DCSAP - test scenarios EN

Document revision: 8, export date: Dec 19, 2020

Document in Polish: Rewizja dokumentu: 10, data wygenerowania: gru 19, 2020

Table of contents

1. The aim of the document 5
2. Description of the test environment 5
3. Protocol and COSEM model 6

DCSAP1: Connecting to the DCU 6
DCSAP2: Starting the session 6
DCSAP3: Maintaining and closing session 6
DCSAP4: Opening Parallel Sessions 7
DCSAP5: Retrieving an empty meter list 7
DCSAP6: Retrieving non-empty meter list 8
DCSAP7: Start of meter profile download 8
DCSAP8: Session Independence verification (1) 9
DCSAP9: Session Independence verification (2) 10
DCSAP10: Session Independence verification (3) 11
DCSAP11: Communication with meters - Remote change of the relay state 11
DCSAP12: Verification of asynchronous command execution (1) 12
DCSAP13: Verification of asynchronous command execution (2) 12
DCSAP14: Transmission of DLMS commands for the meters 13
DCSAP15: Handling error code EUNKNOWN - unknown device ID 13
DCSAP16: Error code ETIMEOUT handling - response timeout 14
DCSAP17: Forwarding DLMS errors 14
DCSAP18: Verification of the correct operation of the DCU debug parameters object 15
DCSAP19: Availability of global DCU objects 15
DCSAP20: Verification of the correct operation of the meter list of the DCU (1) 16
DCSAP21: Verification of the correct operation of the meter list of the DCU (2) 16
DCSAP22: Verification of the correct operation of the meter list of the DCU (3) 17
DCSAP23: Basic information about the DCU 18
DCSAP24: Object of information about the DCU, restart() method 18
DCSAP25: NTP Server List Object (1) 19
DCSAP26: NTP Server List Object (2) 20
DCSAP27: NTP Server List Object (3) 20

DCSAP28: Checking the availability of the DCU's session objects 21
DCSAP29: Checking the availability of meter objects realised by the DCU 22
DCSAP30: Object Meter information (1) 22
DCSAP31: Object Meter information (2) 23
DCSAP32: Object Meter information (3) 24
DCSAP33: Object Meter information (4) 25
DCSAP34: Communication with meters - changing configuration of meters 25
DCSAP35: Remotely enable / disable DCU's interfaces 37
DCSAP36: Network interface configuration - WAN (1) 38
DCSAP37: Network interface configuration - WAN (2) 39
DCSAP38: Network interface configuration - LAN (1) 39
DCSAP40: Support for time stamps in DLMS queries 40
DCSAP41: DLMS query handling 41
DCSAP42: Error code EINVALID handling - invalid DLMS query format 42
DCSAP50: DCSAP SSL support (1) 42
DCSAP51: DCSAP SSL support (2) 43
DCSAP52: DCSAP SSL support (3) 43
DCSAP53: DCSAP SSL support (4) 44
DCSAP54: Effective DCSAP SSL communication 44
DCSAP61: Meters's LLS password configuration 45
DCSAP62: HLS authentication - MGMT Association 46
DCSAP63: HLS authentication - FW Update Association 47
DCSAP64: DLMS packet encryption and signing mechanism - MGMT association 48
DCSAP65: DLMS packet encryption and signing mechanism - FW Update association 49
DCSAP66: DLMS packet encryption mechanism - MGMT association 51
DCSAP67: DLMS packet encryption mechanism - FW Update association 52
DCSAP68: DLMS packet signing mechanism - MGMT association 53
DCSAP69: DLMS packet signing mechanism - FW Update association 54
DCSAP80: FW update of meters in broadcast mode - unencrypted 56
DCSAP81: FW update of meters in broadcast mode - encrypted 57
DCSAP82: FW update of meters in unicast mode - unencrypted 58
DCSAP83: FW update of meters in unicast mode - encrypted 59
DCSAP84: Updating FW of meters in unicast mode - encrypted and signed 59
DCSAP85: FW update of meters - conditional (1) 60
DCSAP86: FW update of meters - conditional (2) 61
DCSAP87: FW update of meters - automatic triggering of updates (1) 62

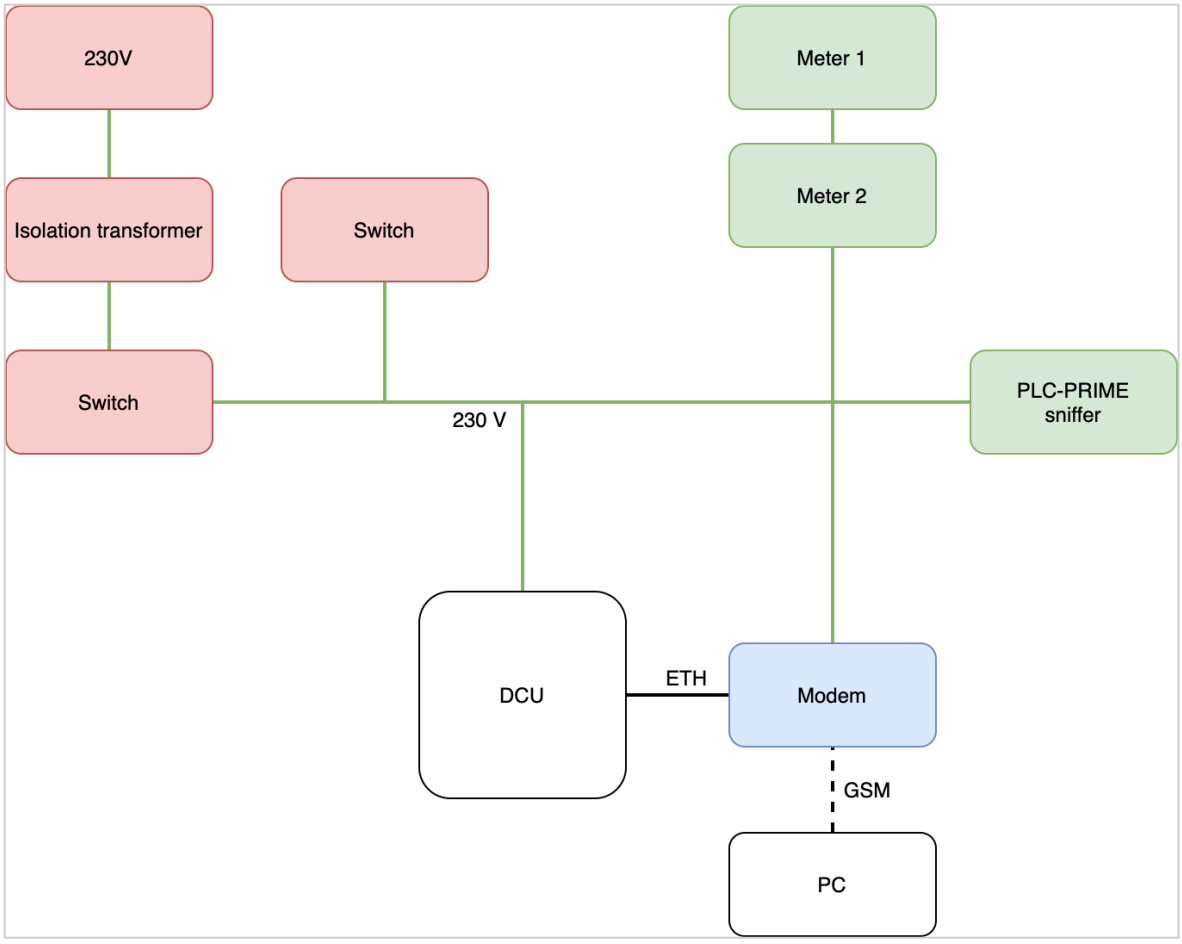
DCSAP88: FW update of meters - automatic triggering of updates (2) 63
DCSAP89: FW update of meters in unicast mode - the meter is temporarily unavailable 64
DCSAP90: FW update of meters in broadcast mode - meters temporarily unavailable 64
DCSAP91: FW update of meters - pausing of the FW upgrade mechanism 65
DCSAP95: Send Emergency Commands in Broadcast Mode (1) 67
DCSAP96: Sending Emergency Commands in Broadcast Mode (2) 67
DCSAP97: Successful sending of emergency commands in broadcast mode 68
DCSAP99: Communication with the ISD module 70
DCSAP100: PRIME 1.4 modem configuration - MAC backward compatibility 70
DCSAP101: PRIME 1.4 modem configuration - one communication channel 71
DCSAP102: PRIME 1.4 modem configuration - multiple communication channels 71
4. Event list 72
ZDA01: Data concentrator event list 72
ZDA02: First registration of the meter in the DCU 73
ZDA03: 'connection' event registration 73
ZDA04: 'login' event registration 74
ZDA05: 'tamper' event registration 75
ZDA06: Event list filtering 76
ZDA10: DCU asynchronous event reporting (1) 77
ZDA11: DCU asynchronous event reporting (2) 77
ZDA12: Asynchronous meter events reporting (1) 78
ZDA13: Asynchronous meter events reporting (2) 78
5. Firmware update 79
UPG01: DCU HTTPS update - invalid certificate 79
UPG02: DCU HTTPS update - invalid url 80
UPG03: DCU HTTPS update - incorrect firmware file 80
UPG04: DCU HTTPS update - firmware file mismatch 81
UPG05: DCU firmware update - update aborted 81
UPG06: DCU HTTPS software update - correct update process 82
UPG07: DCU HTTP update - invalid url 83
UPG08: DCU HTTP update - correct update process 83

1. The aim of the document

This document contains a minimum set of functional test scenarios that can be carried out in order to verify the correct operation of the DCSAP protocol and the compliance of its implementation with version 3.0.1 of the DCSAP specification.

2. Description of the test environment

The test cases described in this document should be run in the test environment outlined below. The figure shows the installation diagram for Firmware testing.



The test installation consists of the following elements:

1. two single-phase municipal electricity meters, communicating in accordance with the PRIME specification in version 1.4 (with backward compatibility mode), using the DLMS protocol, with the COSEM object model,
2. 3GPP / CDMA modem for the test of communication of the DCU with an extensive ICT network,
3. Data Concentrator Unit (DCU)

4. A PC acting as a DCSAP client
5. PLC-PRIME sniffer.

3. Protocol and COSEM model

DCSAP1: Connecting to the DCU

Description:
Connecting to the DCU

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network.

Steps:	Expected results:
1. Connect the DCSAP client to the DCU via TCP / IP on port 16000. 2. Send an empty DCSAP ('ping') message at least once every 5 minutes to keep the session alive.	1. The connection is established. 2. The DCU responds to the 'ping' commands with the same messages. 3. The connection is not broken by the DCU.

DCSAP2: Starting the session

Description:

The purpose of the test is to verify the correctness of communication between the DCSAP client and the DCU, test is done using a single TCP / IP connection that starts the sessions.

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network.

Steps:	Expected results:
1. Open a communication session with the DCU. 2. Send the command to DCU (device_id = 0) Get-Request-Normal for object 40002 / 0-100: 0.0.5 * 255/3 (network statistics - number of currently open DCSAP sessions). 3. Close the session.	1. The session has been set up. 2. DCU sends a Get-Response-Normal response with the DLMS data. In response, the number of currently open sessions is as expected. 3. The session has been closed.

DCSAP3: Maintaining and closing session

Description:

The purpose of the test is to verify if the the DCU connection and session maintenance when receiving messages and closing it as a result of inactivity on the part of the DCSAP client is correct.

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network.

Steps:	Expected results:
1. Open a communication session with the DCU. 2. Send a blank message (ping) to the DCU every 5 minutes for at least 15 minutes. 3. Wait 15 minutes without sending any inquiries to the DCU.	1. The session is set up. 2. The DCU sends back empty messages unchanged, the session is held. 3. After 10-15 minutes, DCU closes the TCP connection (the session is closed).

DCSAP4: Opening Parallel Sessions

Description:

The purpose of the test is to verify if the acquisition system can establish many sessions with a single DCU.

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network.

Steps:	Expected results:
1. Open a communication session with the DCU. 2. Open a second, parallel session with the DCU. 3. Wait a minute and close the first session.	1. The session has been set up. 2. The second session has been set up - two sessions are open. 3. The first session has closed. The second session remains open.

DCSAP5: Retrieving an empty meter list

Description:

The purpose of the test is to verify the correctness of downloading the empty meter list.

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network,
3. PLC network without any PRIME meters.

Steps:	Expected results:
1. Connect with the DCSAP client to the DCU.	1. The list of meters contains one item: balancing meter. 2. The balancing meter is visible ('visible' field is 'true').

2. Download the meter list using the DCSAP protocol. 3. Download the extended meter list with the DCSAP protocol.	3. The list may additionally contain other meters, if the DCU is not brand new, but these meters are not visible in the network. 4. All data on the extended list of meters are consistent with the expected (medium, meter configuration, range of profile data)

DCSAP6: Retrieving non-empty meter list

Description:

The purpose of the test is to verify the correctness of downloading a non-empty list of meters.

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network,
3. PLC network with connected PRIME meters (PLC meters from different manufacturers).

Steps:	Expected results:
1. Connect with the DCSAP client to the DCU. 2. Download the meter list using the DCSAP protocol. 3. Connect some of the meters to the PLC network. 4. Wait 2 minutes. 5. Download the meter list using the DCSAP protocol. 6. Download an extended list of meters using the DCSAP protocol.	1. The first list of meters contains only one visible meter: the balancing meter. 2. In the second meter list, all connected PLC meters have 'visible' set to 'true' and unconnected meters have 'visible' set to 'false'. 3. All data on the extended meter list are consistent with the expected ones (MAC address, PLC topological address, medium, meter configuration, range of profile data)

DCSAP7: Start of meter profile download

Description:

The purpose of the test is to verify the correctness of communication with meters and profiles acquisition.

Test requirements:

1. DCSAP client,
2. DCU connected to the Ethernet network - NTP server available for the DCU,
3. PLC network with connected PRIME meters.
4. The DCU does not yet collect the hourly profile for the connected meters.

Steps:	Expected results:

1. Connect with the DCSAP client to the DCU. 2. Download the list of meters using the DCSAP protocol. 3. Wait until the meter is visible on the list. 4. Get hourly profile configuration (attribute 3) from the meter. 5. Wait up to 10 minutes for the DCU to collect the profile. 6. Get hourly profile configuration (attribute 3) from the meter. 7. Download hourly profile entries (attribute 2) for -7 days to date. 8. Download event profile entries. 9. Wait 6 hours. 10. Download profile entries from all connected meters. 11. Download an extended list of meters.	1. The first download of the profile configuration will return the error DCSAP - 11 (EASKLATER) 2. The second download of the profile configuration will return the correct answer. 3. Retrieving profile entries will return valid entries, the entry table returned will be limited to the earliest 64 entries. 4. Profile entries retrieved in step 10 should be current, i.e. latched by meters in the last 6 hours. 5. All data on the extended list of meters are consistent with the expected ones (MAC address, PLC topological address, medium, meter configuration, range of profile data)

DCSAP8: Session Independence verification (1)

Description:

The purpose of the test is to verify 'session independence' setting session parameters to one session should not affect other sessions - both ongoing and future ones.

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network.

Steps:	Expected results:
1. Open a communication session with the DCU. 2. Open a second network session with the DCU. 3. In the first session, send the command Get-Request-Normal 40002 / 0-100: 0.0.5.255/3 to the DCU (network statistics - number of currently open DCSAP sessions) 4. In the second session, send a command to the DCU (device_id = 0) Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 (for example - reading the list of meters).	1. The session opens. 2. A second session opens. 3. In the first session, a Get-Response-Normal response was received with the appropriate number of currently open DCSAP sessions (2). The second session remains active. No data was received in the second session. 4. A Get-Response-Normal response was received in the second session. The first session remains open, no messages have been received in it.

5. Close the second session. 6. In the first session, send the command Get-Request-Normal 40002 / 0-100: 0.0.5.255/3 to the DCU (network statistics - number of currently open DCSAP sessions)	5. The second session has closed. The first session remains active with no messages received in it. 6. In the first session, a Get-Response-Normal response was received with the appropriate number of currently open DCSAP sessions (1).

DCSAP9: Session Independence verification (2)

Description:

The purpose of the test is to verify 'session independence' setting session parameters to one session should not affect other sessions - both ongoing and future ones.

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network,
3. At least one meter is connected,
4. Automatic download of profiles is enabled for the connected meter.

Steps:	Expected results:

1. Open a communication session with the DCU.
2. Open a second communication session with the DCU.
3. In the second session, send the command Set-Request-Normal 1 / 0-100: 32.0.0 / 2 2 (DCU meter data caching enable) to DCU with the value of the Boolean type = false.
4. In both sessions, send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 (read the list of meters) to the DCU. note the device_id of the selected meter as L1.
5. In both sessions send the profile 1 read command to the L1 meter simultaneously:
`Get-Request-Normal 7/1-0:99.1.0*255/2,`
`z parametrami Selective-Access:`
`range_descriptor,`
`restricting_object = Clock (8/0-0:1.0.0*255/2),`
`from_value = T1,  -> now - 6h`
`to_value = T0,  -> now - 1h`
`Selected_values = empty table.`
6. Disconnect the L1 meter from the network and repeat step 5.
	1. The session opens.
	2. A second session opens.
	3. In the second session, a Set-Response-Normal response was received with code 0 (Success). The first session remains active.
	4. Get-Response-Normal replies with a list of meters were received in both sessions, both replies are identical.
	5. In the first session, a Get-Response-Normal response was received with profile 1 data within the given range; In the second session, a Get-Response-Normal response was received with Profile 1 data ranging from T1-T2, where T1 <= T2 <= T0 (or an empty table). The data from both sessions is consistent (they do not have to be identical if the time format correcting in the profile data is enabled for a given meter)
	6. Profile 1 data (per step 5 criteria) was received in the first session, no response was received in the second session. After about 10 minutes the ETIMEOUT answer will be received.

