Data Concentrator
Simple Acquisition Protocol

Wersja: 2.0.2

Data: 2016-07-15

Wyłącznie do użytku wewnętrznego w projekcie.

1 Historia zmian dokumentu

Projekt	Aplikacja AMI		
Tytuł dokumentu	Data Concentrator Simple Acquisition Protocol		
Wersja	2.0.2	**Data**	2016-07-15
Wykonanie	**Przegląd**	**Zatwierdzenie**	
Rafał Jurkiewicz Michał Mirosław Przemysław Pawełczyk Paweł Pisarczyk Dominik Bocheński	Dominik Bocheński	Rafał Jurkiewicz	

Tab. 1: Metryka dokumentu

Wersja	Data	Opis	Wykonanie
0.9	2013-01-16	Stworzenie pierwszej wersji dokumentu	Rafał Jurkiewicz Michał Mirosław Przemysław Pawełczyk Paweł Pisarczyk Dominik Bocheński
0.9.1	2013-02-10	Rozszerzenie słownika o terminy HTTPS i URL. Poprawienie rysunku sesji DCSAP by używał poprawnych kodów OBIS obiektów. Zmiana typu pola *dlms-size* w nagłówku komunikatu. DCSAP z Unsigned32 na INTEGER. Zdefiniowanie kodów błędów na poziomie nagłówka komunikatów DCSAP. Poprawienie literówek w kodowaniu przykładowych poleceń i odpowiedzi. Opisanie persystentności obiektów.	Przemysław Pawełczyk

		Dodanie opisów wszystkich atrybutów wprowadzanych klas. Rozbicie identyfikatora sieciowego licznika w liście liczników na identyfikator producenta licznika oraz nazw nadanych przez producenta – taki sam schemat jaki jest stosowany w bezpośredniej komunikacji DLMS z licznikami [4]. Doprecyzowanie procesu aktualizacji oprogramowania i wprowadzenie wymogu użycia protokołu HTTPS. Zdefiniowanie klasy *Device run information* i globalnego obiektu koncentratora będącego jego instancja. Podanie rekomendowanego rozmiaru listy liczników. Podanie rekomendacji nt. synchronizacji czasu – użycie tych samych serwerów czasu co Aplikacja AMI. Lepszy układ i konsekwencja w formatowaniu tekstu, Zwiększające czytelność i ułatwiające nawigacje po dokumencie.	

0.9.2	2013-02-19	Wyjaśnienie zmiany kodowania z BER na A-XDR. 4, bibl. Ograniczenie błędów na poziomie nagłówka komunikatów. Zmiana nazwy pola opisującego dostępność licznika na liście liczników. Opisanie kodów statusowych aktualizacji. Dodanie klas *Event list* i *String list* oraz obiektów z nich korzystających. Zmiana aktualizacji oprogramowania na odbywająca się w pełni asynchronicznie, z możliwością jej anulowania. Rozszerzenie klasy *Device firmware* i dostosowanie opisów atrybutów oraz metod.	Przemysław Pawełczyk

		Zdefiniowanie czasu co jaki ma być wysyłany pusty komunikat od MDG oraz czas po jakim sesja uznawana jest za bezczynna w DCU. Komentarz nt. czasu i nr sekwencyjnego ostatniej zmiany rekordu przy odczycie topologii. Uzupełnienie opisu odbierania zdarzeń z układu, z ustępem nt. mechanizmu powiadomień. Dodanie kilku diagramów sekwencji. Poprawienie rekomendacji nt. liczby równolegle obsługiwanych sesji i przepływności pojedynczej sesji.	
0.9.3	2013-02-20	Uproszczenie deskryptora wyboru selektywnego z klasy *Meter list*. Opisanie kodów zwracanych przez metodę *restart()* klasy *Device run information*. Przemianowanie kodu statusowego aktualizacji *EFWUPDATE* na *EMAINTAIN* i rozszerzenie jego znaczenia. Uzupełnienie opisu metody *start_update()* o natychmiastowe anulowanie zlecenia aktualizacji oprogramowania jeżeli rozpoczęto uprzednio procedurę restartu urządzenia. Zmiana typu pól statusowych w klasie *Device run information* na *integer*. Opisanie obiektów *Data concentrator event list* i *Data concentrator NTP server list*. Opisanie zdarzeń zatrzaskiwanych w dzienniku zdarzeń koncentratora. Usuniecie zbędnego obiektu *Data concentrator configuration id*. Poprawienie kilku diagramów sekwencji i dodanie nowych. Opisanie implementacji referencyjnej.	Przemysław Pawełczyk Rafał Jurkiewicz
0.9.4	2013-02-25	Opisanie wspólnych cech obiektów oraz wprowadzenie reguł numerowania nowych klas i obiektów. Uproszczenie rekordów klasy *Meter list* i	Przemysław Pawełczyk

