

# UDS\_GW Module Documentation

CSM GmbH, Filderstadt, Germany  
[www.csm.de/unicom/](http://www.csm.de/unicom/)

May 15, 2025

Date	Version	Name	Changes
2013-03-25	1.00	CSM/RN	first release
2013-07-05	1.01	CSM/RN	UDS Download
2016-03-02	1.04	CSM/RN	MULTI_MODULE
2016-11-23	1.04a	CSM/RN	Typos corrected
2017-03-14	1.05	CSM/TO	Tester present added
2019-03-21	1.20	CSM/RN	configurable download, extended download
2020-09-02	1.30	CSM/TO	security algorithm command added
2023-05-04	1.70	CSM/RN	more error codes
2023-05-09	1.80	CSM/RN	another download command
2023-10-11	1.90	CSM/RN	CONFIG_MODULE command
2024-04-05	2.10	CSM/RN	FASTFLASH
2024-06-19	2.20	CSM/RN	Tester Present, Timeouts
2024-07-04	2.40	CSM/RN	controlling of download size
2024-07-05	2.50	CSM/RN	configurable internal command time-out
2025-05-15	2.70	CSM/RN	FASTFLASH more configurable

All concepts and procedures introduced in this document are intellectual properties of CSM GmbH. Copying or usage by third parties without written permission of CSM GmbH is strictly prohibited. All trademarks mentioned in this document are properties of their respective owners. **This document is subject to changes without notice!**



CSM GmbH Computer-Systeme-Messtechnik  
Raiffeisenstrasse 36 70794 Filderstadt-Bonlanden  
Phone ++49 711 77964 0 Fax ++49 711 77964 40  
mailto:unicom@csm.de http://www.csm.de

Copyright © 2025 by CSM GmbH

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Overview</b>	<b>4</b>
<b>3</b>	<b>Loading and Configuration</b>	<b>5</b>
3.1	MODULE Command . . . . .	5
3.2	CONFIG_INTERFACE Command . . . . .	7
<b>4</b>	<b>FASTFLASH</b>	<b>8</b>
<b>5</b>	<b>UDS_GW Commands</b>	<b>10</b>
5.1	UDS_GW::CONFIG_MODULE (1) . . . . .	10
5.2	UDS_GW::READ_VERSION (2) . . . . .	12
5.3	UDS_GW::CONFIG_FASTFLASH (3) . . . . .	13
5.4	UDS_GW::CONF_DOWNLOAD (4) . . . . .	15
5.5	UDS_GW::SECURITY_ALGORITHM (6) . . . . .	17
5.6	UDS_GW::CONFIG_TIMEOUTS (7) . . . . .	18
5.7	UDS_GW::FASTFLASH_UDS (14) . . . . .	19
5.8	UDS_GW::DOWNLOAD (22) . . . . .	21
5.9	UDS_GW::DOWNLOAD_EX (23) . . . . .	22
5.10	UDS_GW::DOWNLOAD_EXX (24) . . . . .	23
5.11	UDS_GW::DOWNLOAD_EXXX (25) . . . . .	24
5.12	UDS_GW::TESTER_PRESENT (96) . . . . .	25
5.13	UDS_GW::GATEWAY (99) . . . . .	27
5.14	UDS_GW::ErrorCodes . . . . .	29

# Chapter 1

## Introduction

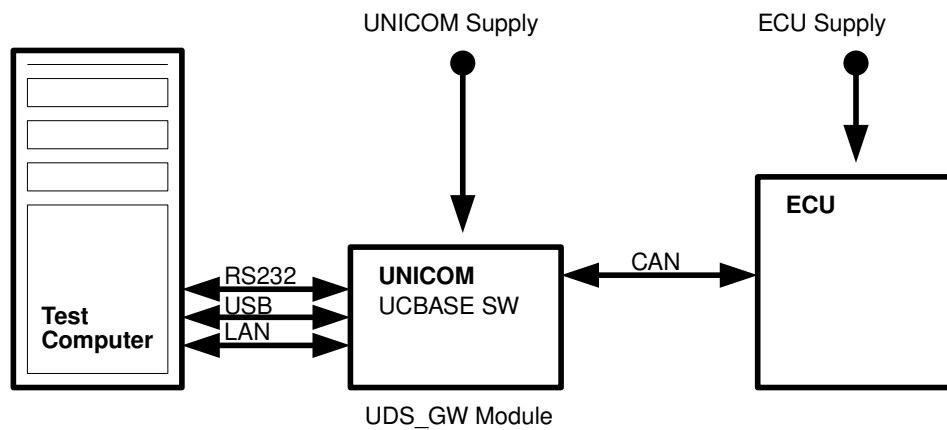
*UDS\_GW* is a module for extending the *UCBASE* software running on *UNICOM*. It implements UDS over CAN protocol ("UDS Gateway").