DCSAP10: Session Independence verification (3)

Description:

The purpose of the test is to verify 'session independence' setting session parameters to one session should not affect other sessions - both ongoing and future ones.

Test requirements:

1. Klient protokołu DCSAP z otwartą sesją do koncentratora z wyłączonym *cache*,
2. Niepodłączony co najmniej jeden licznik - L1.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters); 2. Note the device_id of the selected unconnected meter as L1. 3. Send the command to change the relay state for the L1 meter via the DCSAP protocol. 4. Close the DCSAP session with the DCU. 5. Connect the L1 meter. 6. Check the condition of the relay on the L1 meter's LCD display.	1. Get-Response-Normal response with a list of meters received. 2. No response was received to the command to change the meter reading. 3. After connecting the meter, the relay status remained unchanged

DCSAP11: Communication with meters - Remote change of the relay state

Description:

The purpose of the test is to verify the correctness of communication with the meters, turning the relay on / off.

Test requirements:

1. DCSAP client,
2. A DCU connected to the Ethernet network,
3. At least one PLC meter connected to the DCU - L1.

Steps:	Expected results:
1. Connect with the DCSAP client to the DCU. 2. Send the command Action-Request-Normal 70 / 0-0: 96.3.10 * 255/1 to the L1 meter with the parameter of type Integer with the value 0 (zero) 3. Check the status of its relay on the meter's display.	1. The session opens. 2. Received an Action-Response-Normal response with code Action-Result = 0 (success). 3. The meter has disconnected the relay - the meter display shows the current status of the relay (check the meter documentation). 4. The operation lasted no longer than 1 minute from the moment of sending the command to change the relay status until the information on the relay disconnection appeared on the meter display.

DCSAP12: Verification of asynchronous command execution (1)

Description:

The purpose of the test is to verify whether the unused operations are performed after connecting the meter.

Test requirements:

1. DCSAP client with an open session to a DCU with a disabled *cache* ,
2. At least one meter - L1 is not connected.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters); 2. Note the device_id of the selected unconnected meter as L1. 3. Send the command to change the relay state for the L1 meter via the DCSAP protocol. 4. Connect the L1 meter. 5. Check the condition of the relay on the LCD display of L1 meter.	1. Get-Response-Normal response with a list of meters received. 2. No response was received to the command to change the meter reading. 3. After connecting the meter, the relay status was changed. 4. Response received with status code: success.

DCSAP13: Verification of asynchronous command execution (2)

Description:

The purpose of the test is to verify that operations to different devices are performed asynchronously.

Test requirements:

1. DCSAP client with an open session to a DCU with a disabled *cache*,
2. At least two meters (L1, L2) are connected,
3. The L1 meter is visible and has latched data in profile 1 from time $t_1 = t_0$ - 12h to time t_0 (test start time),
4. The L2 meter is invisible (not available via the PLC).

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters); 2. Note the device_id of the selected meters (L1, L2) and the device_id that do not exist on the list (LX). 3. Send command to DCU the command Set-Request-Normal 1 / 0-100: 32.0.0.255/2 (DCU meter data caching enable) with the value of the Boolean type = false. 4. Do the following commands in order (for LX, L2, L1 meters) without waiting for responses, specifying a different message_id for each command: a. for LX, L2, L1 meters:	1. Get-Response-Normal response with a list of meters received. 2. Responses were received with message_id, device_id pairs matching the sent commands: a. Get-Response-Normal response was received for L1 with profile data. b. Received L2 response with DCSAP error (timeout or unavailable meter).

Get-Request-Normal 7/1-0:99.1.0*255/2, z parametrami Selective-Access: range_descriptor, restricting_object = Clock (8/0-0:1.0.0*255/2), from_value = T1, to_value = T0, Selected_values = empty table. b. to the DCU: Get-Request-Normal 40000/0-100:0.0.0*255/2.	c. Response received for L3 with error DCSAP = 1 (EUNKNOWN - meter unknown). d. Get-Response-Normal response received for DCU with a list of meters. 3. The DCU response was received before the L1 and L2 responses.

DCSAP14: Transmission of DLMS commands for the meters

Description:

The purpose of the test is to verify the DCU's ability to transmit DLMS commands to the meters.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. At least two meters (L1, L2) are connected,
3. Sniffer PLC-PRIME

Steps:	Expected results:
1. Connect the Sniffer PRIME to the PLC network. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters); *note the device_id of the selected meters (L1, L2).* 3. Send a command to the meters L1, L2: Get-Request-Normal 3 / 1-0: 1.8.0 * 255/2.	1. The sniffer receives BCN packets sent generated by the DCU. Positive registration of meters in the DCU can be traced. 2. Get-Response-Normal response with a list of meters received. 3. Get-Response-Normal responses with DLMS data were received in the DCU session: integers from L1, L2 meters; PLC / PRIME frame sniffer to L1, L2 meters was observed; In the frames received by the sniffer, the sent commands and the received responses were observed in accordance with those sent in the session with the DCU.

DCSAP15: Handling error code EUNKNOWN - unknown device ID

Description:

The purpose of the test is to verify whether the DCU will respond with the EUNKNOWN error code for the command addressed to the unknown device .

Test requirements:

DCSAP client with an open session to the DCU.

Steps:	Expected results:

1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters); *select and note any device_id = LX that is not in the list.* 2. Send a command to the LX meter: Get-Request-Normal: 8 / 0-0: 1.0.0 * 255/2 (clock).	1. Get-Response-Normal response with a list of meters received. 2. DCSAP error response with code -1 (EUNKNOWN) received.

DCSAP16: Error code ETIMEOUT handling - response timeout

Description:

The purpose of the test is to verify whether the device reports a timeout error.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. At least one L1 meter is connected,
3. Timeout for commands in the DCU set to the TTO time.

Steps:	**Expected results:**
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters); *note the device_id of the L1 meter.* 2. Disconnect the power supply from the L1 meter. 3. Send a command to L1 meter: Get-Request-Normal 8 / 0-0: 1.0.0 * 255/2 (clock), wait for more than TTO, no more than TTO + 1 minute.	1. Get-Response-Normal response with a list of meters was received. The line of the list of meters for the L1 meter in the present field has a value other than zero. 2. Meter disconnected from the network. 3. DCSAP error response with code -5 (ETIMEOUT) received.

DCSAP17: Forwarding DLMS errors

Description:

The test aims to verify the correctness of reporting errors as DLMS codes received for for proper DCSAP queries.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. At least one L1 meter is connected,

Steps:	**Expected results:**
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters); select and note the device_id of the L1 meter 2. Send the command Get-Request-Normal 99 / 0-0: 1.0.0 * 255/2 to the L1 meter.	1. Get-Response-Normal response with a list of meters received. 2. Get-Response-Normal response with code 9 (class-inconsistent) or 4 (object-undefined) received.

3. Send the command Get-Request-Normal 3 / 1-0: 1.8.0 * 255/2 to the L1 meter. Note the data type. 4. Send the command Set-Request-Normal 55 / 1-0.1.8.0 * 255/2 = Double-Long-Unsigned to the L1 meter: 0 NOTE: Replace the Double-Long-Unsigned type with the one noted in step 3. 5. Send the command Set-Request-Normal 3 / 1-0: 1.8.0 * 255/2 = Double-Long-Unsigned: 0 to the L1 meter NOTE: Replace the Double-Long-Unsigned type with the one noted in step 3.	3. Get-Response-Normal response received with data: number. 4. Set-Response-Normal response with code 9 (class-inconsistent) or 4 (object-undefined) was received. 5. Set-Response-Normal response with code 3 (read-write-denied) received. The received types of DLMS errors depend on the implementation of the meter DLMS / COSEM model.

DCSAP18: Verification of the correct operation of the DCU debug parameters object

Description:

The purpose of the test is to verify the correct operation of the DCU debug parameters object.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. Telnet / ssh client with an open session to the DCU,
3. At least one L1 meter is connected,

Steps:	Expected results:
1. Send the command Set-Request-Normal 1 / 0-100: 0.130.0.255/2 = Boolean: true to the DCU. 2. Perform the DCSAP14 test. Watch the system log with telnet / ssh.	1. Set-Response-Normal response with *success* code received . 2. In the system log, one can observe sending / receiving packets to individual meters written in hexadecimal format.

DCSAP19: Availability of global DCU objects

Description:

The purpose of the test is to check whether the global objects of the DCU are implemented.

Test requirements:

1. DCSAP protocol client with an open session to the DCU.

Steps:	Expected results:
1. Send the Get-Request-Normal command to the DCU sequentially for all attributes of all objects defined in the COSEM model of the DCU.	1. At each step, a Get-Response-Normal response was received with DLMS data consistent with the expectations (e.g. in the case of DCU settings - consistent with the data on the webGUI). The DLMS type of the received data conforms to the COSEM DCU model.

DCSAP20: Verification of the correct operation of the meter list of the DCU (1)

Description:

The purpose of the test is to verify the correct operation of the DCU's meter list - verification of basic functionalities.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. No municipal meters connected,
3. At least one L1 meter ready for connection,

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU. *Make a note of the largest value of the last_change_seq_id = MAX_CH_ID field.* 2. Connect the L1 meter to the DCU via the PLC interface; wait TD = 2 minutes, according to the DCU's documentation. 3. Send commands Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 and Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5 to the DCU *Note the record values for the L1 meter.* 4. Disconnect the L1 meter from the PLC network; wait at least 5 minutes. 5. Send the command to DCU Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 and Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5	1. Get-Response-Normal response received with DLMS data: meter table; All entries except for LB have the field present = *false* . 2. The L1 meter is registered with the DCU. 3. Get-Response-Normal response received with DLMS data: meter table (and extended meter list); All entries except for LB and L1 have the field present = *false* ; The entry for L1 includes: - last_change_seq_id> MAX_CH_ID - last_change_time = meter connection time - present = *true* . 4. L1 meter disconnected. 5. Get-Response-Normal response received with DLMS data: meter table (and extended meter list); All entries, except for LB, have the field present = *false* ; The entry for L1 includes: - last_change_seq_id = last_change_seq_id (from step 3) + 1 - last_change_time = meter disconnection moment - present = *false* - other fields identical to the values received in step 3

DCSAP21: Verification of the correct operation of the meter list of the DCU (2)

Description:

The purpose of the test is to verify the correct operation of the DCU's meter list. Limiting the amount of data transferred by selective selection - determination of the highest known register change number in the sequence.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. Completed test DCSAP20 (causing changes on the meter list).

Steps:	Expected results:

1. Send commandGet-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU *Note down any maximum last_change_seq_id = CH_ID.* 2. Send a command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU, access-selector 1 with a parameter of type Long64-Unsigned with the value CH_ID .	1. Get-Response-Normal Response Received with Data: meter List. The answer contains at least two entries. 2. Get-Response-Normal response received with data: meter list; The answer contains at least one entry; All values in the last_change_seq_id column are greater than CH_ID.

DCSAP22: Verification of the correct operation of the meter list of the DCU (3)

Description:

The purpose of the test is to verify the correct operation of the DCU's meter list. Limiting the amount of data transferred in the extended list of meters.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. Completed test DCSAP20 (causing changes on the list of meters).

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5 to the DCU *Note any non-maximum last_change_seq_id = CH_ID.* 2. Send a command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5 to the DCU, access-selector 1 with a `seq_id` Long64-Unsigned field of CH_ID. 3. Send a command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5 to the DCU, access-selector 1 with a `field_mask` double-long-unsigned field of 0xffffffff value. 4. Send a command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5 to the DCU, access-selector 1 with a `field_mask` double-long-unsigned field of 0x1f7b value. 5. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5 to the DCU, access-selector 1 with a `field_mask` double-long-unsigned field with a random value. 6. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5 to the DCU, access-selector 1 with a `max_cnt` double-long-unsigned field value of 2.	In each case, a Get-Response-Normal response was received with the data: extended list of meters. 1. The answer contains at least two entries. 2. The answer contains at least one entry; All values in the last_change_seq_id column are greater than CH_ID. 3. The response contains a field `field_mask` limited to the values defined in DCSAP 3 (0x3ffff). All returned values for the meters are as expected. 4. The response contains a field `field_mask` equal to the value in the query (0x1f7b). Only the requested values were returned for each meter. All returned values for the meters are as expected. 5. The response contains a field `field_mask` equal to the value in the query limited to the values defined in DCSAP 3 (0x3ffff). Only the requested values were returned for each meter. All returned values for the meters are as expected. 6. The answer contains a maximum of 2 entries. 7. For each numerator, the field values `prof_range` represent the status of the profile 1-0: 99.2.0.255 (daily) - in particular, the values `oldest/newest/capture-period` are consistent with the expectations and different than in (3).

7. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/5 to the DCU, access-selector 1 with `range_prof_obis` an octet-string [6] value of 0100630200FF .	

DCSAP23: Basic information about the DCU

Description:

The purpose of the test is to verify the correct operation of the objects containing information about the DCU.

Test requirements:

1. DCSAP client with an open session to the DCU.

Steps:	Expected results:
1. Send the command Get-Request-Normal 3 / 0-100: 0.100.0.255 / 2 to the DCU (DCU application statistics). 2. Send the command Get-Request-Normal 3 / 0-100: 0.99.0.255 / 2 to the DCU (DCU status flags). 3. Send the command Get-Request-Normal 40101 / 0-100: 0.0.1.255 / 2 to the DCU (DCU firmware / version). 4. Send the command Get-Request-Normal 40103 / 0-100: 0.0.2.255 / 4 to the DCU (DCU run information / curr_uptime_secs). 5. Send the command Get-Request-Normal 1 / 0-0: 96.1.0.255/2 to the DCU (ID of the device). 6. Send the command Get-Request-Normal 1 / 0-100: 128.0.0.255/2 to the DCU (ID of the device). 7. Send the command Get-Request-Normal 1/0-100:128.0.1.255/2 to the DCU (FW version of the device). 8. Use webGUI to verify the correctness of the data received in (1)	1. Get-Response-Normal response received with data meeting the criteria: `time` equal to the current UNIX TIME in GMT, `uptime` has a slight discrepancy with the data downloaded in (4), `version` contains the same data as the response to the query (3) and (7), `dev_id1, dev_id2` contains the same information about the response to the inquiries (5) and (6), `dev_id1, dev_id2, plc_mac` is consistent with the information on the device housing, `ntp_sync` consistent with the state of the last event of the *EV_TIME_VALIDATION* type ,`status_flags` contains the same data as the response to the inquiry (2). *Other inquiries end with the code success. The data is consistent with the data presented in the webGUI.*

DCSAP24: Object of information about the DCU, restart() method

Description:

The purpose of the test is to verify the DCU restart procedure by calling the restart() method of the information object of the DCU.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. The DCU update process is not active.

Steps:	Expected results:
1. Send the Get-Request-With-List command to the DCU for object 40103 / 0-100: 0.0.2 * 255 attributes 2,3. *Make a note of the received values = START_CNT, START_TM.* 2. Send command to the DCU the Action-Request-Normal 40103 / 0-100 command: 0.0.2 * 255/1 without parameters. 3. Restart the DCSAP connection (open connections until success, no longer than 5 minutes). 4. Send the Get-Request-Normal command to the DCU for object 40103 / 0-100: 0.0.2 * 255/2. 5. Send the command Get-Request-Normal: 40001 / 0-100: 0.0.3 * 255/2 to the DCU.	1. Get-Response-With-List response received with 2 parts with DLMS data: - Part 1: Double-Long-Unsigned - Part 2: Date-Time 2. Received an Action-Response-Normal response with Action-Result = 0 (succcess). DCSAP session is closed. DCU restarts. 3. DCSAP session compiled. 4. Get-Response-Normal response received with DLMS data type Double-Long-Unsigned, value = START_CNT + 1. 5. Get-Response-Normal response received with data: event list. There is an event on the list that meets the criteria: - device_id = 0, - time> START_TM, - reason = 1 (EV_RESTART);

DCSAP25: NTP Server List Object (1)

Description:

The purpose of the test is to verify the possibility of setting the list of NTP server addresses, enabling the use of the same time servers in the entire infrastructure.