		wprowadzenie klasy *Device basic information* oraz obiektu *Meter information*. Podkreślenie atomowości i kompletności zmian zachodzących w obiekcie *Data concentrator meter list*. Poprawienie i dodanie kilku diagramów sekwencji.	
0.9.5	2013-02-26	Przeformatowanie i uszczegółowienie historii zmian. Zmiana typu pola *dlms-data* w nagłówku komunikatu DCSAP z *INTEGER* na *Integer32*. Poprawienie kardynalności klas interfejsów, która wg standardu odnosi się do logicznych urządzeń. Dodanie tabeli wymieniającej wszystkie klasy interfejsów wprowadzone na potrzeby protokołu DCSAP oraz ich kardynalności z punktu widzenia DCU oraz licznika.	Przemysław Pawełczyk
0.9.6	2013-02-27	Doprecyzowanie sposobu odwoływania się do obiektów liczników realizowanych przez koncentrator. Dodanie klasy *DCSAP network statistics*. Dodanie obiektu *Data concentrator network statistics*. Zmiana używanych typów całkowitoliczbowych w implementacji referencyjnej na typedef-owane (w miejscach gdzie oczekiwany jest konkretny rozmiar przekazywanych liczb), co zapewnia lepszą przenośność kodu oraz podnosi jego czytelność.	Przemysław Pawełczyk Rafał Jurkiewicz
0.9.7	2013-02-27	Podkreślenie obowiązkowości implementacji mechanizmu notyfikacji. Uproszczenie i drobna zmiana reguł doboru kodów OBIS. Przenumerowanie istniejących obiektów. Rozszerzenie struktury *event_list_entry* o numer sekwencyjny zdarzenia i dodanie obsługi wyboru selektywnego z atrybutu *event_log* klasy *Event list*. Zdefiniowanie zachowania obiektu *Data concentrator event list* w przypadku osiągnięcia maksymalnej liczby rekordów.	Przemysław Pawełczyk Rafał Jurkiewicz

		Podanie typów atrybutów *value* obiektów *Data concentrator meter data caching enable* i *Data concentrator asynchronous notification enable* oraz ich wartości domyślnych. Poprawienie sekwencji komunikatów wysyłanych przy rozpoczęciu sesji. Usuniecie rekomendacji nt. bufora notyfikacji, ponieważ ulotność powiadomień (jak i odpowiedzi) nie jest nigdzie dozwolona, a sugerowanie konkretnej implementacji gwarancji dostarczenia notyfikacji (oraz jej parametrów) jest zagadnieniem zbyt specyficznym do rozpatrywania w tym dokumencie. Usuniecie rekomendacji nt. adresowania układów pomiarowych, ponieważ sposób nadawania przez koncentrator liczbowych identyfikatorów urządzeniom (używanych przez system akwizycji w adresacji komunikatów protokołu DCSAP) jest detalem implementacyjnym, który w żaden sposób nie wpływa na przebieg oraz jakość komunikacji miedzy systemem akwizycji a tymi urządzeniami.	
0.9.8	2013.03.04	Aktualizacja rozdziału „Obiekt *Data concentrator meter data caching enable*”	Rafał Jurkiewicz
1.0	2013-03-05	Aktualizacja rozdziału „Opis implementacji referencyjnej”. Poprawki edytorskie.	Rafał Jurkiewicz
1.0.3	2013-05-14	Poprawienie diagramu sekwencji dla 4 przypadku niepowodzenia aktualizacji oprogramowania koncentratora. Aktualizacja i dodanie rekomendacji dotyczących rozmiaru listy liczników, dziennika zdarzeń oraz listy serwerów NTP. Podkreślenie opcjonalności weryfikacji obrazu oprogramowania w procesie jego aktualizacji w licznikach.	Rafał Jurkiewicz
2.0	2014-05-12	Przeprojektowanie schematu obiektów i klas realizowanych przez koncentrator. Zmiana struktury nagłówka i kodów błędów	Michał Mirosław

