the requirements listed in the specifications and that there were no defects on the date of acceptance. The warranty period shall be 12 months commencing as of the date of acceptance or purchase (with express declaration or implied, by customer's conclusive behavior, e.g. putting into operation permanently).

The warranty obligation for equipment (hardware) we produce is 36 months, calculated as of the date of delivery ex works. The aforementioned provisions shall not apply if longer warranty periods are mandatory by law pursuant to Section 438 (1.2) BGB, Section 479 (1) BGB and Section 634a (1) BGB [Bürgerliches Gesetzbuch; German Civil Code] If, despite of all due care taken, the delivered product should have a defect, which already existed at the time of the transfer of risk, it shall be at our discretion to either repair the product or to deliver a replacement product, subject to timely notification of defect.

The warranty obligation shall not apply if the notification of defect is not asserted promptly, if the purchaser or third party has tampered with the products, if the defect is the result of natural wear, was caused by unfavorable operating conditions or is due to violations against our operating regulations or against rules of good electrical engineering practice, or if our request to return the defective object is not promptly complied with.

Costs of support, maintenance, customization and product care

Please be advised that any subsequent improvement shall only be free of charge if a defect is found. Any form of technical support, maintenance and customization is not a warranty service, but instead shall be charged extra.

Additional guarantees

Although the hardware and software was developed and tested in-depth with greatest care, Hilscher Gesellschaft für Systemautomation mbH shall not assume any guarantee for the suitability thereof for any purpose that was not confirmed in writing. No guarantee can be granted whereby the hardware and software satisfies your requirements, or the use of the hardware and/or software is uninterruptable or the hardware and/or software is fault-free.

It cannot be guaranteed that patents and/or ownership privileges have not been infringed upon or violated or that the products are free from third-party influence. No additional guarantees or promises shall be made as to whether the product is market current, free from deficiency in title, or can be integrated or is usable for specific purposes, unless such guarantees or promises are required under existing law and cannot be restricted.

Confidentiality

The customer hereby expressly acknowledges that this document contains trade secrets, information protected by copyright and other patent and ownership privileges as well as any related rights of Hilscher Gesellschaft für Systemautomation mbH. The customer agrees to treat as confidential all of the information made available to customer by Hilscher Gesellschaft für Systemautomation mbH and rights, which were disclosed by Hilscher Gesellschaft für Systemautomation mbH and that were made accessible as well as the terms and conditions of this agreement itself.

The parties hereby agree to one another that the information that each party receives from the other party respectively is and shall remain the intellectual property of said other party, unless provided for otherwise in a contractual agreement.

The customer must not allow any third party to become knowledgeable of this expertise and shall only provide knowledge thereof to authorized users as appropriate and necessary. Companies associated with the customer shall not be deemed third parties. The customer must obligate authorized users to confidentiality. The customer should only use the confidential information in connection with the performances specified in this agreement.

The customer must not use this confidential information to his own advantage or for his own purposes or rather to the advantage or for the purpose of a third party, nor must it be used for commercial purposes and this confidential information must only be used to the extent provided for in this agreement or otherwise to the extent as expressly authorized by the disclosing party in written form. The customer has the right, subject to the obligation to confidentiality, to disclose the terms and conditions of this agreement directly to his legal and financial consultants as would be required for the customer's normal business operation.

Export provisions

The delivered product (including technical data) is subject to the legal export and/or import laws as well as any associated regulations of various countries, especially such laws applicable in Germany and in the United States. The products / hardware / software must not be exported into such countries for which export is prohibited under US American export control laws and its supplementary provisions. You hereby agree to strictly follow the regulations and to yourself be responsible for observing them. You are hereby made aware that you may be required to obtain governmental approval to export, reexport or import the product.

6.4 Contacts

Headquarters

Germany
Hilscher Gesellschaft für Systemautomation mbH
Rheinstraße 15
D-65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-mail: info@hilscher.com
Support
Phone: +49 (0) 6190 9907-990
E-mail: hotline@hilscher.com

Subsidiaries

China
Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-mail: info@hilscher.cn
Support
Phone: +86 (0) 21-6355-5161
E-mail: cn.support@hilscher.com

France
Hilscher France S.a.r.l.
69800 Saint Priest
Phone: +33 (0) 4 72 37 98 40
E-mail: info@hilscher.fr
Support
Phone: +33 (0) 4 72 37 98 40
E-mail: fr.support@hilscher.com

India
Hilscher India Pvt. Ltd.
Pune, Delhi, Mumbai, Bangalore
Phone: +91 8888 750 777
E-mail: info@hilscher.in
Support
Phone: +91 8108884011
E-mail: info@hilscher.in

Italy
Hilscher Italia S.r.l.
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-mail: info@hilscher.it
Support
Phone: +39 02 25007068
E-mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-mail: info@hilscher.jp
Support
Phone: +81 (0) 3-5362-0521
E-mail: jp.support@hilscher.com

Republic of Korea
Hilscher Korea Inc.
13494, Seongnam, Gyeonggi
Phone: +82 (0) 31-739-8361
E-mail: info@hilscher.kr
Support
Phone: +82 (0) 31-739-8363
E-mail: kr.support@hilscher.com

Austria
Hilscher Austria GmbH
4020 Linz
Phone: +43 732 931 675-0
E-mail: sales.at@hilscher.com
Support
Phone: +43 732 931 675-0
E-mail: at.support@hilscher.com

Switzerland
Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-mail: info@hilscher.ch
Support
Phone: +41 (0) 32 623 6633
E-mail: support.swiss@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-mail: info@hilscher.us
Support
Phone: +1 630-505-5301
E-mail: us.support@hilscher.com