Test requirements:

1. DCSAP protocol client with an open session to the DCU.

Steps:	Expected results:
1. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 0.0.4 * 255. Attributes 2-4. *Make a note of the list returned in Part 2.* 2. Send the command Set-Request-Normal 40100 / 0-100: 0.0.4 * 255/1 to the DCU with the value of the Array of Octet-String = modified list obtained in step 1. 3. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 0.0.4 * 255. Attributes 2-4.	1. Get-Response-With-List response received in 4 parts with DLMS data: - part 2 of the Array of Octet-String type, values = list of IP addresses (may be empty) - part 3 of the Double-Long-Unsigned type value = number of lines in part 2 - part 4 of type Double-Long-Unsigned value> = values in part 3 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. The list matches that sent in step 2. *Additionally, the compliance of the settings available by DCSAP with the settings available by the test and diagnostic software should be verified*

DCSAP26: NTP Server List Object (2)

Description:

The purpose of the test is to verify the possibility of setting the list of NTP server addresses, enabling the use of the same time servers in the entire infrastructure.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. The NTP server list is not empty.

Steps:	Expected results:
1. Send the command Get-Request-Normal: 40100 / 0-100: 0.0.4 * 255/2 to the DCU. 2. Send the command Set-Request-Normal 40100 / 0-100: 0.0.4 * 255/2 to the DCU with the value of the Array type, without lines (empty array). 3. Send the command Get-Request-Normal: 40100 / 0-100: 0.0.4 * 255/2 to the DCU. 4. Send the command Get-Request-Normal: 40100 / 0-100: 0.0.4 * 255/4 to the DCU. *Make a note of the value returned = MAX_N.* 5. Send the command Set-Request-Normal 40100 / 0-100: 0.0.4 * 255/2 to the DCU with a value of the Array, of Octet-String type containing valid IP addresses, number of lines> MAX_N. 6. Send the command Get-Request-Normal: 40100 / 0-100: 0.0.4 * 255/2 to the DCU. 7. Send the command Set-Request-Normal 40100 / 0-100: 0.0.4 * 255/2 to the DCU with the value of the Array type, of Octet-String containing the correct IP addresses, number of lines = MAX_N. 8. Send the command Get-Request-Normal: 40100 / 0-100: 0.0.4 * 255/2 to the DCU.	1. Get-Response-Normal response received with DLMS data: IP address list. 2. DLMS error 12 response received (type unmatched). 3. The Get-Response-Normal response was received with DLMS data: IP address list from (1). 4. A Get-Response-Normal response was received with Double-Long-Unsigned data with a value > 1. 5. DLMS error 12 response received (type unmatched). 6. The Get-Response-Normal response was received with DLMS data: IP address list from (1). 7. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 8. A Get-Response-Normal response was received with DLMS data identical to that sent in the Set-Request-Normal command in step 7. *Additionally, the compliance of the settings available by DCSAP with the settings available by the test and diagnostic software should be verified.*

DCSAP27: NTP Server List Object (3)

Description:

The purpose of the test is to verify the possibility of an immediate attempt to synchronise time with NTP servers.

Test requirements:

1. A DCU with an enabled NTP time synchronisation service,
2. DCSAP protocol client with an open session to the DCU,
3. Ethernet network configuration allowing for blocking access from DCU to NTP server on request

4. Blocked access to the NTP server from the beginning of the DCU's operation.

Steps:	Expected results:
1. Send the command to the DCU: Get-Request-Normal 3 / 0-100: 0.100.0.255 / 2 (DCU application statistics). 2. Unblock DCU's access to the NTP server. 3. Send the command to the DCU: Get-Request-Normal 3 / 0-100: 0.100.0.255 / 2 (DCU application statistics). 4. Send the Action-Request-Normal command to the DCU for object 40100 / 0-100: 0.0.4 * 255/1. *Wait 10-15 seconds.* 5. Send the command to the DCU: Get-Request-Normal 3 / 0-100: 0.100.0.255 / 2 (DCU application statistics). 6. Verify your observations against webGUI.	1. Get-Response-Normal response received with code *success*. The field `ntp_sync` has a value `false`. 2. (network access from DCU to NTP is possible) 3. Get-Response-Normal response received with code *success*. The field `ntp_sync` has a value `false`. 4. Action-Response-Normal response received with code *success.* 5. Get-Response-Normal response received with code *success*. The field `ntp_sync` is true. 6. The NTP synchronization status on the webGUI is as expected. The *EV_TIME_VALIDATION* event with the status 1 appeared in the DCU's event *log*.

DCSAP28: Checking the availability of the DCU's session objects

Description:

The purpose of the test is to check the availability of the DCU's session objects.

Test requirements:

1. DCSAP client with an open session to the DCU.

Steps:	Expected results:
In the next steps, send the Get-Request-Normal command to the DCU for the given objects: 1. 1/0-100:32.0.0*255/2, 2. 1/0-100:32.0.1*255/2, 3. 1/0-100:63.0.0*255/2, 4. 1/0-100:63.0.1*255/2, 5. 1/0-100:63.0.2*255/2, 6. 1/0-100:32.1.0*255/2, 7. 1/0-100:32.1.1*255/2,	A Get-Response-Normal response with DLMS data was received at each step. The received data types and data are consistent with the default values - according to the DCU's COSEM model.

8. 1/0-100:32.1.2*255/2.	

DCSAP29: Checking the availability of meter objects realised by the DCU

Description:

The purpose of the test is to check the availability of meter objects implemented by the DCU.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. L1 municipal meter available.

Steps:	Expected results:
Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). *Note the device_id of the L1 meter.* In the next steps send the Get-Request-Normal command to L1 for the given objects: 1. 40102/0-100:64.0.0*255/**, 2. 40101/0-100:64.0.1*255/**, 3. 40160/0-100:160.0.1*255/**, 4. 40199/0-100:65.0.2*255/**, 5. 40199/0-100:65.0.3*255/**, 6. 40199/0-100:65.0.4*255/**, 7. 40199/0-100:65.0.6*255/**, 8. 1/0-100:66.0.2*255/2, 9. 1/0-100:66.128.2*255/2, 10. 1/0-100:66.0.3*255/2, 11. 1/0-100:66.128.3*255/2, 12. 1/0-100:66.0.4*255/2, 13. 1/0-100:66.128.4*255/2, 14. 1/0-100:66.1.4*255/2, 15. 1/0-100:66.1.6*255/2. ** → *query sequentially for all the attributes according to the class definition*	Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. A Get-Response-Normal response with DLMS data was received at each step. The received data types and data are consistent with the default values - according to the DCU's COSEM model.

DCSAP30: Object Meter information (1)

Description:

The purpose of the test is to verify the implementation of the Meter information object for the meter.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. L1 municipal meter available.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). *Note the device_id of the L1 meter.* 2. Send the Get-Request-With-List command to L1 for object 40102 / 0-100: 64.0.0 * 255. Attributes 2-12.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Get-Response-With-List response received with 11 parts with DLMS data: - Part 2 of Double-Long-Unsigned type, - Part 3 of Data type (depending on the meter manufacturer), non-empty value, - Part 4 of Double-Long-Unsigned type, value 0 (unless communication with L1 has errors), - part 5 of the Octet-String type, value consistent with the Logical-Device-Name of the meter (and in the meter_name on the list of meters), - part 6 of the Double-Long-Unsigned type, value from the 1-16 range (1 when the meter does not support With-List queries), - part 7 of the Double-Long-Unsigned type, value depending on the type of the meter and FW version, - part 8 of the Double-Long-Unsigned type, value depending on meter type and FW version, - part 9 of the Double-Long-Unsigned type, the value most often consistent with the negotiated max_pdu_size when establishing a DLMS association, - Part 10 of the Double-Long type, value in the range + -30 if the clock synchronisation of the meter is enabled (using the CLOCK SET algorithm), - part 11 of the Double-Long-Unsigned type, the value depends on the meter type and FW version, - Part 12 of the Double-Long-Unsigned type, the value depends on the meter type and FW version.

DCSAP31: Object Meter information (2)

Description:

The purpose of the test is to verify the implementation of the Meter information object for the meter.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. L1 municipal meter available as a switch in PRIME topology,
3. PLC-PRIME sniffer.

Steps:	Expected results:

1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). *Note the device_id of the L1 meter.* 2. Send Action-Request-Normal command to L1 meter for object 40102 / 0-100: 64.0.0 * 255/5 (disconnect_con ()). 3. Observe PLC traffic with the PLC-PRIME sniffer. 4. Wait for the reconnection of the L1 meter. 5. Send Action-Request-Normal command to L1 meter for object 40102 / 0-100: 64.0.0 * 255/4 (disconnect_topo ()). 6. Observe PLC traffic with the PLC-PRIME sniffer. 7. Wait for the reconnection of the L1 meter.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Received Action-Response-Normal response with *success* . 3. The DCU sent a PRIME CON packet to the L1 meter with the flag negative = 1. All the DLMS associations with the meter were broken. The meter is temporarily marked as unavailable on the list of meters. The descendants of the L1 switch in the topology are still available and can be communicated with. 4. The L1 meter is available again in the DCU. Its place in the PRIME topology has not changed. The meter has the same PRIME MAC address (NID). 5. Received Action-Response-Normal response with *success* . 6. The DCU has unregistered the L1 meter and all its descendants (note: it could have done so without active communication with the meters). The L1 meter and all its descendants in the PRIME topology are temporarily unavailable. 7. The L1 meter is available again in the DCU. Its place in the PRIME topology may have changed. The meter has a different PRIME MAC address (NID).

DCSAP32: Object Meter information (3)

Description:

The purpose of the test is to verify the implementation of the Meter information object for the meter.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. L1 municipal meter available.

Steps:	**Expected results:**
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). *Note the device_id of the L1 meter.* 2. Send the Action-Request-Normal command to L1 meter for object 40102 / 0-100: 64.0.0 * 255/2 (delete ()). 3. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters).	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Received Action-Response-Normal response with *success*. 3. Get-Response-Normal response with list of meetrs received; an entry for the L1 meter is missing from the list.

DCSAP33: Object Meter information (4)

Description:

The purpose of the test is to verify the implementation of the Meter information object for the meter.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. L1 municipal meter available.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). *Note the device_id of the L1 meter.* 2. Send the command Get-Request-Normal 40160 / 0-100: 160.0.1.255/2 to the L1 meter. 3. Send the Action-Request-Normal command to L1 meter for object 40102 / 0-100: 64.0.0 * 255/3 (delete_profiles ()). 4. Send the command Get-Request-Normal 40160 / 0-100: 160.0.1.255/2 to the L1 meter.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. A Get-Response-Normal response was received with a list of profiles currently defined for the L1 meter. 3. Received Action-Response-Normal response with *success*. 4. Get-Response-Normal response was received with an empty list.

DCSAP34: Communication with meters - changing configuration of meters

Description:

The purpose of the test is to verify the correctness of the meter configuration change.

Test requirements:

1. DCSAP protocol client with an open session to the DCU,
2. L1 municipal meter available,
3. The active tariff on the meter is different than that set in step 2 of the test.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters); *Note the device_id of the L1 meter.* 2. Send Set-Request-With-List command to L1 meter for the following attributes and object value 20 / 0-0: 13.0.0.255: a. attribute 6, value of the Octet-String type = 090447313257, b. attribute 7, value of type Array of Structure Seasonal profile = `0101020309010109OCFFFF0101FF0000000080000090101` ,	1.

c. attribute 8, value of the Array of Structure type Weekly profile = 0101020809010111011101110111011101110111021102 ,

d. Attribute 9, Array of Structure. Daily profile = ,
0102020211010105020309040000000000906FFFFFFFFFFFF12000202030
904060000000906FFFFFFFFFFFF120001020309040D0000000906FFFFFF
FFFFFF120002020309040F0000000906FFFFFFFFFFFF120001020309041
60000000906FFFFFFFFFFFF12000202021102010102030904000000000 9
06FFFFFFFFFFF120002

e. Attribute 10, Date-Time value as Octet-String = Any time in the past in relation to the meter clock.

3. Send the Set-Request-Normal command to the meter for the Special Days (Passive) object (11 / 0-0; 11.0.4 * 255)
11 / 0-0: 11.0.4 * 255/2 (entries) with DLMS data = ,
010D020312000 1090507DE0101FF1102020312000 2090507DE0106FF1102020312000
3090507DE0414FF1102020312000 4090507DE0415FF1102020312000 5090507DE0501
FF1102020312000 6090507DE0503FF1102020312000 7090507DE0608FF1102020312 0
008090507DE0613FF1102020312000 9090507DE080FFF1102020312000A090507DE0B
01FF1102020312000B090507DE0B0BFF1102020312000C090507DE0C19FF110202031
2000D090507DE0C1AFF1102

4. Send the Action-Request-Normal command to the meter for the calendar activation object (20 / 0-0: 13.0.0.255) to activate the G12W tariff:

a. 20/0-0:13.0.0*255/1 (activate_passive_calendar) with data: DLMS = 00.

5. Send command to the Get-Request-With-List for object 20/0-0:13.0.0*255.
20/0-0:13.0.0*255/2 (calendar_name_active)
20/0-0:13.0.0*255/3 (season_profile_active)
20/0-0:13.0.0*255/4 (week_profile_table_active)
20/0-0:13.0.0*255/5 (day_profile_table_active)

6. Send the Get-Request-Normal command to the meter for Active Special days (11 / 0-0: 11.0.0 * 255):
11 / 0-0: 11.0.0 * 255/2 (entries).

7. Check if the new tariff has been activated in the meter.

	2.

	3.

	4.

	5.

	6.

	7. *Verification of the tariff activation should be performed using the diagnost*

	ic soft war e pro vide d by the met er ma nuf actu rer (via the opt opo rt) or usin g the info rma tion avai labl e on the met er LC D disp lay.

DCSAP35: Remotely enable / disable DCU's interfaces

Description:

The purpose of the test is to verify the possibility of remote (DCSAP protocol) switching off and on of individual DCU's interface.

Test requirements:

1. DCSAP client with session connected to the DCU.

Steps:	**Expected results:**

1. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 170.0.1 * 255. attributes 2-4 *Make a note of the list returned.* 2. Send the command Set-Request-Normal 40100 / 0-100: 170.0.1 * 255/2 to the DCU with the value of the Array of Octet-String type = modified list obtained in step 1 in such a way that it contains the content 'web = 0' 3. Connect to the DCU's web interface 4. Send the command Set-Request-Normal 40100 / 0-100: 170.0.1 * 255/2 to the DCU with the value of the Array of Octet-String type = original list received in step 1 5. Reconnect to the DCU's web interface. 6. Repeat steps 2-5 for the SSH interface ('ssh = 0').	1. After sending the interface configuration with the disabled WWW interface (web = 0) to the DCU via DCSAP protocol, the WWW interface stops working 2. After restoring the web interface with the DCSAP protocol, the functionality of the web interface is restored. 3. The SSH interface behaves in a similar way.

DCSAP36: Network interface configuration - WAN (1)

Description:

The purpose of the test is to verify the possibility of reconfiguration of network interfaces.

Test requirements:

1. DCSAP client with session connected to the DCU.

Steps:	Expected results:
1. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 170.0.2 * 255. attributes 2-4 *Make a note of the list returned.* 2. Send the command Set-Request-Normal 40100 / 0-100: 170.0.2 * 255/2 to the DCU with the value of the Array of Octet-String type = modified list received in step 1 - modified static IP address of the device; example value: `[ octet-string('static 192.168.1.8 255.255.255.0 192.168.1.1') ]` *If the DCSAP session was established via the WAN interface, the DCSAP session should be re-established to the new IP address.* 3. Verify that the DCU is available at the new IP address. 4. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 170.0.2 * 255. attributes 2-4 5. Use webGUI to verify the correctness of the network interface settings on the website and the event list.	1. Get-Response-Normal was obtained with the *success* result and data consistent with the actual state. 2. (depending on the implementation) - Set-Response-Normal was obtained with a *success* result or the DCSAP session was closed. 3. (the DCU is available at the new IP address) 4. Get-Response-Normal received with the result of *success* and data consistent with the actual state (set in (2)) 5. The state of the network interfaces on the webGUI is as set. In the DCU events, one can observe a collision about the reconfiguration of the WAN network interface.

DCSAP37: Network interface configuration - WAN (2)

Description:

The purpose of the test is to verify the possibility of reconfiguration of network interfaces.

Test requirements:

1. DCSAP client with session set up for DCU,
2. WAN interface of the DCU configured for a static IP address,
3. DHCP server available on the subnet connected to the WAN port of the DCU.

Steps:	Expected results:
1. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 170.0.2 * 255. attributes 2-4 *Make a note of the list returned.* 2. Send the command Set-Request-Normal 40100 / 0-100: 170.0.2 * 255/2 to the DCU with the value of the Array of Octet-String = type `[ octet-string('dhcp') ]` *If the DCSAP session was established via the WAN interface, the DCSAP session should be re-established to the new IP address* 3. Verify that the DCU is available at the new IP address. 4. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 170.0.2 * 255. attribute 2 5. Use webGUI to verify the correctness of the network interface settings on the website and the event list.	1. Get-Response-Normal was obtained with the *success* result and data consistent with the actual state. 2. (depending on the implementation) - Set-Response-Normal was obtained with a *success* result or the DCSAP session was closed. 3. (the DCU is available at the new IP address) 4. Get-Response-Normal received with the result *success* and data consistent with the actual state (apart from the entry 'dhcp' there will also be information about the received IP address from the DHCP server) 5. The state of the network interfaces on the webGUI is as set. In the DCU events, one can observe a collision about the reconfiguration of the WAN network interface.