		protokołu DCSAP. Doprecyzowanie relacji licznik – koncentrator dla przypadku ZKB. Zmiana SAK -> MDG.	
1.0.4	2014-10-17	Doprecyzowanie wymagań dla koncentratorów.	Tomasz Piasecki Michał Mirosław
2.0.1	2016-07-13	Konsolidacja zmian i poprawek linii specyfikacji 1.x i 2.x, uzupełnienie specyfikacji o zagadnienia zabezpieczonej komunikacji	Tomasz Piasecki
2.0.2	2016-07-15	Przegląd i uspójnienie dokumentu. Ujednolicenie stosowanych szablonów i korekta edycyjna dokumentu.	Tomasz Piasecki Rafał Jurkiewicz Michał Popiołkiewicz Bartosz Nowak

Tab. 2: Historia zmian dokumentu

Spis treści

Spis tabel

Spis rysunków

1 Słownik pojęć

Termin/skrót	Objaśnienie
AES	*Advanced Encryption Standard* – symetryczny szyfr blokowy przyjęty przez National Institute of Standard and Technology.
AMI	*Advanced Metering Infrastructure* (Zaawansowana Infrastruktura Pomiarowa) – kompleksowy system liczników, systemów komunikacyjnych i aplikacji do gromadzenia, przechowywania i analizowania danych pomiarowych oraz zarządzania infrastrukturą pomiarową.
ASN.1	*Abstract Syntax Notation One* – standard służący do opisu struktur przeznaczonych do reprezentacji, kodowania, transmisji i dekodowania danych.
A-XDR	*Adapted Extended Data Representation* – metoda serializacji danych.
COSEM	*COmpanion Specification for Energy Metering* – zbiór specyfikacji opracowanych przez DLMS UA definiujący model informatyczny obiektów m.in. liczników energii elektrycznej.
DCU	*Data Concentrator Unit* (koncentrator) – element infrastruktury licznikowej, prowadzi m.in. lokalną akwizycję danych z liczników w swoim obszarze, urządzenie ma za zadanie automatycznie wykrywać i adresować liczniki.
DLMS	*Device Language Message Specification* – zorientowany połączeniowo protokół warstwy aplikacji przeznaczony m.in. do dwukierunkowej wymiany danych z licznikami energii elektrycznej.
idempotentność	Własność operacji, która pozwala na jej powtórne wykonanie bez zmiany efektu.
MDG	*Metering Data Gateway* – część systemu AMI odpowiedzialna za pozyskiwanie odczytów z liczników energii elektrycznej poprzez koncentratory danych.
OBIS	*OBject Identification System* – system kodowania obiektów modelu COSEM.
PDU	*Protocol Data Unit* – jednostka danych w protokole.
PRIME	*PoweRline Intelligent Metering Evolution* – otwarta specyfikacja definiująca łączność w najniższych warstwach systemu komunikacyjnego po PLC od urządzeń końcowych (liczników) do koncentratora danych umieszczonego w stacji transformatorowej SN/nn.
SSL	*Secure Socket Layer* – protokół mający na celu zapewnienie poufności i integralności transmisji danych oraz zapewnienie uwierzytelnienia, opiera się na szyfrach asymetrycznych oraz certyfikatach standardu X.509.
TCP/IP	*Transmission Control Protocol/Internet Protocol* – zestaw protokołów warstwy transportowej (TCP) oraz sieciowej (IP), zapewniający ujednolicony sposób

	przesyłania danych w różnych typach sieci.
TLS	*Transport Layer Security* – protokół zastępujący SSL
URL	*Uniform Resource Locator* – format adresowania zasobów w sieci.