## Chapter 2

# Overview

To use UNICOM device with UDS\_GW module, UCBASE software version 1.21 or newer must be installed on UNICOM.

The figure below shows the components of the system.



## Chapter 3

# Loading and Configuration

### 3.1 MODULE Command

This command downloads and runs the UDS\_GW module.

#### Command, form 1 (unload module)

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	0xC0	20,40,41,42,43	

#### Command, form 2 (load module)

byte 0	byte 1	byte 2	byte 3	...	byte N-2	byte N-1	byte N
len	ecu	cmd	mod 1	...	mod m	EOS mod	cks
N=m+4	0xC0	20,40,41,42,43				0	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>mod</b>	filename of module (here: uds_gw.mod)
<b>EOS mod</b>	end-of-string of module filename (0)
<b>cks</b>	checksum of telegram

#### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	0xC0		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

### Remarks

- Before the UDS\_GW module can be applied, the CAN controller(s) that should be used for UDS protocol must be configured properly with the commands *INIT\_CAN(98)* and *CAN\_CONFIG(94)* of the UCBASE software.
- After loading the module with command 20, at least one interface slot must be configured for *MODULE* interface using the *CONFIG\_UNICOM(1)* command of UCBASE software. After that, the commands that UDS\_GW implements can be used.
- If the module has been loaded with one of the *MULTI\_MODULE* commands (40..43), exactly the slot must be configured for *MODULE* interface where the module is loaded to. Please refer *ucbase.pdf* for more information about *MULTI\_MODULE*.
- Per default, the number of slot is correlated with the number of CAN controller that is used for UDS: slot 0 uses CAN 1, slot 1 uses CAN 2 and so on. The mapping of CAN controllers can be changed with the *CONFIG\_INTERFACE(4)* command (ref. chapter 3.2 on page 7).

## 3.2 CONFIG\_INTERFACE Command

The *CONFIG\_INTERFACE* command can configure an interface that is activated at the specified slot. If slot is configured for *MODULE* interface and the UDS\_GW module is loaded, the UDS\_GW module can be configured this way.

The syntax of CONFIG\_INTERFACE command for UDS\_GW module is identical to that one for *STPonCAN* interface described in *ucbase.pdf* because STPonCAN uses the same underlying transport protocol as UDS.



## Chapter 4

# FASTFLASH

A special variant of FASTFLASH is implemented by the module. It is embedded into UDS commands:

- A *Request Download* command (0x34 ...) which contains start address in flash and size of the data block
- A *Routine Control* command (0x31 0x01 0xF3F3) which starts FASTFLASH
- After ending FASTFLASH, a *Request Transfer Exit* command (0x37) which ends the sequence.

FASTFLASH itself works as known. The *New Generation* protocol is used. One or two CANs can be configured (FASTFLASH SINGLE or DUAL). After module start, FASTFLASH SINGLE is selected.

The maximum size of data portion which is used by FASTFLASH can also be adjusted. After module start, a size of 256 bytes is selected.

The primary CAN for FASTFLASH is being selected by the interface slot which is used to talk with the module (slot 0: CAN1, slot1: CAN2...), however, the mapping between slots and CANs can be adjusted by the CONFIG\_INTERFACE(4) command (ref. chapter 3.2 on page 7). This settings are also taken into account.

The secondary CAN for FASTFLASH DUAL, if activated, must be different to the primary one. Refer to CONFIG\_FASTFLASH(3) command (chapter 5.3 on page 13).

The CAN configuration for FASTFLASH must be done prior starting FASTFLASH by using the INIT\_CAN(98) command, virtual CAN 0 (refer to `ucbase.pdf`).