DCSAP38: Network interface configuration - LAN (1)

Description:

The purpose of the test is to verify the possibility of network interfaces reconfiguration.

Test requirements:

1. DCSAP client with session connected to the DCU.

Steps:	Expected results:
1. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 170.0.8 * 255. attributes 2-4 *Make a note of the list returned.* 2. Send the command Set-Request-Normal 40100 / 0-100: 170.0.8 * 255/2 to the DCU with the value of the Array of Octet-String type = modified list received in step 1 - modified static IP address of the device; example value: `[ octet-string('static 192.168.1.8 255.255.255.0 192.168.1.1') ]` *If the DCSAP session was established via the WAN interface, the DCSAP session should be re-established to the new IP address.* 3. Verify that the DCU is available at the new IP address. 4. Send the Get-Request-With-List command to the DCU for object 40100 / 0-100: 170.0.8 * 255. attributes 2-4 5. Use webGUI to verify the correctness of the network interface settings on the website and the event list.	1. Get-Response-Normal was obtained with the *success* result and data consistent with the actual state. 2. (depending on the implementation) - Set-Response-Normal was obtained with a *success* result or the DCSAP session was closed. 3. (the DCU is available at the new IP address) 4. Get-Response-Normal received with the result of *success* and data consistent with the actual state (set in (2)) 5. The state of the network interfaces on the webGUI is as set. In the DCU events, one can observe a collision concerning the reconfiguration of the LAN network interface.

DCSAP40: Support for time stamps in DLMS queries

Description:

Verification whether the DCU interprets time stamps in inquiries received from the acquisition system correctly. The DCU software runs in UTC time, the test can be performed in any time zone in summer or winter time.

Test requirements:

1. DCU connected to the PLC network,
2. DCSAP client with session set up for the DCU,
3. L1 meter connected to the PLC network and registered in the DCU. The L1 meter must implement the correct time stamp handling.
4. The DCU has a registered hourly profile for the L1 meter (DCSAP7 test performed for the L1 meter).
5. The DCU has the data of the L1 meter hourly profile for the T0-T1 time (UTC) stored in the local database (DCSAP7 test performed for the L1 meter).

Steps:	Expected results:
1. Get meter list from DCU and determine device_id of L1 meter (= DEVID1).	1. L1 meter connected to the DCU as device_id DEVID1.

2. Send a query to the DCU (DEVID1) for an hourly profile, with a time limit: start: = T0, deviation = 0, DST = 0 end: = T1, deviation = 0, DST = 0 3. Send command to the DCU (DEVID1) a request for an hourly profile, with a time limit: start: = T0 + 2h, deviation = 120, DST = 1 end: = T1 + 2h, deviation = 120, DST = 1 4. Send a query to the DCU (DEVID1) for an hourly profile, with a time limit: start: = T0 + 1h, deviation = 60, DST = 0 end: = T1 + 2h, deviation = 120, DST = 1 5. Send command to DCU the (DEVID1) a request for an hourly profile, with a time limit: start: = T0 + 2h, deviation = 120, DST = 1 end: = T1 + 1h, deviation = 60, DST = 0 6. Send an hourly profile request to the DCU (DEVID1) with a time limit: start: = T0-3h, deviation = -180, DST = 0 end: = T1 + 6h, deviation = 480, DST = 1	DLMS responses were received, with profile data for T0-T1.

DCSAP41: DLMS query handling

Description:

Verification whether the DCU supports various types of DLMS queries.

Test requirements:

1. DCSAP client with session connected to the DCU.

Steps:	Expected results:
1. Send the command Get-Request-Normal: 40101 / 0-100: 0.0.1 * 255/2 (dcu_firmware / version) to the DCU (device_id = 0). 2. Send command to the DCU: (device_id=0) polecenie Get-Request-With-List: 40101/0-100:0.0.1*255/2 (dcu_firmware/version), 40101/0-100:0.0.1*255/4 (dcu_firmware/last_update_time), 40101/0-100:0.0.1*255/66 (dcu_firmware/invalid), 40101/0-100:0.0.1*255/5 (dcu_firmware/last_update_id). 3. Send command to the DCU:(device_id=0) polecenie Set-Request-Normal: 1/0-100.32.0.0*255/2 (dcsap_session_cached/value) + data=bool(true). 4. Send command to the DCU: (device_id=0) polecenie Set-Request-With-List: 1/0-100.32.0.0*255/2 (dcsap_session_cached/value) + data=bool(true), 40101/0-100:0.0.1*255/2 (dcu_firmware/version) + data=bool(true), 1/0-100.32.0.1*255/2 (dcsap_session_notifications/value) + data=bool(true). 5. Send command to the DCU: (device_id=0) request Action-Request-Normal: 40001/0-100.0.0.3*255/2 (dcu_event_list/push) + data=(seq=0, timestamp=0, dev_id=0, reason=0, status=0, data=octet-string('action-normal-1'))	1. Get-Response-Normal response received with octet-string data. 2. Received Get-Response-With-List with following data: SUCCESS + octet-string, SUCCESS + double-long-unsigned, DLMS ERROR -4 (object undefined), SUCCESS + double-long. 3. Set-Response-Normal response with SUCCESS result. 4. Received Set-Response-With-List with following results: SUCCESS, DLMS ERROR -3 (read-write-

6. Send command to the DCU: (device_id=0) polecenie Action-Request-With-List: 40001/0-100.0.0.3*255/2 (dcu_event_list/push) + data=(seq=0, timestamp=0, dev_id=0, reason=0, status=0, data=octet-string('action-list-1')), 40001/0-100.0.0.3*255/66 (dcu_event_list/invalid) + data=(seq=0, timestamp=0, dev_id=0, reason=0, status=0, data=octet-string('action-list-1a')), 40001/0-100.0.0.3*255/2 (dcu_event_list/push) + data=(seq=0, timestamp=0, dev_id=0, reason=0, status=0, data=octet-string('action-list-2')),	denied), SUCCESS. 5. Action-Response-Normal response with SUCCESS result, 6. Received Action-Response-With-List with following results: SUCCESS, DLMS ERROR -4 (object undefined), SUCCESS.

DCSAP42: Error code EINVALID handling - invalid DLMS query format

Description:

Verification whether the DCU will respond with the error code EINVALID (-4) for a command with an incorrect DLMS format.

Test requirements:

1. DCSAP client with session connected to the DCU.

Steps:	Expected results:
1. Send the command Get-Request-Normal: 1 / 0-100: 32.0.0 * 255/2 to the DCU, but with the last byte of the DLMS query truncated.	1. DCSAP error response with code -4 (EINVALID) received.

DCSAP50: DCSAP SSL support (1)

Description:

Verification whether the DCU supports encryption and authorization using SSL in the DCSAP protocol.

Test requirements:

1. DCSAP SSL client,
2. CA-signed client certificate used in the DCSAP SSL service,
3. Network traffic sniffer connected inbetween the DCSAP client and the DCU.

Steps:	Expected results:
1. Establish a DCSAP SSL connection with the DCU. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters).	1. A DCSAP SSL session has been successfully established. The process of establishing an SSL session can be observed on the network traffic sniffer. 2. Correct Get-Response-Normal response with a list of meters received. The data seen on the traffic sniffer is encrypted. 3. Get-Response-Normal response received with result *success* and value 2 (remote SSL)

3. Send the command Get-Request-Normal 1 / 0-100: 32.1.0.255/2 (session origin) to the DCU. 4. Send the command Get-Request-Normal 1 / 0-100: 32.1.1.255/2 (session common name) to the DCU. 5. Send the command Get-Request-Normal 1 / 0-100: 32.1.2.255/2 (session permission group) to the DCU.	4. Get-Response-Normal response was received with *success* result and value equal to the Common Name field of the certificate. 5. Get-Response-Normal response received with result *success* and value 2 (default permission group for SSL connection).

DCSAP51: DCSAP SSL support (2)

Description:

Verification whether the DCU supports encryption and authorization using SSL in the DCSAP protocol.

Test requirements:

1. DCSAP SSL client,
2. Client certificate unsigned by CA used in DCSAP SSL service (e.g. self-signed certificate),
3. Network traffic sniffer connected inbetween the DCSAP client and the DCU.

Steps:	Expected results:
1. Try to establish a DCSAP SSL connection with the DCU.	1. DCSAP SSL session has been rejected. The unsuccessful process of establishing an SSL session can be observed on the network traffic sniffer.

DCSAP52: DCSAP SSL support (3)

Description:

Verification whether the DCU supports encryption and authorization using SSL in the DCSAP protocol.

Test requirements:

1. DCSAP SSL client,
2. Client certificate signed by CA used in DCSAP SSL service (e.g. self-signed certificate),
3. Network traffic sniffer connected inbetween the DCSAP client and DCU.

Steps:	Expected results:
1. Try to establish a DCSAP SSL connection with the DCU.	1. A DCSAP SSL session has been successfully established. The process of establishing an SSL session can be observed on the network traffic sniffer. DCU presented itself with the appropriate SSL certificate - the Common Name field contains the serial number of the device or its domain name.

DCSAP53: DCSAP SSL support (4)

Description:

Verification whether the DCU supports encryption and authorization using SSL in the DCSAP protocol.

Test requirements:

1. DCSAP SSL client,
2. CA-signed client certificate used in the DCSAP SSL service,
3. The Common Name field of the certificate ends with the string '\\ 3' ('\\' → delimiter of SSL authorization group, '3' → read-only authorization group),
4. Network traffic sniffer connected inbetween the DCSAP client and the DCU.

Steps:	Expected results:
1. Establish a DCSAP SSL connection with the DCU. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). 3. Send the command Get-Request-Normal 1 / 0-100: 32.1.0.255/2 (session origin) to the the DCU. 4. Send the command Get-Request-Normal 1 / 0-100: 32.1.1.255/2 (session common name) to the DCU. 5. Send the command Get-Request-Normal 1 / 0-100: 32.1.2.255/2 (session permission group) to the DCU. 6. Send the command Set-Request-Normal 40100 / 0-100: 0.0.4 * 255/2 (NTP list) to the DCU with the value of the Array of Octet-String type containing correct IP addresses.	1. A DCSAP SSL session has been successfully established. The process of establishing an SSL session can be observed on the network traffic sniffer. 2. Correct Get-Response-Normal response with a list of meters received. The data seen on the traffic sniffer is encrypted. 3. Get-Response-Normal response received with result *success* and value 2 (remote SSL) 4. Get-Response-Normal response was received with *success* result and value equal to the Common Name field of the certificate. 5. Get-Response-Normal response was received with result *success* and value 3 (read-only SSL permission group). 6. Set-Response-Normal response received with *read-write denied result.* Object value has not changed

DCSAP54: Effective DCSAP SSL communication

Description:

Verification whether the DCU supports encryption and authorization using SSL in the DCSAP protocol.

Test requirements:

1. DCSAP SSL client,
2. CA-signed client certificate used in the DCSAP SSL service,
3. Network traffic sniffer connected inbetween the DCSAP client and the DCU.

Steps:	Expected results:

1. Establish a DCSAP SSL connection with the DCU. 2. Perform the **DCSAP11** test . 3. Perform the **DCSAP34** test. 4. Perform the **DCSAP40** test. 5. Perform the **DCSAP64** test. 6. Perform the **DCSAP95** test.	1. A DCSAP SSL session has been successfully established. The process of establishing an SSL session can be observed on the network traffic sniffer. The result of all tests is the same as in the case of using the DCSAP protocol without SSL.

DCSAP61: Meters's LLS password configuration

Description:

Validate the correctness of the LLS password configuration for the meter.

Test requirements:

1. DCSAP client with session set up for DCU,
2. Disconnected municipal meter (L1),
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send command to L1 Set-Reqest-Normal meter by setting attribute 40199 / 0-100: 65.0.3 * 255/4 to octet-string [8] with value '12345678'. 3. Connect the L1 meter. 4. Send command to the L1 meter: Get-Request-Normal with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PRIME sniffer. 5. Send command to L1 meter: Get-Request-Normal request for 40199 / 0-100 association error flags: 65.0.3 * 255/8. 6. Send command to L1 meter: Set-Reqest-Normal, set attribute 40199 / 0-100: 65.0.3 * 255/4 to octet-string [0]. 7. Send command to L1 meter: Get-Request-Normal request for 40199 / 0-100 association error flags: 65.0.3 * 255/8.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. The meter will connect to the DCU via the PLC. 4. DCSAP error = -15 (meter_handshake_falied) received. The calling-authentication-value field of the AARQ message (should be '12345678') was verified with the PRIME sniffer. 5. This association's error flags show that it is not usable (at least the fatal and fatal_aarq_rejected flags should be on). 6. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 7. This association's error flags were automatically reset by changing the association encryption / authentication parameters. 8. Get-Response-Normal with clock value received. Using the PRIME sniffer, the calling-authentication-value field of the AARQ message was verified (it should have the correct value calculated according to the algorithm from the meter number).

8. Send command to the L1 meter: Get-Request-Normal with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PRIME sniffer.	

DCSAP62: HLS authentication - MGMT Association

Description:

Checking the correctness of the HLS authentication mechanism in the MGMT association.

Test requirements:

1. DCSAP client with session set up for DCU,
2. Disconnected municipal meter (L1) with known authentication key for MGMT association,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send command to the L1: meter Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.3 * 255/7 to octet-string [16] with the authentication key value for the MGMT association. 3. Send command to the L1: meter Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.3 * 255/3 to unsigned with value 5. 4. Send command to the L1: meter Get-Request-Normal with attribute 1 / 0-100: 66.0.3 * 255/2. Note the value of the FC_RX frame counter. 5. Connect the L1 meter. 6. Send command to the L1: meter Get-Request-Normal with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PRIME sniffer. 7. Send command to the L1: meter Get-Request-Normal with attribute 1 / 0-100: 66.0.3 * 255/2. 8. Send command to the L1: meter Get-Request-Normal with attribute 1 / 0-1: 43.1.3 * 255/2.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 4. A Get-Response-Normal response was received with a double-long-unsigned value (RX frame counter for this meter - FC_RX). 5. The meter will connect to the DCU via the PLC. 6. Get-Response-Normal with clock value received. The PRIME sniffer verified: - Fetching the frame counter of the meter in the PUBLIC association (note the counter value as FC_RX), - the AARQ message mechanism-name field (it should have the HLS-with-GMAC mechanism OID), - Sending the correct calling-ap-title (system-title of the DCU), - AARE received with the result 14 (field result-source-diagnostic - 'high level security required'), - Sending an Action-Request-Normal to the association object by calling the reply_to_hls_challange method (the data should contain a frame counter with the value FC_RX +1), - An Action-Response-Normal response with the code Action-Result = 0 (Success) and data containing the correct frame count (FC_TX independent of FC_RX) and the generated StoC signature was received.

	7. A Get-Response-Normal response was received with a double-long-unsigned value equal to FC_RX + 1 (and equal to the value read with the sniffer). 8. Get-Response-Normal response received with double-long-unsigned value FC_RX + 1.

DCSAP63: HLS authentication - FW Update Association

Description:

Checking the correctness of the HLS authentication mechanism in the FW Update association.

Test requirements:

1. DCSAP client with session set up for DCU,
2. Disconnected municipal meter (L1) with known authentication key for FW Update association,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send Set-Reqest-Normal to L1 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/7 to octet-string [16] with the authentication key value for the FW Update association. 3. Send command to the L1 meter Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.4 * 255/3 to unsigned with value 5. 4. Send command to the L1 meter Get-Request-Normal with attribute 1 / 0-100: 66.0.4 * 255/2. Note the value of the FC_RX frame counter. 5. Connect the L1 meter. 6. Send Set-Request-Normal to DCU by setting attribute 1 / 0-100: 63.0.0 * 255/2 to unsigned (3) (Session attribute that sets the association client_id in which subsequent requests will be executed). 7. Send command to the L1 meter: Get-Request-Normal with a request for Firmware Version (1 / 1-0: 0.2.0 * 255/2) and observe communication on PRIME sniffer. 8. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-100: 66.0.4 * 255/2. 9. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-1: 43.1.4 * 255/2.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 4. A Get-Response-Normal response was received with a double-long-unsigned value (RX frame counter for this counter - FC_RX). 5. The meter will connect to the DCU via the PLC. 6. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 7. Get-Response-Normal with clock value received. The PRIME sniffer verified: - Fetching the frame counter of the meter in the PUBLIC association (note the value of the counter as FC_RX), - The mechanism-name field of the AARQ message (it should have the OID of the HLS-with-GMAC mechanism), - Sending the correct calling-ap-title (system-title of the DCU), - AARE received with the result 14 (field result-source-diagnostic - 'high level security required'), - Sending an Action-Request-Normal to the association object by calling the reply_to_hls_challange method (the data should contain a frame counter with the value FC_RX +1), - An Action-Response-Normal response was received with the Action-Result = 0 (Success) code and data containing

	the correct frame count (FC_TX, independent of FC_RX) and the generated StoC signature. 8. Get-Response-Normal response was received with double-long-unsigned value FC_RX + 1 (equal to value read with sniffer). 9. Get-Response-Normal response received with double-long-unsigned value FC_RX + 1. *In addition: frame counters for FW association are independent of frame counters for MGMT association (verification against DCSAP62 test results).*

DCSAP64: DLMS packet encryption and signing mechanism - MGMT association

Description:

Checking the correctness of the DLMS packet encryption and signing mechanism in the MGMT association

Test requirements:

1. DCSAP client with session set up for the DCU,
2. Disconnected municipal meter (L1) with known encryption and authentication key for MGMT association and security_policy for MGMT association set to 0,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send command to the L1 meter: Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.3 * 255/5 to octet-string [16] with the encryption key value for MGMT association. 3. Send command to the L1 meter: Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.3 * 255/7 to octet-string [16] with the authentication key value for the MGMT association. 4. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-100: 66.0.3 * 255/2. Note the value of the FC_RX frame counter. 5. Connect the L1 meter. 6. Send command to the L1 meter: Action-Request-Normal by calling the method 64 / 0-0: 43.0.3 * 255/1 with the enum (3 or 12) parameter (security_activate - forcing encryption and signing of packets in the MGMT association on the meter). 7. Send command to the L1 meter: Get-Request-Normal with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PRIME sniffer.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 4. A Get-Response-Normal response was received with a double-long-unsigned value (RX frame counter for this meter - FC_RX). 5. The meter will connect to the DCU via the PLC. 6. An Action-Response-Normal response was received with the code Action-Result = 0 (Success). 7. DCSAP error (-15) or Exception DLMS received. The PRIME sniffer verified that the query was not encrypted and the meter rejected it.