Przedmowa do wersji 2.0.2

Wersja protokołu DCSAP z linii 2.x została zaproponowana po dłuższym czasie użytkowania i testowania szeregu implementacji powstałych w oparciu o specyfikacje w wersji 1.x i doświadczeń zebranych na tej podstawie.

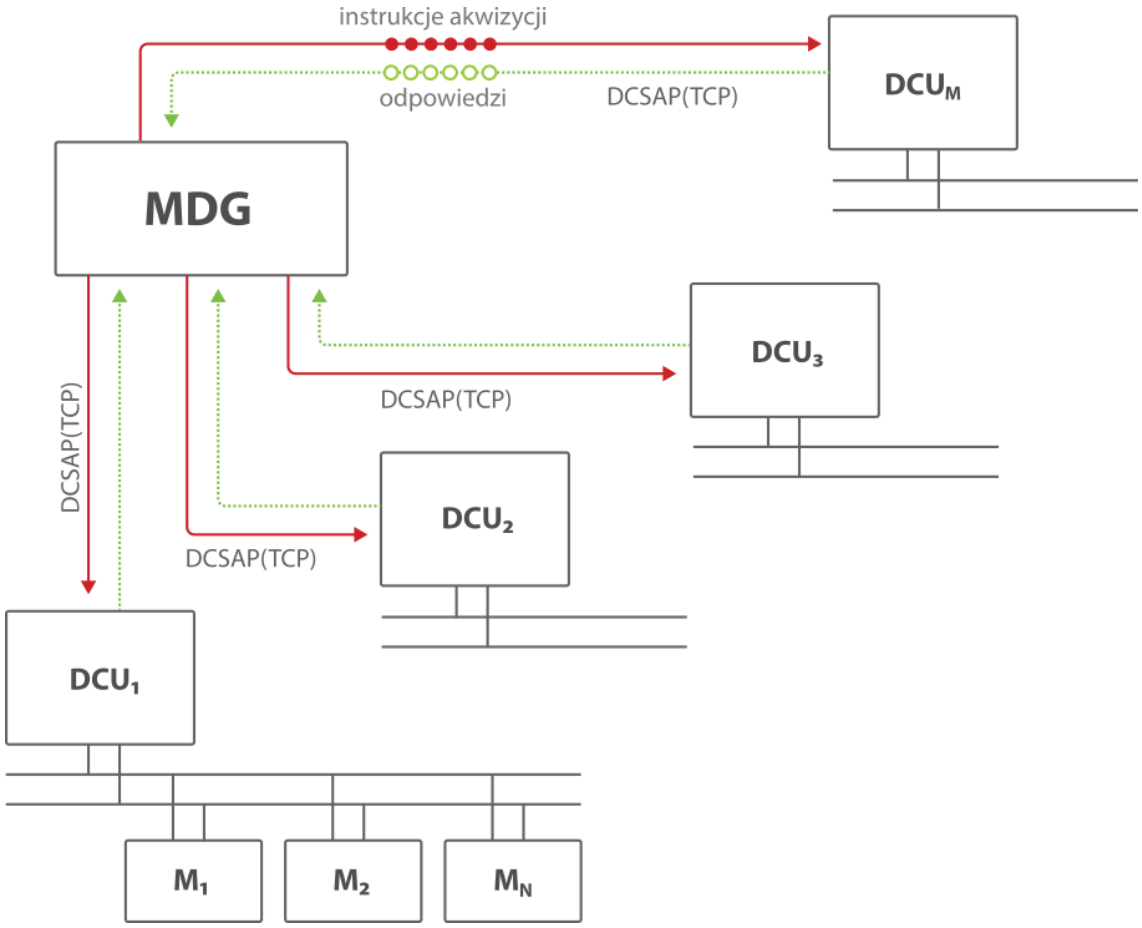
Naczelną zasadą było przede wszystkim uproszczenie struktur danych – to znaczy zastąpienie skomplikowanych struktur klas i obiektów COSEM z wersji 1.x formami prostszymi i znacznie bardziej zestandaryzowanymi. Stąd nastąpiła dość istotna modyfikacja obiektowego modelu danych do implementacji przez koncentrator.