Only X\_FASTFLASH(14) and X\_FASTFLASH\_TAB(15) with one data set are supported by the module.

Only BINARY files are supported to be used.

There is another command for starting this FASTFLASH sequence, by using an alternate command format:

FASTFLASH\_UDS(14), refer to chapter 5.7 on page 19.

## Chapter 5

# UDS\_GW Commands

### 5.1 UDS\_GW::CONFIG\_MODULE (1)

With this command, parameters of uds\_gw module can be adjusted.

#### Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5
len	ecu	cmd	bs		cks
5	xx	1	MSB	LSB	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>bs</b>	maximum used buffer size for the transfer data commands, 10..4090 bytes
<b>cks</b>	checksum of telegram

#### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

### Remarks

- Normally, the maximum buffer size is provided by the counterpart with the response of the *request download* command. However, sometimes the software of counterpart reports a larger buffer size then can be used. The CONFIG\_MODULE command limits this value to an appropriable one.

## 5.2 UDS\_GW::READ\_VERSION (2)

This command reports about the module version information.

### Command

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	2	

**len** length of telegram  
**ecu** target address  
**cmd** command code  
**cks** checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3	...	byte 18	byte 19
len	ecu	status	ver 1	...	ver 16	cks
19	xx					

**len** length of telegram  
**ecu** source address  
**status** result status  
**ver 1..16** version string  
**cks** checksum of telegram

### Remarks

- As version string `uds_gwVVx.yy` should be reported.

### 5.3 UDS\_GW::CONFIG\_FASTFLASH (3)

With this command, parameters concerning FASTFLASH implementation of this module can be adjusted.

#### Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
len	ecu	cmd	cfg	scan	psize		cks
7	xx	3			MSB	LSB	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>cfg</b>	select suppression modes and file formats: <ul style="list-style-type: none"> <li>• bits 0..1: 0: no suppression, 1: FF suppression, 2: 00 suppression</li> <li>• bits 2..3: 0: binary file, 1: SRECORD file, 2: INTEL-HEX file</li> <li>• bit 7: 0: normal suppression, 1: autofill instead of suppression</li> </ul>
<b>scan</b>	second CAN (for FASTFLASH DUAL), 0: off (default), 1..4: used second CAN
<b>psize</b>	size of one data portion witch is used by FASTFLASH, max. 0x1000, default: 0x0100
<b>cks</b>	checksum of telegram

#### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

#### Remarks

- The CONFIG\_UNICOM(1) command which sets interface slots to MODULE turns off second CAN. It must be (re-)enabled by CONFIG\_FASTFLASH.

- Second CAN must differ to the primary one which is defined by the interface slot which is set to MODULE.
- Refer to Chapter FASTFLASH (chapter 4 on page 8) and command FASTFLASH\_UDS(14) (chapter 5.7 on page 19).

## 5.4 UDS\_GW::CONF\_DOWNLOAD (4)

This command configures some download parameters such as ALF, format and algorithm.

### Command, Form 1 (without setting internal timeout)

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6
len	ecu	cmd	ALF	format	flags	cks
6	xx	4				

### Command, Form 2 (with setting internal timeout)

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
len	ecu	cmd	ALF	format	flags	int_to	cks
7	xx	4					