8. Send command to the L1 meter: Get-Request-Normal asking for 40199 / 0-100 association error flags: 65.0.3 * 255/8. 9. Send command to the L1 meter: Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.3 * 255/2 to enum (3) (enable encryption and signing of communication with this meter on DCU). 10. Send command to the L1 meter: Get-Request-Normal asking for 40199 / 0-100 association error flags: 65.0.3 * 255/8. 11. Wait for DLMS association timeout. 12. Send command to the L1 meter: Get-Request-Normal with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PRIME sniffer. 13. Send command to the L1 meter: Get-Request-Normal meter with attribute 1 / 0-100: 66.0.3 * 255/2. 14. Send command to the L1 meter: Get-Request-Normal meter with attribute 1 / 0-1: 43.1.3 * 255/2.	8. This association's error flags show that it is not usable (at least fatal flags should be lit). 9. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 10. This association's error flags were automatically reset by changing the association encryption / authentication parameters. 11. (association will be re-established at the next request) 12. Get-Response-Normal with clock value received. Using the PRIME sniffer verified: Getting the frame counter of the meter in the PUBLIC association (note the value of the counter as FC_RX), All messages are encrypted and signed The frame counters have the correct values 13. A Get-Response-Normal response was received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the meter). 14. Get-Response-Normal response received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the DCU).

DCSAP65: DLMS packet encryption and signing mechanism - FW Update association

Description:

Checking the correctness of the encryption and signing mechanism of DLMS packets in the FW Update association.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. Disconnected utility meter (L1) with known encryption and authentication key for FW association and security_policy for FW Upgrade association set to 0,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list.

2. Send Set-Reqest-Normal to the L1 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/5 to octet-string [16] with the encryption key value for the FW Update association. 3. Send Set-Reqest-Normal to the L1 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/7 to octet-string [16] with the authentication key value for the FW Update association. 4. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-100: 66.0.4 * 255/2. Note the value of the FC_RX frame counter. 5. Connect the L1 meter. 6. Send command to the L1 meter: Action-Request-Normal by calling the method 64 / 0-0: 43.0.4 * 255/1 with the enum (3 or 12) parameter (security_activate - force encryption and signing of packets in the FW Update association on the meter). 7. Send command to the L1 meter: Get-Request-Normal with a request for Firmware Version (1 / 1-0: 0.2.0 * 255/2) and observe communication on PRIME sniffer. 8. Send command to the L1 meter: Get-Request-Normal asking for 40199 / 0-100 association error flags: 65.0.4 * 255/8. 9. Send command to the L1 meter: Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.4 * 255/2 to enum (3) (enable encryption and signing of communication with this meter on DCU). 10. Send command to the L1 meter: Get-Request-Normal asking for 40199 / 0-100 association error flags: 65.0.4 * 255/8. 11. Wait for DLMS association timeout. 12. Send command to the L1 meter: Get-Request-Normal with a request for Firmware Version (1 / 1-0: 0.2.0 * 255/2) and observe communication on PRIME sniffer. 13. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-100: 66.0.4 * 255/2. 14. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-1: 43.1.4 * 255/2.	2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 4. A Get-Response-Normal response was received with a double-long-unsigned value (RX frame counter for this meter - FC_RX). 5. The meter will connect to the DCU via the PLC. 6. An Action-Response-Normal response was received with the code Action-Result = 0 (Success). 7. DCSAP error (-15) or Exception DLMS received. The PLC-PRIME sniffer verified that the query was not encrypted and the meter rejected it. 8. This association's error flags show that it is not usable (at least fatal flags should be lit). 9. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 10. This association's error flags were automatically reset by changing the association encryption / authentication parameters. 11. (association will be re-established at the next request) 12. Get-Response-Normal with clock value received. Using the PRIME sniffer verified: Getting the frame counter of the meter in the PUBLIC association (note the value of the counter as FC_RX), All messages are encrypted and signed. The frame counters have the correct values. 13. A Get-Response-Normal response was received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the meter). 14. Get-Response-Normal response received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the DCU). *Moreover: the frame counters for the FW Update association are independent of the frame counters for the MGMT association (verified against the results of the DCSAP64 test).*

DCSAP66: DLMS packet encryption mechanism - MGMT association

Description:

Checking the correctness of the DLMS packet encryption mechanism in the MGMT association

Test requirements:

1. DCSAP client with session set up for the DCU,
2. Disconnected municipal meter (L1) with known encryption key for MGMT association and security_policy for MGMT association set to 0,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send command to the L1 meter: Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.3 * 255/5 to octet-string [16] with the encryption key value for MGMT association. 3. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-100: 66.0.3 * 255/2. Note the value of the FC_RX frame counter. 4. Connect the L1 meter. 5. Send command to the L1 meter: Action-Request-Normal by calling the method 64 / 0-0: 43.0.3 * 255/1 with the enum (2 or 8) parameter (security_activate - forcing packet encryption in the MGMT association on the meter). 6. Send command to the L1 meter: Get-Request-Normal with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PRIME sniffer. 7. Send command to the L1 meter: Get-Request-Normal asking for 40199 / 0-100 association error flags: 65.0.3 * 255/8. 8. Send command to the L1 meter: Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.3 * 255/2 to enum (2) (enable encryption of communication with this meter on DCU). 9. Send command to the L1 meter: Get-Request-Normal asking for 40199 / 0-100 association error flags: 65.0.3 * 255/8. 10. Wait for DLMS association timeout. 11. Send command to the L1 meter: Get-Request-Normal with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PLC-PRIME sniffer. 12. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-100: 66.0.3 * 255/2.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. A Get-Response-Normal response was received with a double-long-unsigned value (RX frame counter for this meter - FC_RX). 4. The meter will connect to the DCU via the PLC. 5. An Action-Response-Normal response was received with the code Action-Result = 0 (Success). 6. DCSAP error (-15) or Exception DLMS received. The PRIME sniffer verified that the query was not encrypted and the meter rejected it. 7. This association's error flags show that it is not usable (at least fatal flags should be lit). 8. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 9. This association's error flags were automatically reset by changing the association encryption / authentication parameters. 10. (association will be re-established at the next request) 11. Get-Response-Normal with clock value received. Using the PLC-PRIME sniffer verified: Getting the frame counter of the meter in the PUBLIC association (note the value of the counter as FC_RX), All messages are encrypted Frame counters have correct values

13. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-1: 43.1.3 * 255/2.	12. A Get-Response-Normal response was received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the meter). 13. Get-Response-Normal response received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the DCU).

DCSAP67: DLMS packet encryption mechanism - FW Update association

Description:

Checking the correctness of the DLMS packet encryption mechanism in the FW Update association.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. Disconnected municipal meter (L1) with a known encryption key for FW association and security_policy for FW Upgrade association set to 0,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send Set-Reqest-Normal to the L1 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/5 to octet-string [16] with the encryption key value for the FW Update association. 3. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-100: 66.0.4 * 255/2. Note the value of the FC_RX frame counter. 4. Connect the L1 meter. 5. Send command to the L1 meter: Action-Request-Normal by calling the method 64 / 0-0: 43.0.4 * 255/1 with the enum (2 or 8) parameter (security_activate - forcing packet encryption in the FW Update association on the meter). 6. Send command to the L1 meter: Get-Request-Normal with a request for Firmware Version (1 / 1-0: 0.2.0 * 255/2) and observe communication on PRIME sniffer. 7. Send command to the L1 meter: Get-Request-Normal asking for 40199 / 0-100 association error flags: 65.0.4 * 255/8. 8. Send command to the L1 meter: Set-Reqest-Normal by setting attribute 40199 / 0-100: 65.0.4 * 255/2 to enum	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. A Get-Response-Normal response was received with a double-long-unsigned value (RX frame counter for this meter - FC_RX). 4. The meter will connect to the DCU via the PLC. 5. An Action-Response-Normal response was received with the code Action-Result = 0 (Success). 6. DCSAP error (-15) or Exception DLMS received. The PLC-PRIME sniffer verified that the query was not encrypted and the meter rejected it. 7. This association's error flags show that it is not usable (at least fatal flags should be lit). 8. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 9. This association's error flags were automatically reset by changing the association encryption / authentication parameters. 10. (association will be re-established at the next request)

(2) (enable encryption of communication with this meter on DCU). 9. Send command to the L1 meter: Get-Request-Normal asking for 40199 / 0-100 association error flags: 65.0.4 * 255/8. 10. Wait for DLMS association timeout. 11. Send command to the L1 meter: Get-Request-Normal with a request for Firmware Version (1 / 1-0: 0.2.0 * 255/2) and observe communication on PRIME sniffer. 12. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-100: 66.0.4 * 255/2. 13. Send command to the L1 meter: Get-Request-Normal with attribute 1 / 0-1: 43.1.4 * 255/2.	11. Get-Response-Normal with clock value received. Using the PLC-PRIME sniffer verified: Getting the frame counter of the meter in the PUBLIC association (note the value of the counter as FC_RX). All messages are encrypted and signed. The frame counters have the correct values. 1. A Get-Response-Normal response was received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the meter). 2. Get-Response-Normal response received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the DCU). *Moreover: the frame counters for the FW Update association are independent of the frame counters for the MGMT association (verified with the results of the DCSAP66 test).*

DCSAP68: DLMS packet signing mechanism - MGMT association

Description:

Checking the correctness of the DLMS packet signing mechanism in the MGMT association

Test requirements:

1. DCSAP client with session set up for the DCU,
2. Disconnected utility meter (L1) with known authentication key for MGMT association and security_policy for MGMT association set to 0,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	**Expected results:**
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send Set-Reqest-Normal to the L1 meter by setting attribute 40199 / 0-100: 65.0.3 * 255/7 to octet-string [16] with the authentication key value for the MGMT association. 3. Send Get-Request-Normal to the L1 meter with attribute 1 / 0-100: 66.0.3 * 255/2. Note the value of the FC_RX frame counter. 4. Connect the L1 meter.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. A Get-Response-Normal response was received with a double-long-unsigned value (RX frame counter for this meter - FC_RX). 4. The meter will connect to the DCU via the PLC.

5. Send Action-Request-Normal to the L1 meter by calling the method 64 / 0-0: 43.0.3 * 255/1 with the enum (1 or 4) parameter (security_activate - force signing packets in the MGMT association on the meter). 6. Send Get-Request-Normal to the L1 meter with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PRIME sniffer. 7. Send Get-Request-Normal to the L1 meter asking for 40199 / 0-100 association error flags: 65.0.3 * 255/8. 8. Send Set-Reqest-Normal to the L1 meter by setting attribute 40199 / 0-100: 65.0.3 * 255/2 to enum (1) (enable signing of communication with this meter on DCU). 9. Send Get-Request-Normal to the L1 meter asking for 40199 / 0-100 association error flags: 65.0.3 * 255/8. 10. Wait for DLMS association timeout. 11. Send Get-Request-Normal to the L1 meter with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PRIME sniffer. 12. Send Get-Request-Normal to the L1 meter with attribute 1 / 0-100: 66.0.3 * 255/2. 13. Send Get-Request-Normal to the L1 meter with attribute 1 / 0-1: 43.1.3 * 255/2.	5. An Action-Response-Normal response was received with the code Action-Result = 0 (Success). 6. DCSAP error (-15) or Exception DLMS received. The PRIME sniffer verified that the query was not encrypted and the meter rejected it. 7. This association's error flags show that it is not usable (at least fatal flags should be lit). 8. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 9. This association's error flags were automatically reset by changing the association encryption / authentication parameters. 10. (association will be re-established at the next request) 11. Get-Response-Normal with clock value received. The PRIME sniffer verified: Getting the frame counter of the meter in the PUBLIC association (note the value of the counter as FC_RX), All messages are signed Frame counters have correct values 12. A Get-Response-Normal response was received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the meter). 13. Get-Response-Normal response received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the DCU).

DCSAP69: DLMS packet signing mechanism - FW Update association

Description:

Checking the correctness of the DLMS package signing mechanism in the FW Update association.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. Disconnected communal meter (L1) with known authentication key for FW association and security_policy for FW Upgrade association set to 0,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:

1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send Set-Reqest-Normal to the L1 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/7 to octet-string [16] with the authentication key value for the FW Update association. 3. Send Get-Request-Normal to the L1 meter with attribute 1 / 0-100: 66.0.4 * 255/2. Note the value of the FC_RX frame counter. 4. Connect the L1 meter. 5. Send Action-Request-Normal to the L1 meter by calling the method 64 / 0-0: 43.0.4 * 255/1 with the enum (1 or 4) parameter (security_activate - force signing packets in the FW Update association on the meter). 6. Send Get-Request-Normal to the L1 meter with a request for Firmware Version (1 / 1-0: 0.2.0 * 255/2) and observe communication on PRIME sniffer. 7. Send Get-Request-Normal to the L1 meter asking for 40199 / 0-100 association error flags: 65.0.4 * 255/8. 8. Send Set-Reqest-Normal to the L1 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/2 to enum (1) (enable signing of communication with this meter on DCU). 9. Send Get-Request-Normal to the L1 meter asking for 40199 / 0-100 association error flags: 65.0.4 * 255/8. 10. Wait for DLMS association timeout. 11. Send Get-Request-Normal to the L1 meter with a request for Firmware Version (1 / 1-0: 0.2.0 * 255/2) and observe communication on PLC-PRIME sniffer. 12. Send Get-Request-Normal to the L1 meter with attribute 1 / 0-100: 66.0.4 * 255/2. 13. Send Get-Request-Normal to the L1 meter with attribute 1 / 0-1: 43.1.4 * 255/2.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. A Get-Response-Normal response was received with a double-long-unsigned value (RX frame counter for this meter - FC_RX). 4. The meter will connect to the DCU via the PLC. 5. An Action-Response-Normal response was received with the code Action-Result = 0 (Success). 6. DCSAP error (-15) or Exception DLMS received. The PLC-PRIME sniffer verified that the query was not encrypted and the meter rejected it. 7. This association's error flags show that it is not usable (at least fatal flags should be lit). 8. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 9. This association's error flags were automatically reset by changing the association encryption / authentication parameters. 10. (association will be re-established at the next request) 11. Get-Response-Normal with clock value received. The PLC-PRIME sniffer verified: Getting the frame counter of the meter in the PUBLIC association (note the value of the counter as FC_RX), All messages are signed Frame counters have correct values 12. A Get-Response-Normal response was received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the meter). 13. Get-Response-Normal response received with a double-long-unsigned value greater than FC_RX (equal to the value in the last message sent by the DCU). *Moreover: the frame counters for the FW Update association are independent of the frame counters for the MGMT association (verified with the results of the DCSAP68 test).*

DCSAP80: FW update of meters in broadcast mode - unencrypted

Description:

Checking the correctness of broadcast transmission - to transfer software (*firmware*) to meters

Test requirements:

1. DCSAP client with session set up for the DCU,
2. At least 2 meters connected to the DCU (L1, L2) of the same type supporting encrypted broadcast FW upgrade - for which the broadcast encryption key for the FW upgrade association is known,
3. DCSAP67 test performed for L1, L2 meters - encryption turned on in the FW upgrade association,
4. Image package for updating meters available via URL in http (or https) - *fw_update_url* ,
5. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Set the minimum number of meters for which broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to 2. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 and L2 meters. 3. Send command to L1 Set-Reqest-Normal meter by setting attribute 40199 / 0-100: 65.0.4 * 255/6 to octet-string [16] with broadcast encryption key value for FW Update association. 4. Send Set-Reqest-Normal to L2 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/6 to octet-string [16] with broadcast encryption key value for FW Update association. 5. Start the FW upgrade procedure on the L1 meter by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L1 with the parameter Octet-String *fw_update_url.* 6. Start the FW upgrade procedure on the L2 meter by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L2 with the Octet-String parameter *fw_update_url* (important: this step must be performed before the FW upgrade procedure on the L1 meter is completed). 7. Observe PLC movement with the PLC-PRIME sniffer. 8. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; the list includes an entry for the L1 and L2 meters. 3. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 4. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 5. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 6. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 7. After completing step (3) - the DCU establishes an encrypted FW upgrade association in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode, After completing the step (4) - the DCU establishes an encrypted FW upgrade association in the unicast mode with the L2 meter to start the update process, and then starts transmitting encrypted blocks in broadcast mode. Periodically the DCU exchanges unicast messages with the L1 and L2 meters to prevent breaking the association FW Upgrade, It is impossible to view the content of DLMS messages. 8. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update 9. The FW upgrade process has been successfully completed - both the DCSAP and the LCD can be used to read the new software version.