Drugim aspektem, który wpłynął na kształt tej wersji specyfikacji było wprowadzenie opisu niezbędnych do realizacji mechanizmów na potrzeby zabezpieczenia komunikacji, w szczególności w segmencie koncentrator danych – sieć liczników.

2 Wstęp

Protokół DCSAP został opracowany z myślą o komunikacji pomiędzy systemem akwizycji danych pomiarowych (MDG) a koncentratorami liczników energii elektrycznej (DCU), pośredniczącymi w komunikacji z licznikami.

U podstaw akwizycji za pomocą protokołu DCSAP leży założenie o komunikacji z koncentratorami za pomocą strumienia instrukcji akwizycji przekazywanego poprzez dwukierunkowe, bezstratne połączenie utrzymywane z każdym koncentratorem. Instrukcja akwizycji (operacja) to pojedyncze polecenie zapisu/odczytu atrybutu lub wywołanie metody rejestru układu pomiarowego/koncentratora. Akwizycja za pomocą DCSAP została schematycznie przedstawiona na Rys. 1.

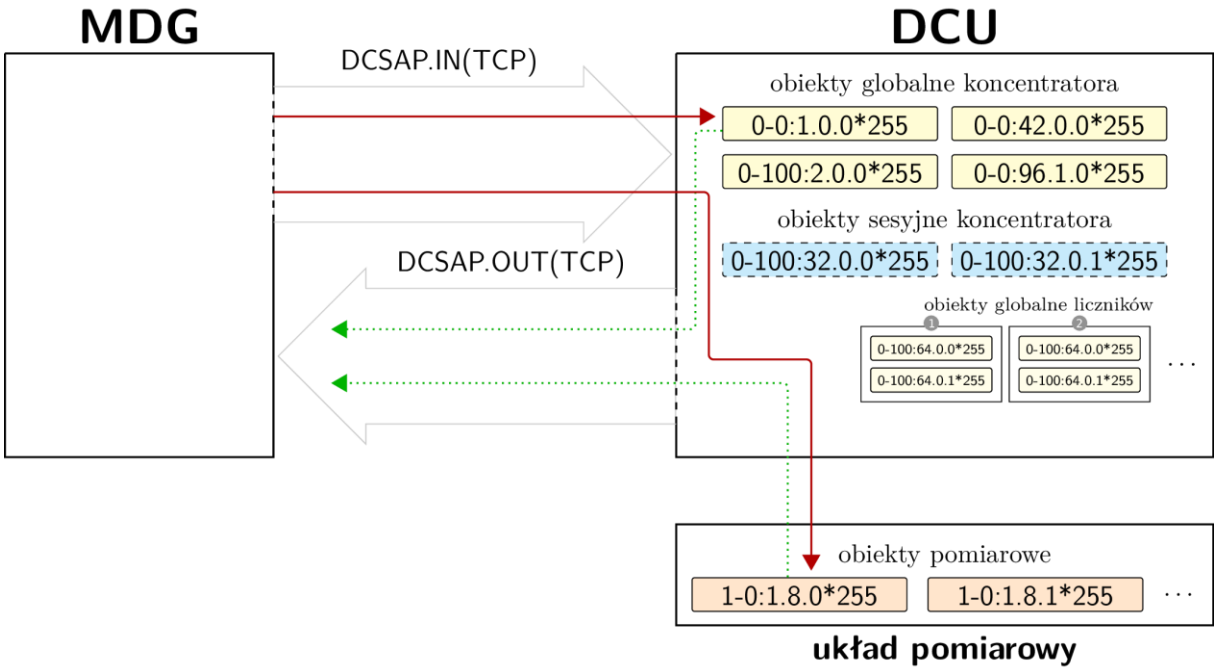


Rys. 1 Akwizycja danych oparta o protokół DCSAP

System akwizycji (MDG) nawiązuje połączenia ze wszystkimi koncentratorami, a następnie zleca im wykonywanie operacji. Operacje służą m.in. do odczytywania danych pomiarowych oraz zmiany stanu i konfiguracji koncentratorów i liczników. Za pomocą tego samego połączenia wysyłane