<b>len</b>	length of telegram								
<b>ecu</b>	target address								
<b>cmd</b>	command code								
<b>ALF</b>	second byte of ALF ID: high nibble: number of bytes of length parameter in request download command (1..4) low nibble: number of bytes of address parameter in request download command (1..4)								
<b>format</b>	format parameter in request download command								
<b>flags</b>	bit coded control flags for DOWNLOAD: <table> <tr> <td><b>bit 0 = 0</b></td><td>(default) parallel send/receive and read from file. No repetition in case of error.</td></tr> <tr> <td><b>bit 0 = 1</b></td><td>serial send/receive and read from file. Up to 10 tries to send the transfer data command in case of error.</td></tr> <tr> <td><b>Bit 1 = 0</b></td><td>(default) size parameter in command DOWNLOAD_EX (23) is checked for validity (less or equal to filesize)</td></tr> <tr> <td><b>Bit 1 = 1</b></td><td>size parameter in command DOWNLOAD_EX (23) is placed into UDS-RequestDownload (0x34), but entire file is transferred (may be smaller than size). Use this option for downloading compressed data from ODX files.</td></tr> </table>	<b>bit 0 = 0</b>	(default) parallel send/receive and read from file. No repetition in case of error.	<b>bit 0 = 1</b>	serial send/receive and read from file. Up to 10 tries to send the transfer data command in case of error.	<b>Bit 1 = 0</b>	(default) size parameter in command DOWNLOAD_EX (23) is checked for validity (less or equal to filesize)	<b>Bit 1 = 1</b>	size parameter in command DOWNLOAD_EX (23) is placed into UDS-RequestDownload (0x34), but entire file is transferred (may be smaller than size). Use this option for downloading compressed data from ODX files.
<b>bit 0 = 0</b>	(default) parallel send/receive and read from file. No repetition in case of error.								
<b>bit 0 = 1</b>	serial send/receive and read from file. Up to 10 tries to send the transfer data command in case of error.								
<b>Bit 1 = 0</b>	(default) size parameter in command DOWNLOAD_EX (23) is checked for validity (less or equal to filesize)								
<b>Bit 1 = 1</b>	size parameter in command DOWNLOAD_EX (23) is placed into UDS-RequestDownload (0x34), but entire file is transferred (may be smaller than size). Use this option for downloading compressed data from ODX files.								



**int\_to** configures the internal command timeout for the UDS commands of download sequence.  
 0 (default): the timeout time which has been configured with the CONFIG\_UNICOM(1) command is used  
 else: *int\_to* is used (timeout time in seconds). Up to 255 seconds can be configured this way.

**cks** checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

**len** length of telegram  
**ecu** source address  
**status** result status  
**cks** checksum of telegram

### Remarks

- This command can be used if these parameters are known, e.g. from an ODX file.

## 5.5 UDS\_GW::SECURITY\_ALGORITHM (6)

This command performs project specific seed-key-calculations.

### Command

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	algo	
8	xx	6		

	byte 4	byte 5	byte 6	byte 7	byte 8
	seed				cks

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>algo</b>	algorithm selection (see project documentation)
<b>seed</b>	the 4 byte seed
<b>cks</b>	checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7
len	ecu	status	key				cks
7	xx						

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>key</b>	the 4 byte key
<b>cks</b>	checksum of telegram

## 5.6 UDS\_GW::CONFIG\_TIMEOUTS (7)

This command adjusts granularity of *to* parameter of GATEWAY(99) command (chapter 5.13 on page 27) and the value of *Protocol Timeout* (e.g. maximum time between two Response Pending telegrams)

### Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5
len	ecu	cmd	uto	pto	cks
5	xx	7			

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>uto</b>	0: units of 100 msec for <i>to</i> (default) 1: units of 1000 msec for <i>to</i>
<b>pto</b>	protocol timeout in units to 100 msec (default: 5sec)
<b>cks</b>	checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

### Remarks

- *pto* must not be 0
- With *uto* set to 1, timeout time of GATEWAY(99) command can be adjusted up to 255 sec.
- The real used value for protocol timeout is the minimum of *to* (specified by GATEWAY command) and *pto*.

## 5.7 UDS\_GW::FASTFLASH\_UDS (14)

This command is an alternate way for triggering FASTFLASH as described in the FASTFLASH chapter (ref. chapter 4 on page 8) with different command telegram format.

### Command

byte 0	byte 1	byte 2	byte 3	...	byte 6	
len	ecu	cmd	start			
N	xx	14	MSB	...	LSB	

byte 7	...	byte 10	byte 11	...	byte 14	
size			offset			
MSB	...	LSB	MSB	...	LSB	

byte 15	...	byte N-2	byte N-1	byte N
file			eos	cks
			0	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>start</b>	start address in flash
<b>size</b>	size of flash area
<b>offset</b>	start offset in file
<b>file</b>	data file, BINARY format, must reside in UNICOM's storage medium
<b>eos</b>	end of string, must be 0
<b>cks</b>	checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

### **Remarks**

- The command does exactly the same as X\_FASTFLASH(14) of the UCBASE software. The only difference is that it doesn't use "end address" but "size". In some cases it is more easily for the test PC to generate such a command.