(40101 / 0-100: 0.0.1.255/7) for updating the L1 and L2 meters. 9. Wait until the FW upgrade on both meters is completed - verify the FW version on the meters using DCSAP / LCD / meter diagnostic software.	

DCSAP81: FW update of meters in broadcast mode - encrypted

Description:

Checking the correctness of encrypted broadcast transmission - in order to transfer software (*firmware*) to meters

Test requirements:

1. DCSAP client with session set up for the DCU,
2. At least 2 meters connected to the DCU (L1, L2) of the same type supporting encrypted broadcast FW upgrade - for which the broadcast encryption key for the FW upgrade association is known,
3. DCSAP67 test performed for L1, L2 meters - encryption turned on in the FW upgrade association,
4. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
5. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	**Expected results:**
1. Set the minimum number of meters for which broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to 2. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 and L2 meters. 3. Send Set-Reqest-Normal to the L1 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/6 to octet-string [16] with broadcast encryption key value for FW Update association. 4. Send Set-Reqest-Normal to the L2 meter by setting attribute 40199 / 0-100: 65.0.4 * 255/6 to octet-string [16] with broadcast encryption key value for FW Update association. 5. Start the FW upgrade procedure on L1 by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to the L1 meter with the parameter Octet-String *fw_update_url.* 6. Start the FW upgrade procedure on L2 by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to the L2 meter with the Octet-String parameter *fw_update_url*	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; the list includes an entry for the L1 and L2 meters. 3. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 4. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 5. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 6. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 7. After completing step (3) - the DCU establishes an encrypted FW upgrade association in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode, After completing the step (4) - the DCU establishes an encrypted FW upgrade association in the unicast mode with L2 meter to start the update process, and then starts transmitting encrypted blocks in broadcast mode. Periodically the DCU exchanges unicast messages with L1, L2 meters to prevent breaking the association FW Upgrade,

(important: this step must be performed before the FW upgrade procedure on L1 is completed). 7. Observe PLC movement with the PRIME sniffer. 8. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) for updating the L1 and L2 meters. 9. Wait until the FW upgrade on both meters is completed - verify the FW version on the meters using DCSAP / LCD / meter diagnostic software.	It is impossible to suspect the content of DLMS messages. 8. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update 9. The FW upgrade process has been successfully completed - both the DCSAP and the LCD can be used to read the new software version.

DCSAP82: FW update of meters in unicast mode - unencrypted

Description:

Checking the correctness of unicast transmission - in order to transfer software (*firmware*) to meters.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. One meter connected to DCU (L1),
3. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
4. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	**Expected results:**
1. Set the minimum number of meters where broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to more than 1. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 3. Start the FW upgrade procedure on the L1 meter by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L1 with the parameter Octet-String *fw_update_url.* 4. Observe PLC movement with the PRIME sniffer. 5. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) of the L1 meter update 6. Wait until the FW upgrade on the meter is completed - verify the FW version on the meter using DCSAP / LCD / meter diagnostic software.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 3. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 4. The DCU establishes the association of FW upgrade in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode, 5. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update 6. The FW upgrade process has been successfully completed - both the DCSAP and the LCD can be used to read the new software version.

DCSAP83: FW update of meters in unicast mode - encrypted

Description:

Verification of the correctness of the encrypted unicast transmission - in order to send the software (*firmware*) to the meters.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. One meter connected to DCU (L1),
3. DCSAP67 test performed for L1, L2 meters - encryption turned on in FW upgrade association,
4. Image package for updating meters available via URL in http (or https) - *fw_update_url* ,
5. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Set the minimum number of meters where broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to more than 1. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 3. Start the FW upgrade procedure on the L1 meter by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L1 with the parameter Octet-String *fw_update_url.* 4. Observe PLC traffic with the PLC-PRIME sniffer. 5. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) for updating the L1 and L2 meters. 6. Wait until the FW upgrade on the meter is completed - verify the FW version on the meters using DCSAP / LCD / meter diagnostic software.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 3. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 4. The DCU establishes an encrypted association of FW upgrade in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode. Suspecting the content of DLMS messages is impossible - all messages are encrypted. 5. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update 6. The FW upgrade process has been successfully completed - both the DCSAP and the LCD can be used to read the new software version.

DCSAP84: Updating FW of meters in unicast mode - encrypted and signed

Description:

Verification of the correctness of the encrypted unicast transmission - in order to send the software (*firmware*) to the meters.

Test requirements:

1. DCSAP client with session set up for DCU,
2. One meter connected to DCU (L1),

3. DCSAP65 test performed for L1, L2 meters - enabling encryption and signing in the FW upgrade association,
4. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
5. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Set the minimum number of meters where broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to more than 1. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 3. Start the FW upgrade procedure on the L1 meter by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L1 with the parameter Octet-String *fw_update_url.* 4. Observe PLC traffic with the PLC-PRIME sniffer. 5. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) for updating the L1 and L2 meters. 6. Wait until the FW upgrade on the meter is completed - verify the FW version on the meters using DCSAP / LCD / meter diagnostic software.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 3. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 4. The DCU establishes an encrypted association of FW upgrade in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode. Suspicion of the DLMS message content is impossible - all messages are encrypted and signed. 5. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update 6. The FW upgrade process has been successfully completed - both the DCSAP and the LCD can be used to read the new software version.

DCSAP85: FW update of meters - conditional (1)

Description:

Verification of the correctness of the conditional FW update of meters.

Test requirements:

1. DCSAP client with session set up for DCU,
2. One meter connected to DCU (L1),
3. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
4. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Set the minimum number of meters where broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to more than 1. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list.

3. Send command to the L1 meter Get-Request-Normal 1 / 1-0: 0.2.0.255/2 (reading of the current FW version of the meter). 4. Start to conditional FW upgrade L1 on L1 sending a Request-Action-Normal 40101 / 0-100: 0.0.1.255/3 parameter struct: Field `https_url` Type Octet-String *fw_update_url* field `dest_fw_version` acts on the data received in (3) 5. Observe PLC traffic with the PLC-PRIME sniffer. 6. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) of the L1 meter update.	3. Get-Response-Normal response with current version of meter software received. 4. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 5. The DCU establishes communication with the L1 meter in order to verify the current version of the meter firmware. The DCU does not start the FW upgrade procedure. 6. The update status changes according to the DCSAP specification. The FW upgrade process was completed successfully - despite the lack of updating from the PLC.

DCSAP86: FW update of meters - conditional (2)

Description:

Verification of the correctness of the conditional FW update of meters.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. One meter connected to DCU (L1),
3. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
4. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	**Expected results:**
1. Set the minimum number of meters where broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to more than 1. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 3. Send command to the L1 meter Get-Request-Normal 1 / 1-0: 0.2.0.255/2 (reading of the current FW version of the meter). 4. Start FW upgrade procedure conditional on sending L1 L1 Action Request-Normal-40101 / 0-100: 0.0.1.255/3 parameter struct: field `https_url` type octet-string *fw_update_url* box `dest_fw_version` set on different data than that obtained in (3) → best target FW version of the meter 5. Observe PLC traffic with the PLC-PRIME sniffer.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 3. Get-Response-Normal response with current version of meter software received. 4. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 5. The DCU establishes the association of FW upgrade in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode, 6. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update

6. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) of the L1 meter update. 7. Wait until the FW upgrade on the meter is completed - verify the FW version on the meter using DCSAP / LCD / meter diagnostic software.	7. The FW upgrade process has been successfully completed - both the DCSAP and the meter's LCD can be used to read the new software version.

DCSAP87: FW update of meters - automatic triggering of updates (1)

Description:

Verification of the correctness of the conditional FW update of meters.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. Three meters of the same manufacturer disconnected from the DCU (L1, L2, L3)
   L1 and L2 meters with the same software version *FW_VER1* ,
   L3 meter with a different software version *FW_VER3,*
3. One meter from another manufacturer L4 disconnected from the DCU.
4. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
5. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	**Expected results:**
1. Send the command Set-Request-Normal 40054 / 0-100: 0.131.0.255/3 to the DCU with an array of structures: 2. Connect the L1, L2, L3, L4 meters 3. Using the webGUI, view the DCU events and the update status of individual meters.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. The meters have successfully registered in the DCU 3. Observed: a. Automatic start of FW update of L1 and L2 meters b. The update of the L3 and L4 meters has not started 4. The FW upgrade process has been successfully completed - both the DCSAP and the meter's LCD can be used to read the new software version.

```
array [
   structure {
      prio : double-long-unsigned :=
100
      -- wildcard matching the meter
type used for performing the test
      name_wildcard  : octet-
string[13] := SAG203???????
      curr_fw : data := FW_VER1
      update_cmd : structure {
         https_url : octet-string :=
fw_update_url
         dest_fw_version : data :=
future contents of 1/1-0:0.2.0.255/2
      }
   }
]
```

4. Wait for the FW upgrade to finish on all meters - verify the FW version on the meter using DCSAP / LCD / meter diagnostic software.	

DCSAP88: FW update of meters - automatic triggering of updates (2)

Description:

Verification of the correctness of the conditional FW update of meters.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. Three meters of the same manufacturer connected to the DCU (L1, L2, L3)
   L1 and L2 meters with the same software version *FW_VER1* ,
   L3 meter with a different software version *FW_VER3,*
3. One meter from another manufacturer L4 connected to the DCU.
4. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
5. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Set-Request-Normal 40054 / 0-100: 0.131.0.255/3 (Meter firmware control / schedule) to the DCU with an array of structures: `array [structure { prio : double-long-unsigned := 100 -- wildcard matching the meter type used for performing the test name_wildcard : octet-string[13] := SAG203??????? curr_fw : data := FW_VER1 update_cmd : structure { https_url : octet-string := fw_update_url dest_fw_version : data := future contents of 1/1-0:0.2.0.255/2 } }]` 2. Using the webGUI, view the DCU's events and the update status of individual meters. 3. Wait for the FW upgrade to finish on all meters - verify the FW version on the meter using DCSAP / LCD / meter diagnostic software.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 3. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 4. The DCU establishes the association of FW upgrade in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode, 5. The DCU unregisters the unavailable meter and stops the FW upgrade process. 6. After reconnecting the meter, the process continues (block transfer continues from the last block transferred before shutdown). 7. (same result as in step (6)) 8. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update. 9. The FW upgrade process has been successfully completed - both the DCSAP and the LCD can be used to read the new software version.

DCSAP89: FW update of meters in unicast mode - the meter is temporarily unavailable

Description:

Verification of the correct implementation of the FW upgrade unicast mechanism of meters in the case of meters temporarily unavailable.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. One meter connected to DCU (L1),
3. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
4. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Awaited results:
1. Set the minimum number of meters where broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to more than 1. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 3. Start the FW upgrade procedure on L1 by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L1 with the parameter Octet-String *fw_update_url.* 4. Observe PLC traffic with the PRIME sniffer. 5. Disconnect the L1 meter from the DCU. 6. Wait for 1 minute. Connect the L1 meter to the DCU. 7. Wait 2 minutes. Do step (6) again. 8. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) of the L1 meter update. 9. Wait until the FW upgrade on the meter is completed - verify the FW version on the meter using DCSAP / LCD / meter diagnostic software.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 3. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 4. The DCU establishes the association of FW upgrade in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode, 5. The DCU unregisters the unavailable meter and stops the FW upgrade process. 6. After reconnecting the meter, the process continues (block transfer continues from the last block transferred before shutdown). 7. (same result as in step (6)) 8. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update. 9. The FW upgrade process has been successfully completed - both the DCSAP and the LCD can be used to read the new software version.

DCSAP90: FW update of meters in broadcast mode - meters temporarily unavailable

Description:

Checking the correctness of implementation of the FW upgrade broadcast mechanism of meters in the case of meters temporarily unavailable.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. At least 2 meters connected to the DCU (L1, L2) of the same type supporting broadcast FW upgrade,
3. Image package for updating meters available via URL in http (or https) - *fw_update_url*,
4. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Set the minimum number of meters for which broadcast is used - Set-Request-Normal 40054 / 0-100: 0.131.0.255/6 of Double-long-unsigned type to 2. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 and L2 meters. 3. Start the FW upgrade procedure on L1 by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L1 with the parameter Octet-String *fw_update_url.* 4. Start the FW upgrade procedure on L2 by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L2 with the Octet-String parameter *fw_update_url* (important: this step must be performed before the FW upgrade procedure on L1 is completed). 5. Observe PLC traffic with the PLC-PRIME sniffer. 6. Disconnect the L1 meter from the DCU. 7. Wait for 1 minute. Disconnect meter L2 from the DCU. 8. Wait for 1 minute. Connect the L1 and L2 meters to the DCU. 9. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) of the L1 meter update. 10. Wait until the FW upgrade on the meter is completed - verify the FW version on the meter using DCSAP / LCD / meter diagnostic software.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; the list includes an entry for the L1 and L2 meters. 3. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 4. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 5. After completing the step (3) - the DCU establishes the FW upgrade association in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode, After completing the step (4) - the DCU establishes the FW upgrade association in the unicast mode with the L2 meter to start the update process, and then start broadcasting blocks. Periodically, the DCU exchanges unicast messages with the L1, L2 meters to prevent breaking the FW Upgrade association. 6. The DCU unregisters the unavailable L1 meter, the update of the L2 meter is continued in the unicast mode. 7. The DCU unregisters the unavailable L2 meter, the update process is interrupted. 8. After reconnecting the L1 and L2 meters, the process continues in the broadcast mode (the block transfer continues from the last block sent before switching off). 9. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update. 10. The FW upgrade process has been successfully completed - both the DCSAP and the meter's LCD can be used to read the new software version.

DCSAP91: FW update of meters - pausing of the FW upgrade mechanism

Description:

Checking the correctness of the pause mechanism FW upgrade of meters.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. At least 2 meters connected to the DCU (L1, L2) of the same type supporting broadcast FW upgrade,
3. Image package for updating meters available via URL in http (or https) - *fw_update_url* ,
4. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	**Expected results:**
1. Set the minimum number of meetrs for which broadcast is used - Set-Request-Normal 40054.0-100: 0.131.0.255/6 of Double-long-unsigned type to 2. 2. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 and L2 meters. 3. Start the FW upgrade procedure on L1 by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L1 with the parameter Octet-String *fw_update_url.* 4. Start the FW upgrade procedure on L2 by sending Action-Request-Normal 40101 / 0-100: 0.0.1.255/1 to L2 with the Octet-String parameter *fw_update_url* (important: this step must be performed before the FW upgrade procedure on L1 is completed). 5. Observe PLC traffic with the PRIME sniffer. 6. Wait 2 minutes. Send the command Set-Request-Normal 40054 / 0-100: 0.131.0.255/6 (is_updater_paused) unsigned to 1 to the DCU. 7. Wait 2 minutes. Send the command Set-Request-Normal 40054 / 0-100: 0.131.0.255/6 (is_updater_paused) to the DCU of type unsigned to 0. 8. Watch the update progress by asking for the status (40101 / 0-100: 0.0.1.255/6) and the progress (40101 / 0-100: 0.0.1.255/7) of the L1 meter update. 9. Wait until the FW upgrade on the meter is completed - verify the FW version on the meter using DCSAP / LCD / meter diagnostic software.	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Get-Response-Normal response with list of meters received; the list includes an entry for the L1 and L2 meters. 3. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 4. Action-Response-Normal response received with Data-Access-Result = 0 (success) and a new *update_id.* 5. After completing the step (3) - the DCU establishes the FW upgrade association in the unicast mode with the L1 meter to start the update process, and then starts sending blocks in the unicast mode, After completing the step (4) - the DCU establishes the FW upgrade association in the unicast mode with the L2 meter to start the update process, and then start broadcasting blocks. Periodically, the DCU exchanges unicast messages with the L1, L2 meters to prevent breaking the FW Upgrade association. 6. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. The DCU stops sending image blocks for updating. The current update status and progress for each meter will remain unchanged. 7. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. The update process continues (block transfers continue from the last block transferred before pause the FW upgrade mechanism). 8. The update status changes according to the DCSAP specification. Progress increases as more blocks with an image are sent for update. 9. The FW upgrade process has been successfully completed - both the DCSAP and the LCD can be used to read the new software version.