są operacje oraz odbierane są wyniki zleconych operacji pomiarowych oraz asynchroniczne notyfikacje zdarzeń pochodzące od koncentratora i liczników.

Komunikacja z koncentratorami i licznikami realizowana jest w jednolity sposób, tzn. instrukcje akwizycji dla układów pomiarowych i koncentratorów maja taki sam format oraz dotyczą obiektów pomiarowych albo kontrolnych, dostępnych zarówno w układach pomiarowych jak i w koncentratorach. Sposób przekazywania instrukcji akwizycji do koncentratora i układów pomiarowych przedstawiono schematycznie na Rys. 2.



Rys. 2 Sesja DCSAP

Na Rys. 2 zaznaczono strumień instrukcji adresowanych do obiektów koncentratora oraz do obiektów układów pomiarowych. Założono, ze struktura obiektów koncentratora oraz układów pomiarowych (klasy tych obiektów) jest zgodna ze standardem COSEM [1], [7].

Obiekty koncentratora służą do realizacji jego podstawowych funkcji takich jak przekazywanie listy dostępnych układów pomiarowych, informacji o topologii sieci tych układów, aktualizacji oprogramowania, itp. W ramach obiektów koncentratora wyróżniono obiekty globalne, współdzielone pomiędzy sesjami oraz obiekty sesyjne, przypisane do i dostępne wyłącznie w ramach danego połączenia. Obiekty sesyjne sterują parametrami pojedynczej sesji z systemem akwizycji. Obiektem globalnym jest np. obiekt zawierający listę dostępnych układów pomiarowych.

Obiekty układów pomiarowych to standardowe obiekty realizujące funkcje pomiarowe i sterujące.

W kolejnych rozdziałach dokumentu opisano: założenia dla komunikacji za pomocą protokołu DCSAP, syntaktykę i semantykę komunikacji, strukturę obiektów koncentratora (globalnych

i sesyjnych), przykłady wykorzystania protokołu DCSAP (przypadki użycia), rekomendacje dla implementacji protokołu oraz opracowaną implementację referencyjną serwera protokołu DCSAP, która może zostać zintegrowana z oprogramowaniem koncentratora.

3 Założenia protokołu

Poniżej przedstawiono podstawowe założenia protokołu DCSAP.

3.1 Utrzymywanie sesji sieciowej z DCU

Komunikacja pomiędzy serwerem systemu akwizycji a koncentratorem odbywa się poprzez połączenia TCP/IP, każde stanowiące oddzielną sesję. Sesja nie jest ograniczona czasowo i może być podtrzymywana tak długo jak tylko pozwalają na to warunki telekomunikacyjne. Nie jest wymagane obecność opcji *keepalive* połączenia TCP – w celu sprawdzenia połączenia w przypadku braku nowych zleceń, MDG wysyła okresowo pusty komunikat (ping), który koncentrator musi odesłać w niezmienionej postaci. Pozwala to sprawdzić czy połączenie z koncentratorem nie zostało utracone. Połączenie nawiązywane jest przez MDG do koncentratora (z tego względu wszystkie koncentratory musza być osiągalne adresowo w warstwie IP dla systemu akwizycji). System akwizycji może nawiązać wiele sesji z pojedynczym DCU.

Wymagana jest obsługa co najmniej 3 równoległych sesji.