## 5.8 UDS\_GW::DOWNLOAD (22)

With this command, a typical UDS download sequence can be automated:

- Service "Request Download"
- multiple Services "Transfer Data"
- Service "Request Transfer Exit"

### Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	
len	ecu	cmd	start addr				
N=8+n	xx	22	MSB			LSB	

byte 7	...	byte N-2	byte N-1	byte N
file 1	...	file n	EOS	cks
	...		0	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>start addr</b>	destination start address of code
<b>file</b>	name of file that contains download data (binary format)
<b>EOS</b>	End-Of-String, 0
<b>cks</b>	checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

## 5.9 UDS\_GW::DOWNLOAD\_EX (23)

Same as DOWNLOAD(22) command but with adjustable file size (ref. chapter 5.8 on page 21).

### Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	
len	ecu	cmd	start addr				
N=12+n	xx	23	MSB			LSB	

byte 7	byte 8	byte 9	byte 10	
size				
MSB			LSB	

byte 11	...	byte N-2	byte N-1	byte N
file 1	...	file n	EOS	cks
	...		0	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>start addr</b>	destination start address of code
<b>file</b>	name of file that contains download data (binary format)
<b>size</b>	uncompressed file size
<b>EOS</b>	End-Of-String, 0
<b>cks</b>	checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

## 5.10 UDS\_GW::DOWNLOAD\_EXX (24)

Same as DOWNLOAD(22) (ref. chapter 5.8 on page 21), but with adjustable start and end address and offset in file.

### Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	
len	ecu	cmd	start addr				
N=16+n	xx	24	MSB			LSB	

byte 7	byte 8	byte 9	byte 10	byte 11	byte 12	byte 13	byte 14	
end addr				file offset				
MSB			LSB	MSB			LSB	

byte 15	...	byte N-2	byte N-1	byte N
file 1	...	file n	EOS	cks
	...		0	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>start addr</b>	destination start address of code
<b>end addr</b>	destination end address of code
<b>file offset</b>	offset in source file in bytes
<b>cks</b>	checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

### Remarks

- *start addr* must be less or equal then *end addr*.
- $(\text{end addr} - \text{start addr} + 1) + \text{offset}$  must not exceed size of file.



## 5.11 UDS\_GW::DOWNLOAD\_EXXX (25)

Yet another variant of DOWNLOAD(22), here with adjustable start address, area size and offset in file.

### Command

byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	
len	ecu	cmd	start addr				
N=16+n	xx	25	MSB			LSB	

byte 7	byte 8	byte 9	byte 10	byte 11	byte 12	byte 13	byte 14	
size				file offset				
MSB			LSB	MSB			LSB	

byte 15	...	byte N-2	byte N-1	byte N
file 1	...	file n	EOS	cks
	...		0	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>start addr</b>	destination start address of code
<b>size</b>	size of download data
<b>file offset</b>	offset in source file in bytes
<b>cks</b>	checksum of telegram

### Response

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>cks</b>	checksum of telegram

### Remarks

- *size + offset* must not exceed size of file.

## 5.12 UDS\_GW::TESTER\_PRESENT (96)

With this command, a tester present message can be enabled and configured

### Command form 1: enable/disable

byte 0	byte 1	byte 2	byte 3	byte 4	
len	ecu	cmd	mode	period	
N=8	xx	96	0,1,2		

	byte 5	byte 6	byte 7	byte 8
	to	message		cks

### Command form 2: get status

byte 0	byte 1	byte 2	byte 3
len	ecu	cmd	cks
3	xx	96	

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>mode</b>	mode of operation, see remarks.
<b>period</b>	the period with which the message is to be send as multiple of 50 ms
<b>to</b>	the timeout between the message and the answer, as multiple of 5 ms
<b>message</b>	2 byte content of the tester present message
<b>cks</b>	checksum of telegram

### Response form 1: enable/disable

byte 0	byte 1	byte 2	byte 3
len	ecu	status	cks
3	xx		

### Response form 2: get status

byte 0	byte 1	byte 2	byte 3	byte 4
len	ecu	status	tpstat	cks
4	xx			

<b>len</b>	length of telegram
<b>ecu</b>	source address

<b>status</b>	result status
<b>tpstat</b>	status of tester present handshake
<b>cks</b>	checksum of telegram