DCSAP95: Send Emergency Commands in Broadcast Mode (1)

Description:

Verification of the correctness of the implementation of broadcast transmissions in order to handle emergency control commands.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. At least one meter connected to the DCU (L1) with a correctly configured Limiter object (71 / 0-0.17.0.1.255) - emergency profile configured,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Action-Request-Normal: 40055 / 0-100: 0.132.0.255/1 to the DCU with the emergency_profile parameter (meaning the same as for the emergency_profile attribute of the Limiter class) - a structure with 3 fields: `long-unsigned: 1octet_string: dlms-date-time w przyszłościdouble-long-unsigned: 120` 2. Observe PLC traffic with the PLC-PRIME sniffer. 3. Check the last events in the DCU's event log. 4. Verify the condition of the Limiter object on connected meters (using the meter diagnostic software)	1. Received Action-Response-Normal with DLMS success code and no additional data. 2. Using the PLC-PRIME sniffer you can observe broadcast packets with the following properties: - the LLC header of the packet should contain information about broadcasting in the pre-established association (client_id = 0x66) - in the Invoke-Id-And-Priority byte, bit 6 (service-class) should have the value set to = 0 (Unconfirmed) - the packet contains Set-Request-Normal (71 / 0-0.17.0.1.255 / 8, emergency_profile) - the data in the emergency_profile structure is consistent with the data sent in step (1) - the number of packets sent and the time between successive packets is as set by the Meter emergency control object (40055 / 0-100: 0.132.0.255) 3. An event indicating that an emergency broadcast message was sent has appeared in the DCU event log 4. The selected emergency profile is active in the meter. Activation parameters are consistent with those sent in step (1)

DCSAP96: Sending Emergency Commands in Broadcast Mode (2)

Description:

Verification of the correctness of the implementation of broadcast transmissions in order to handle emergency control commands.

Test requirements:

1. DCSAP client with session set up for the DCU,
2. At least one meter connected to the DCU (L1) with a correctly configured Limiter object (71 / 0-0.17.0.1.255) - emergency profile configured,
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Set the number of repetitions of the emergency message to 5 - send Set-Request-Normal object 40055 / 0-100: 0.132.0.255/2 - double-long-unsigned value 5. 2. Set the repetition frequency of the emergency message to 1000 milliseconds - send Set-Request-Normal object 40055 / 0-100: 0.132.0.255/3 - double-long-unsigned 1000 value. 3. Send the command Action-Request-Normal: 40055 / 0-100: 0.132.0.255/1 to the DCU with the emergency_profile parameter (meaning the same as for the emergency_profile attribute of the Limiter class) - a structure with 3 fields: `long-unsigned: 1octet_string: dlms-date-time w przyszłościdouble-long-unsigned: 120` 4. Observe PLC traffic with the PLC-PRIME sniffer. 5. Check the last events in the DCU's event log. 6. Verify the condition of the Limiter object on connected meters (using the meter diagnostic software)	1. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 2. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 3. Received Action-Response-Normal with DLMS success code and no additional data. 4. Using the PLC-PRIME sniffer you can observe broadcast packets with the following properties: - the LLC header of the packet should contain information about broadcasting in the pre-established association (client_id = 0x66) - in the Invoke-Id-And-Priority byte, bit 6 (service-class) should have the value set to = 0 (Unconfirmed) - the packet contains Set-Request-Normal (71 / 0-0.17.0.1.255 / 8, emergency_profile) - the data in the emergency_profile structure is consistent with the data sent in step (3) - the number of packets sent and the time between successive packets is as set by the Meter emergency control object (40055 / 0-100: 0.132.0.255) 5. An event indicating that an emergency broadcast message was sent has appeared in the DCU event log 6. The selected emergency profile is active in the meter. Activation parameters are consistent with those sent in step (3)

DCSAP97: Successful sending of emergency commands in broadcast mode

Description:

Verification of the effectiveness of sending emergency broadcast commands

Test requirements:

1. DCSAP client with session set up for the DCU,
2. 215 meter connected and well communicated with the DCU with a correctly configured Limiter object (71 / 0-0.17.0.1.255) - configured emergency profile,
3. Working conditions: network free from interferences from devices not meeting the requirements of electromagnetic compatibility.
4. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Galvanic connection to the DCU 200 meters. 2. Wait 12 hours. 3. Set the number of repetitions of the emergency message to 300 - send Set-Request-Normal object 40055 / 0-100: 0.132.0.255/2 - double-long-unsigned 300. 4. Set the repetition frequency of the emergency message to 2000 milliseconds - send Set-Request-Normal object 40055 / 0-100: 0.132.0.255/3 - double-long-unsigned 2000 value. 5. Send the command Action-Request-Normal: 40055 / 0-100: 0.132.0.255/1 to the DCU with the emergency_profile parameter (meaning the same as for the emergency_profile attribute of the Limiter class) - a structure with 3 fields: `long-unsigned: 1octet_string: dlms-date-time w przyszłościdouble-long-unsigned: 120` 6. Observe PLC traffic with the PLC-PRIME sniffer. 7. Check the last events in the DCU's event log. 8. Verify the state of the Limiter object on the connected meters (using the meters' diagnostic software or remotely using the DCSAP protocol). 9. Wait for emergency mode to end. 10. Make galvanic connection of the remaining meters, wait 12 hours and repeat steps 5-8.	1. (no effect) 2. The DCU has automatically detected and addressed each meter. 3. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 4. Set-Response-Normal response with Data-Access-Result = 0 (success) was received. 5. Received Action-Response-Normal with DLMS success code and no additional data. 6. Using the PLC-PRIME sniffer you can observe broadcast packets with the following properties: - the LLC header of the packet should contain information about broadcasting in the pre-established association (client_id = 0x66) - in the Invoke-Id-And-Priority byte, bit 6 (service-class) should have the value set to = 0 (Unconfirmed) - the packet contains Set-Request-Normal (71 / 0-0.17.0.1.255 / 8, emergency_profile) - the data in the emergency_profile structure is consistent with the data sent in step (5) - the number of packets sent and the time between each packet is as set by the Meter emergency control object (40055 / 0-100: 0.132.0.255) 7. An event indicating that an emergency broadcast message was sent has appeared in the DCU event log 8. The selected emergency profile is active at 95% of the pool of 200 meters. Activation parameters are consistent with those sent in step (5)

	9. (no effect) 10. The selected emergency profile is active on 85% of the pool of 215 meters. Activation parameters are consistent with those sent in step (5)

DCSAP99: Communication with the ISD module

Description:

Checking the correctness of communication with ISD

Test requirements:

1. DCSAP client with session set up for DCU,
2. At least one meter connected to the DCU (L1) with an ISD module connected, supporting the transmission of messages to the ISD.
3. PLC-PRIME sniffer for viewing of the DLMS data.

Steps:	Expected results:
1. Send the command Get-Request-Normal 40000 / 0-100: 0.0.0 * 255/2 to the DCU (reading the list of meters). Note the device_id of the L1 meter. 2. Send a message to the query queue object to the HAN module - according to the meter's COSEM model. 3. Use the diagnostic software to verify the correctness of the query to the ISD module.	1. Get-Response-Normal response with list of meters received; there is an entry for the L1 meter in the list. 2. The add to the ISD query queue operation completed successfully. 3. The state of the ISD module registers is as expected.

DCSAP100: PRIME 1.4 modem configuration - MAC backward compatibility

Description:

Checking the possibility of reconfiguring PRIME BaseNode 1.4 without the need to replace the DCU firmware.

Test requirements:

1. DCSAP client with session set up for DCU,
2. PRIME 1.4 sniffer that allows viewing PRIME data.

Steps:	Expected results:
1. Wyślij do DCU Get-Request-Normal 40051/0-100:0.128.0.255/113 (mac_bc) 2. Observe PLC traffic with the PRIME sniffer. 3. Send Set-Request-Normal 40051 / 0-100: 0.128.0.255/113 (mac_bc) to DCU with the inverse value than in step (1). 4. Restart the DCU.	1. Get-Response-Normal response received with *success* code and current MAC Backward Compatibility value 2. PRIME messages visible in the network are consistent with the set mode (1.4 BC or pure-1.4) 3. Set-Response-Normal response with *success* code received

5. Observe PLC traffic with the PLC-PRIME sniffer.	4. The device restarts correctly. 5. PRIME messages visible in the network comply with the set PRIME 1.4 BC mode

DCSAP101: PRIME 1.4 modem configuration - one communication channel

Description:

Checking the possibility of reconfiguring PRIME BaseNode 1.4 without the need to replace the DCU firmware.

Test requirements:

1. DCSAP client with session set up for DCU,
2. PLC-PRIME 1.4 FCC sniffer that allowsviewing PRIME data.

Steps:	Expected results:
1. Send command to DCU Set-Request-Normal 40051 / 0-100: 0.128.0.255/112 (band) with value double-long-unsigned 1 (channels = [1]) 2. Restart the DCU. 3. Observe PLC traffic with the PRIME sniffer. 4. Repeat steps 1-3 for the following values: 2, 4, 8, 16, 32, 64, 128 (channels = [2], channels = [3], channels = [4], channels = [5], channels = [6], channels = [7], channels = [8])	1. Set-Response-Normal response with *success* code received. 2. The device restarted successfully. 3. PRIME messages are visible in the channel selected in (1). 4. PRIME messages are visible in the channel selected in (1). *Note: to verify the communication channels, it may also be necessary to reconfigure the PRIME 1.4 sniffer.*

DCSAP102: PRIME 1.4 modem configuration - multiple communication channels

Description:

Checking the possibility of reconfiguring PRIME BaseNode 1.4 without the need to replace the DCU firmware.

Test requirements:

1. DCSAP client with session set up for DCU,
2. PLC-PRIME 1.4 FCC sniffer that allows viewing PRIME data.

Steps:	Expected results:
1. Send command to DCU Set-Request-Normal 40051 / 0-100: 0.128.0.255/112 (band) with value double-long-unsigned 3 (channels = [1,2]) 2. Restart the DCU. 3. Observe PLC traffic with the PLC-PRIME sniffer. 4. Repeat steps 1-3 for the value: 7 (channels = [1,2,3]), 15 (channels = [1,2,3,4]),	1. Set-Response-Normal response with *success* code received. 2. The device restarted successfully. 3. PRIME messages are visible in the channel selected in (1). 4. PRIME messages are visible in the channel selected in (1).

31 (channels = [1,2,3,4.5]) , 63 (channels = [1,2,3,4,5,6]), 127 (channels = [1,2,3,4,5,6,7]), 255 (channels = [1,2, 3,4,5,6,7,8]), a random value from the range <1; 255>	*Note: to verify the communication channels, it may also be necessary to reconfigure the PRIME 1.4 sniffer.*

4. Event list

ZDA01: Data concentrator event list

Description:

The purpose of the test is to verify the event data are stored correctly by the DCU.

Test requirements:

1. DCSAP client with an active DCU session connection,
2. Concentrator event mask configured in such a way that the device records tested events (40001 / 0-100: 0.0.3 * 255/5),
3. The moments of changing the concentrator's time during the generation of recorded events must be known.

Steps:	Expected results:
1. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2 to the DCU. 2. Take note of the largest value of the seq_id = MAX_SEQ_ID field. 3. Send the Get-Request-With-List command to the DCU for the 40001 / 0-100: 0.0.3 * 255 object. Attributes 3-4. 4. Send the Action-Request-Normal command 40001 / 0-100: 0.0.3 * 255/1 to the DCU with the parameter - the event_list_entry structure containing the fields: `Long64-Unsigned = dowolny, Double-Long-Unsigned = dowolny, Double-Long-Unsigned = 0, Unsigned = 0, Integer = -1, Structure { Octet-String = 'key', Unsigned = 1 }, Octet-String = 'zdarzenie testowe'` 5. Send to the DCU the Get-Request-Normal command: 40001 / 0-100: 0.0.3	1. Get-Response-Normal response received with data: event list; the entries have a structure compliant with the DCSAP specification; successive event entries have seq_id fields differing by 1, ascending, subsequent event entries have non-decreasing time stamps (except for the first events following the change of the DCU's time); 2. There is an event on the list that meets the criteria: device_id = 0, time > TM, reason = 1 (EV_RESTART); 3. Get-Response-With-List response received with 3 parts with DLMS data: - Part 2 of type Double-Long-Unsigned value = number of entries in the table received in step 1 - Part 3 of type Double-Long-Unsigned value >= values in part 2. 4. An Action-Response-Normal response was received with the code Action-Result = 0 (Success). 5. Get-Response-Normal response received with data: event list; There is exactly one event in the list; The event structure is identical to the parameter sent in step 3, except for the fields: reason (4) = 255 (EV_PUSH), seq_id (1) = MAX_SEQ_ID + 1.

Steps:	Expected results:
* 255/2, access-selector 1 with the parameter of the type of Long64-Unsigned = MAX_SEQ_ID.	

ZDA02: First registration of the meter in the DCU

Description:

The purpose of the test is to check whether the DCU registers the event of the first registration of the meter in a concentrator.

Test requirements:

1. DCSAP client with an active DCU session connection,
2. Concentrator event mask configured in such a way that the device records tested events (40001 / 0-100: 0.0.3 * 255/5),
3. Communal meter (L1) not connected and not registered in the DCU.

Steps:	Expected results:
1. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2 to the DCU. Take note of the largest value of the seq_id = MAX_SEQ_ID field. 2. Connect the L1 meter to the network. Wait TD = 2 minutes. 3. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2, access-selector 1 with the parameter of the type of Long64-Unsigned = MAX_SEQ_ID to the DCU.	1. Received a list of events previously registered by a concentrator. 2. The L1 meter has connected to the concentrator. 3. At least one new event has been registered with the source DCU (DCSAP device_id = 0) with the following values: field reason = 2 (EV_METERSTAT); status field = 1; coment field = name of the connected meter

ZDA03: 'connection' event registration

Description:

Checks if the DCU logs the DCSAP session connection event.

Test requirements:

1. DCU connected to the network,
2. DCSAP client,
3. Concentrator event mask configured in such a way that the device records tested events (40001 / 0-100: 0.0.3 * 255/5)

Steps:	Expected results:
1. Create a new DCSAP session, send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2 to the DCU.	1. Received a list of events previously registered by the DCU.

Steps:	Expected results:

2. Take note of the largest value of the seq_id = MAX_SEQ_ID field.
3. Create a new DCSAP session, send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2, access-selector 1 with the parameter of the type of Long64-Unsigned = MAX_SEQ_ID to DCU.
	2. At least one new event has been registered with DCU as a source (DCSAP device_id = 0) with the following values:
	field reason = 160 ('connection');
	status field = 1;
	comment field = ip address of remote dcsap client.

ZDA04: 'login' event registration

Description:

Verifies that the DCU is logging a web session authentication event.

Test requirements:

1. DCSAP client with an active DCU session connection,
2. The concentrator event mask is configured in such a way that the device records the tested events (40001 / 0-100: 0.0.3 * 255/5).

Steps:	Expected results:

1. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2 to the DCU.
Take note of the largest value of the seq_id = MAX_SEQ_ID field.
2. In a web browser, open the website login window.
3. Log in with the wrong username or password.
4. Log in with a valid username and password.
5. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2, access-selector 1 to the DCU, with the parameter of the type of Long64-Unsigned = MAX_SEQ_ID.
6. Verify the content of the event list using webGUI.
7. Via the web interface, check the current values of the number of login attempts causing the account blocking (x) and the length of the blocking time (y)
8. Try logging in x times with the wrong password
9. Wait y minutes and try to log in again with the correct password
10. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2, access-selector 1 to the DCU, with the parameter of the type of Long64-Unsigned = MAX_SEQ_ID.
11. Verify the content of the event list using webGUI.
	1. Received a list of events previously registered by a concentrator.
	2. WebGUI login page accessed correctly
	3. One new event was registered with the source being the DCU (DCSAP device_id = 0) with the following values:
	field reason = 161 ('login');
	status field = 0;
	comment field = the ip address of the remote dcsap client and the name of the user trying to log into the web page.
	4. One new event was registered with the source DCU (DCSAP device_id = 0) with the following values:
	field reason = 161 ('login');
	status field = 1;
	comment field = the ip address of the remote dcsap client and the name of the user used to access the web page.
	5. The content of the list of events downloaded by DCSAP is consistent with 3-4
	6. The content of the event list available on the webGUI via DCSAP is consistent with 3-4
	7. Parameters are available in the webGUI

Steps:	Expected results:

	8. After completing step 8, the account is locked
	9. After completing step 9 - correctly logging into the web interface
	10. X new events were registered with the source DCU (DCSAP device_id = 0) with the following values:
	field reason = 161 ('login');
	status field = 0 or -1 (blocked);
	comment field = the ip address of the remote dcsap client and the name of the login user.
	11. (the content of the event list available on the webGUI is consistent with the list available via DCSAP)

ZDA05: 'tamper' event registration

Description:

Verifies that the DCU logs the DCU case opening event.

Test requirements:

1. DCSAP client with an active DCU session connection,
2. The concentrator event mask is configured in such a way that the device records the tested events (40001 / 0-100: 0.0.3 * 255/5).