**Remarks**

- *mode* specifies time control of tester present message:
  - 0: tester present message off (default)
  - 1: tester present on, the message will be sent in a fix period of time, even if there are other UDS commands, sent by GATEWAY(99) command.
  - 2: tester present on, every GATEWAY(99) command delays sending of next tester present message by the specified period time.
- tester present messages will not be sent while executing DOWNLOAD or FASTFLASH.
- with the *off* form, even all following parameters must be specified, but *period* and *to* are dummy.
- *message* may be *0x3E 0x00* (forces response from target device) or *0x3E 0x80* (doesn't force response).
- *tpstat* reports 0 if tester present is not enabled so far. Otherwise it reports statuses as command GATEWAY(99), or, if wrong response received, UDS\_COMMAND\_ERROR(0xE1).

### 5.13 UDS\_GW::GATEWAY (99)

With help of this command, UDS telegrams can be sent and received.

#### Command

byte 0	byte 1	byte 2	byte 3	
len	ecu	cmd	to	
N	xx	99		

	byte 4	byte 5	...	byte N-1	byte N
	uds_cmd	data	...	data	cks
			...		

<b>len</b>	length of telegram
<b>ecu</b>	target address
<b>cmd</b>	command code
<b>to</b>	timeout for receiving the response telegram, in units to 100 msec or 1000 msec depending on what is adjusted by the CONFIG_TIMEOUTS(7) command (chapter 5.6 on page 18). if to is set to 0, no response telegram is expected ("send only mode").
<b>uds_cmd,data</b>	the uds telegram that is to send. If not specified, no telegram is sent, but, if to is greater then 0, a response telegram from target is expected ("receive only mode").
<b>cks</b>	checksum of telegram

#### Response

byte 0	byte 1	byte 2	byte 3	
len	ecu	status	rec_status	
N	xx			

	byte 4	byte 5	...	byte N-1	byte N
	uds_stat	data	...	data	cks
			...		

<b>len</b>	length of telegram
<b>ecu</b>	source address
<b>status</b>	result status
<b>rec_status</b>	receive status (0xA0: NO ERROR). Reported if a response telegram was expected only.
<b>uds_stat, data</b>	received uds response telegram, if any.
<b>cks</b>	checksum of telegram

### Remarks

- If STP protocol is active, UDS telegrams with up to 251 bytes can be transferred with one command telegram.
- If XSTP protocol is active, UDS telegrams with up to 4091 bytes can be transferred with one command telegram.
- The CAN controller for UDS communication can be selected with the ECU number of command telegram. The ECU number addresses the interface slot as described in `ucbase.pdf`, chapter 2.3.2 "ECU number Address Scheme". Each interface slot that is configured for *MODULE* interface is associated with one CAN controller. Per default, the corresponding CAN controller is slot number +1, but it can be re-mapped by the `CONFIG_INTERFACE` command (s. chapter 3.2 on page 7).
- The modes "send only" and "receive only" can be used to communicate with up to 4 target devices in parallel:
  - Sending an UDS telegram to all the connected targets using "send only" mode. The targets can be selected by the ECU number of command telegram as described above.
  - Target devices execute the command in parallel.
  - Receiving the response telegram of the targets using the "receive only" mode.

## 5.14 UDS\_GW::ErrorCodes

The following table describes possible error codes reported by the *status* of the response telegrams, and their meanings.

Error	Code	Description
NO_ERROR	0xA0	No error occurred
PARAMETER_ERROR	0xB0	Wrong parameter in command telegram
LENGTH_ERROR	0xB3	Wrong command telegram length
ECU_LENGTH_ERROR	0xC3	Response telegram from target is too long
ECU_TIMEOUT_ERROR	0xC5	No response telegram received within specified timeout time
CAN_SEQUENCE_ERROR	0xC9	Wrong frame sequence
CAN_FORMAT_ERROR	0xCA	Invalid frame received
CAN_TIMEOUT_ERROR	0xCD	CAN timeout while sending or receiving
UDS_FORMAT_ERROR	0xE0	Bad formatter (ALF) received
UDS_COMMAND_ERROR	0xE1	Wrong response code received, or negative response
UNKNOWN_SECURITY_ERROR	0xE9	Wrong security algorithm code specified
UNKNOWN_COMMAND_ERROR	0xFF	Command code not supported by the module