Steps:	Expected results:

1. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2 to the DCU.
Take note of the largest value of the seq_id = MAX_SEQ_ID field.
2. Remove the bottom (connectors) cover
3. Open the top cover.
4. Close the top cover.
5. Replace the bottom (connectors) cover
6. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2, access-selector 1 to the DCU, with the parameter type Long64-Unsigned = MAX_SEQ_ID.
	1. Received a list of events previously registered by the DCU.
	2. At least two new events have been registered with the source DCU (DCSAP device_id = 0) with the following values:
	field reason = 176 ('tamper');
	status field = 1;
	comment field = name of the cover detector (at least one 'top' and at least one 'bottom')
	3. At least one new event has been registered with the source DCU (DCSAP device_id = 0) with the following values:
	field reason = 176 ('tamper');
	status field = 0;
	comment field = name of cover detector (at least one 'top' and at least one 'bottom')

ZDA06: Event list filtering

Description:

Verification of the correct handling of the access selector type 2 for the Event log class object.

Test requirements:

1. DCSAP client with an active DCU session connection,
2. Several events of various types are registered in the list of concentrator events - from different devices.

Steps:	Expected results:

1. Send the Get-Request-Normal command: 40001 / 0-100: 0.0.3 * 255/2 to the DCU.
Take note of the largest value of the seq_id = MAX_SEQ_ID field.
2. Send the Get-Request-Normal command to DCU: 40001 / 0-100: 0.0.3 * 255/2, access-selector 2, in the field structure
`is_backward` = true,
`max_entries = 1`
3. Send the Get-Request-Normal command to the DCU: 40001 / 0-100: 0.0.3 * 255/2, access-selector 2, in the field structure
`is_backward` = true
`max_entries = 25,`
`first_seq_id = MAX_SEQ_ID - 25`,
4. Send the Get-Request-Normal command to the DCU: 40001 / 0-100: 0.0.3 * 255/2, access-selector 2, in the field structure
`is_backward` = true
`max_entries = 25,`
`first_seq_id = MAX_SEQ_ID - 25,`
`device_id = 0,`
5. Send the Get-Request-Normal command to the DCU: 40001 / 0-100: 0.0.3 * 255/2, access-selector 2, in the field structure
`is_backward` = false
`max_entries = 25,`
`first_seq_id = 0`,
`device_id = 0,event_reason = -1`
6. Send the Get-Request-Normal command to the DCU: 40001 / 0-100: 0.0.3 * 255/2, access-selector 2, in the field structure
`is_backward` = false
`max_entries = 10,`
`first_seq_id = 0,`
`device_id = -1,`
`event_reason = 2 (EV_UPDATEINI)`
7. Send the Get-Request-Normal command to the DCU: 40001 / 0-100: 0.0.3 * 255/2, access-selector 2, in the field structure
`is_backward` = true
`max_entries = 25,`
`first_seq_id = MAX_SEQ_ID`,
`device_id = X, (gdzie X to device_id`
	1. A list of events previously registered by the DCU is received.
	2. One most recent event received (content matches (1))
	3. 25 older events were received, out of the 50 most recent.
	The events are sorted in descending order (the sort key is seq_id)
	4. 25 or less older events were received, out of the 50 most recent.
	The events are sorted in descending order (the sort key is seq_id).
	DCU only events (device_id = 0).
	5. Received 25 or less events since the beginning of the event log.
	The events are sorted in ascending order (the sort key is seq_id).
	DCU only events (device_id = 0).
	6. Received 10 or less events since the beginning of the event log.
	The events are sorted in ascending order (the sort key is seq_id).
	Only the start events of the FW upgrade (EV_UPDATEINI) are present.
	Events relate to the DCU or other meters.
	7. Received 25 or less events since the beginning of the event log.
	The events are sorted in descending order (the sort key is seq_id).
	Events for the selected meter only.

Steps:	Expected results:
jakiegoś licznika) event_reason = -1,	

ZDA10: DCU asynchronous event reporting (1)

Description:

Checks if the DCU reports the DCSAP session connection event.

Test requirements:

1. DCSAP client with an active DCU session connection,
2. Concentrator event mask configured in such a way that the device records tested events and sends notifications to them (40001 / 0-100: 0.0.3 * 255/5).

Steps:	Expected results:

1. Sign up to receive DCSAP notifications by sending Set-Request-Normal command 1 / 0-100: 32.0.1 * 255/2 with boolean (true) attribute.
2. In parallel, open a new DCSAP session and then close it.
	1. In the first DCSAP session, at least the following notifications were received:
	a. Event with DCU source (DCSAP device_id = 0) with values:
	field reason = 160 ('connection');
	status field = 1;
	comment field = ip address of remote dcsap client
	b. Event with DCU source (DCSAP device_id = 0) with values:
	field reason = 160 ('connection');
	status field = 0;
	comment field = ip address of remote dcsap client

ZDA11: DCU asynchronous event reporting (2)

Description:

Tests if the DCU reports a 'tamper' event.

Test requirements:

1. DCSAP client with an active DCU session connection,
2. Concentrator event mask configured in such a way that the device records tested events and sends notifications to them (40001 / 0-100: 0.0.3 * 255/5).

Steps:	Expected results:
1. Sign up to receive DCSAP notifications by sending Set-Request-Normal command 1 / 0-100: 32.0.1 * 255/2 with boolean (true) attribute.	1. Set-Response-Normal received with code *success* .

Steps:	Expected results:
2. Perform the ZDA05 test	2. In the DCSAP session, notifications were received in accordance with the data from the concentrator's event log

ZDA12: Asynchronous meter events reporting (1)

Description:

Checks if the DCU reports meter events correctly.

Test requirements:

1. DCSAP client,
2. L1 meter connected to the DCU with an event mask configured in such a way that the device records the tested events and sends notifications to them,
3. PRIME sniffer that allows you to view the transmitted DLMS data.

Steps:	Expected results:
1. Set up a DCSAP D1 session; register to receive DCSAP notifications by sending Set-Request-Normal command 1 / 0-100: 32.0.1 * 255/2 with boolean attribute (true). 2. Set up a DCSAP D2 session; register to receive DCSAP notifications by sending Set-Request-Normal command 1 / 0-100: 32.0.1 * 255/2 with boolean attribute (true). 3. Set the DCSAP D3 session (receiving notifications will be disabled in this session) 4. Perform an operation on the L1 meter that should generate a DLMS notification (e.g. opening the terminal cover) and observe the PLC communication with the PLC-PRIME sniffer.	1. Set-Response-Normal received with code *success* . 2. Set-Response-Normal received with code *success* . 3. Correctly opened DCSAP session. 4. The DLMS notification sent by the meter has been registered on the PRIME sniffer a. DCSAP notification received in DCSAP D1 and D2 sessions: notification is marked with *dev_id* corresponding to the L1 meter on the list of meters the notification contains the same data as the DLMS notification registered by the sniffer b. no notification was received in the DCSAP D3 session

ZDA13: Asynchronous meter events reporting (2)

Description:

Checks if the DCU reports meter events correctly.

Test requirements:

1. DCSAP client,
2. The L1 meter connected to the DCU with an event mask configured in such a way that the device records the tested events and sends notifications in the MGMT association,
3. DCSAP64 test performed (MGMT association encryption and signing enabled),

4. PRIME sniffer that allows viewing of the transmitted DLMS data.

Steps:	Expected results:
1. Set up a DCSAP D1 session; register to receive DCSAP notifications by sending Set-Request-Normal command 1 / 0-100: 32.0.1 * 255/2 with boolean attribute (true). 2. Set up a DCSAP D2 session; register to receive DCSAP notifications by sending Set-Request-Normal command 1 / 0-100: 32.0.1 * 255/2 with boolean attribute (true). 3. Set the DCSAP D3 session (receiving notifications will be disabled in this session) 4. Send the Get-Request-Normal command to the L1 meter with the request for the clock value (8 / 0-0: 1.0.0 * 255/2) and at the same time observe the communication on the PLC-PRIME sniffer. 5. On the L1 meter perform an operation that should generate a DLMS notification (e.g. opening the terminal cover) available in the MGMT association and observe the PLC communication with the PLC-PRIME sniffer.	1. Set-Response-Normal received with code *success* . 2. Set-Response-Normal received with code *success* . 3. Correctly opened DCSAP session. 4. Clock value successfully retrieved. The traffic viewed on the sniffer is encrypted and signed. 5. The DLMS notification sent by the meter was registered on the PLC-PRIME sniffer - encrypted and signed (notification content unknown) a. in DCSAP D1 and D2 sessions DCSAP notification was received: notification is marked with *dev_id* corresponding to the L1 meter on the meter list. The notification contains decrypted data provided by the meter. The content of the notification is consistent with the entry in the appropriate log of the meter events (to be verified by the use of meters diagnostic software) b. no notification was received in the DCSAP D3 session

5. Firmware update

UPG01: DCU HTTPS update - invalid certificate

Description:

Validation of DCU behaviour with invalid SSL certificate.

Test requirements:

1. DCU running and connected to the network,
2. Serwer HTTP.

Steps:	Expected results:
1. Start the https server (IP: PORT) with the server ssl certificate not compatible with the one expected by the DCU (e.g. self-signed).	1. (https server running correctly).

Steps:	Expected results:
2. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'https://IP:PORT/fw.tgz'. 3. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 4. Check the last events in the DCU's event log.	2. The Action-Request-Normal command returns the result: DLMS OK. 3. Get-Request-Normal command returns the result: DLMS OK, data: s32 EWRONGCERT - 3. 4. In the DCU's event log there are EV_UPDATEINIT and EV_UPDATEFINI events with parameters consistent with the results 2-3.

UPG02: DCU HTTPS update - invalid url

Description:

Checks if the DCU behaviour when providing an incorrect https address is correct

Test requirements:

1. DCU running and connected to the network,
2. HTTPS server with a SSL server certificate compliant with the one expected by DCU (provided by the DCU manufacturer).

Steps:	Expected results:
1. Start the https server (IP: PORT), giving it a ssl server certificate that matches the one expected by the DCU. 2. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'https://IP:PORT/fw.tgz'. 3. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 4. Check the last events in the DCU's event log.	1. (https server running correctly). 2. The Action-Request-Normal command returns the result: DLMS OK 3. Get-Request-Normal returns the result: DLMS OK, data: s32 EINVURL -4 4. In the DCU's event log there are EV_UPDATEINIT and EV_UPDATEFINI events with parameters consistent with the results 2-3.

UPG03: DCU HTTPS update - incorrect firmware file

Description:

Checks if the DCU behaviour when providing an incorrect firmware file is correct.

Test requirements:

1. DCU running and connected to the network,
2. HTTPS server with a SSL server certificate compliant with the one expected by DCU (provided by the DCU manufacturer).

Steps:	Expected results:
1. Start the https server (IP: PORT), giving it a server ssl certificate that matches the one expected by the DCU. 2. Place the zeros.tgz file on the server, filled with zeros, 1 MB in size. 3. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'https://IP:PORT/zeros.tgz'. 4. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 5. Check the last events in the DCU's event log.	1. (https server running correctly). 2. (file on the server, available after https). 3. Action-Request-Normal returns the result: *success* and double-long-unsigned *update_id* 4. Get-Request-Normal returns result: *success* , data: s32 EINVCHKSUM -5 or EFWINVALID -7 5. The EV_UPDATEINIT and EV_UPDATEFINI events with parameters compatible with the results 3-4 are present in the DCU's event log.

UPG04: DCU HTTPS update - firmware file mismatch

Description:

Checks if the DCU behaviour when providing mismatched firmware (trying to update 1-phase meter with 3-phase meter firmware of the same supplier) file is correct.

Test requirements:

1. DCU running and connected to the network,
2. HTTPS server with a SSL server certificate compliant with the one expected by DCU (provided by the DCU manufacturer).

Steps:	Expected results:
1. Start the https server (IP: PORT), providing a ssl server certificate that matches the one expected by the DCU. 2. Place the incorrect-dcu-model.tgz file on the server, with the software for a different DCU model than the tested one (package provided by the manufacturer). 3. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'https://IP:PORT/incorrect-dcu-model.tgz'. 4. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 5. Check the last events in the DCU's event log.	1. (https server running correctly). 2. (file on the server, available after https). 3. Action-Request-Normal returns the result: *success* and double-long-unsigned *update_id* 4. Get-Request-Normal returns result: *success* , data: s32 EFWINVALID -7 5. The EV_UPDATEINIT and EV_UPDATEFINI events with parameters compatible with the results 3-4 are present in the DCU's event log.

UPG05: DCU firmware update - update aborted

Description:

Checks if the DCU behaviour when update was aborted is correct.

Test requirements:

1. DCU running and connected to the network,
2. A program that listens for TCP connections on the PORT (eg 'nc -l $ PORT') is avaialble.

Steps:	Expected results:
1. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'https://IP:PORT/fw.tgz'. 2. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 3. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'https://IP:PORT/fw.tgz'. 4. Send the Action-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/2 (dcu_firmware / abort_update) to the DCU with the update_id parameter received in (1). 5. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 6. Check the last events in the DCU's event log.	1. Action-Request-Normal returns the result: *success* and double-long-unsigned *update_id* . 2. Get-Request-Normal returns result: *success* , data: s32 CERTVERIF 1 3. Action-Request-Normal returns the result: dlms-error TEMPORARY_FAILURE 4. Action-Request-Normal returns the result: *success* 5. Get-Request-Normal returns result: *success* , data: s32 EFWABORT -1 6. In the DCU's event log there are EV_UPDATEINIT and EV_UPDATEFINI events with parameters compliant with the results 1-5

UPG06: DCU HTTPS software update - correct update process

Description:

Verification of the whole DCU firmware update process.

Test requirements:

1. DCU running and connected to the network,
2. HTTPS server with a SSL server certificate compliant with the one expected by DCU (provided by the DCU manufacturer).
3. The file DCUFW.tgz is available on the server, with the correct software for the tested DCU model (package provided by the manufacturer).

Steps:	Expected results:
1. Uruchom serwer https (IP:PORT), podając mu serwerowy certyfikat ssl zgodny z oczekiwanym przez ZKB. 2. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'https://IP:PORT/DCUFW.tgz'. 3. Send the Get-Request-Normal command: 40101/0-100:0.0.1*255/6 (dcu_firmware/last_update_status). 4. Wait for the DCU reboot to be performed	1. (http server running correctly). 2. Action-Request-Normal returns the result: *success* and double-long-unsigned *update_id* . 3. The Get-Request-Normal command returns the result: *success* , data: s32 from 1 to 7, i.e. update in progress. 4. (DCSAP connection can be re-established)

Steps:	Expected results:
5. Send Get-Request-Normal command: 40101/0-100:0.0.1*255/6 (dcu_firmware/last_update_status). 6. Send Get-Request-Normal command: 40101/0-100:0.0.1*255/2 (dcu_firmware/version). 7. Check the last events in the DCU's event log.	5. The Get-Request-Normal command after restarting the device returns the result: *success* , data: s32 SUCCESS 0. 6. The FW version of the concentrator is as expected 7. In the DCU's event log there are EV_UPDATEINIT, EV_UPDATEFINI and EV_START events with parameters consistent with the results 1-4

UPG07: DCU HTTP update - invalid url

Description:

Checking the correctness of the DCU behaviour when entering an incorrect http address.

Test requirements:

1. DCU running and connected to the network,
2. HTTP server available.

Steps:	Expected results:
1. Start the http server (IP: PORT). 2. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'http://IP:xxx/fw.tgz'. 3. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 4. Check the last events in the DCU's event log.	1. (http server running correctly). 2. The Action-Request-Normal command returns the result: DLMS OK 3. Get-Request-Normal returns the result: DLMS OK, data: s32 EINVURL -4 4. In the DCU's event log there are EV_UPDATEINIT and EV_UPDATEFINI events with parameters consistent with the results 2-3.

UPG08: DCU HTTP update - correct update process

Description:

Verification of the whole DCU firmware update process.

Test requirements:

1. DCU running and connected to the network,
2. HTTP server available.
3. The file DCUFW.tgz is available on the server, with the correct software for the tested DCU model (package provided by the manufacturer).

Steps:	Expected results:
1. Start the http server (IP: PORT). 2. Send the Action-Request-Normal command: 40101/0-100:0.0.1*255/1 (dcu_firmware/start_update), with parameter octet-string 'http://IP:PORT/DCUFW.tgz'. 3. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 4. Wait for the DCU to reboot. 5. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/6 (dcu_firmware / last_update_status) to the DCU. 6. Send the Get-Request-Normal command: 40101 / 0-100: 0.0.1 * 255/2 (dcu_firmware / version) to DCU. 7. Check the last events in the DCU's event log.	1. (http server running correctly). 2. Action-Request-Normal returns the result: *success* and double-long-unsigned *update_id* **.** 3. The Get-Request-Normal command returns the result: *success* , data: s32 from 1 to 7, i.e. update in progress. 4. (DCSAP connection can be re-established) 5. The Get-Request-Normal command after restarting the device returns the result: *success* , data: s32 SUCCESS 0. 6. The FW version of the concentrator is as expected 7. In the DCU's event log there are EV_UPDATEINIT, EV_UPDATEFINI and EV_START events with parameters consistent with the results 1-4