3.2 Idempotentne operacje i separacja sesji

Wszystkie operacje i ich wyniki przesyłane są w ramach jednej sesji i nie propagują się na kolejne. W przypadku zamknięcia lub zerwania połączenia TCP/IP anulowane są wszystkie niezrealizowane polecenia zlecone w danej sesji. MDG po rozpoczęciu kolejnej sesji ponownie zleci operacje, których wykonanie nie zostało do tej pory potwierdzone. Takie rozwiązanie powoduje możliwość wielokrotnego zlecenia danej operacji jeżeli została ona wykonana w poprzedniej sesji, a potwierdzenie nie zostało dostarczone. Nie zdecydowano się na przeciwdziałanie takim sytuacjom na przykład poprzez użycie unikalnych identyfikatorów operacji. Dzięki temu protokół i jego implementacja staja się prostsze i nie ma potrzeby utrzymywania po stronie koncentratora żadnego kontekstu propagującego się pomiędzy sesjami. Z tego względu wszystkie polecenia musza być idempotentne, a odpowiedzialność za utrzymanie pożądanego stanu koncentratorów i liczników przeniesiona jest na system akwizycji.

3.3 Asynchroniczne przekazywanie odpowiedzi

Odpowiedzi na zlecone polecenia przekazywane są z powrotem do systemu akwizycji tym samym połączeniem TCP/IP. Proces ten odbywa się asynchronicznie w stosunku do kolejno

przekazywanych operacji, dzięki czemu MDG może zlecać kolejne operacje zanim otrzyma potwierdzenie wykonania poprzednich. Tą samą drogą co odpowiedzi przekazywane są asynchroniczne notyfikacje pochodzące z koncentratora i liczników. O kolejności przetwarzania poleceń decyduje koncentrator z uwzględnieniem *priority* w poleceniach DLMS.

3.4 Wykorzystanie protokołu DLMS/COSEM

Protokół DCSAP musi zapewnić możliwość wywoływania poleceń DLMS na poszczególnych licznikach. Dlatego zdecydowano się użyć protokołu DLMS jako bazy, a wszystkie dodatkowe funkcjonalności osiągnięto poprzez zdefiniowanie specyficznych obiektów COSEM reprezentujących logikę koncentratora. Istnieje możliwość definiowania dodatkowych, specyficznych obiektów służących do sterowania unikalnymi mechanizmami opracowanymi przez producentów koncentratorów. Oznacza to, ze protokół DCSAP jest łatwo rozszerzalny i będzie nadążał za nowymi potrzebami.

Koncentrator jest w pełni odpowiedzialny za zarządzanie asocjacjami DLMS z licznikami. Zakładana jest obsługa przez licznik co najmniej trzech typów asocjacji klienta:

- Public (ID = 16) – do identyfikacji licznika
- Management (ID=1) – do regularnej współpracy z systemem akwizycji
- Firmware Update (ID=3) – do aktualizacji oprogramowania licznika

Ze względu na wydajność procesu akwizycji danych zalecane jest ciągłe utrzymywanie przez koncentrator otwartych asocjacji Mangement ze wszystkimi obsługiwanymi licznikami.

Mechanizm dostarczania i zarządzania informacjami niezbędnymi do uwierzytelniania i szyfrowania oraz autentykowania komunikacji jest omówiony w późniejszych rozdziałach niniejszej specyfikacji.

Komunikaty DCSAP (opisane szerzej w rozdziale następnym) zawierające dane kierowane do / odbierane z liczników w polu *dlms-data* mają zawsze postać nieszyfrowaną:

- `get-request/get-response`,
- `set-request/set-response`,
- `action-request/action-response`,
- `event-notification-request`.

przy czym muszą być wspierane mechanizmy *selective-access* i *multiple-references*.

W zależności od ustawień bezpieczeństwa komunikacji koncentratora z licznikami komunikaty te mogą zostać przetransponowane na odpowiedniki szyfrowane / uwierzytelniane z wykorzystaniem kluczy globalnych:

- `glo-get-request/glo-get-response`,
- `glo-set-request/glo-set-response`,
- `glo-action-request/glo-action-response`,
- `glo-event-notification-request`.

Ustawienia bezpieczeństwa komunikacji koncentratora z licznikami decydują również o rodzaju autentykacji wykorzystywanego podczas nawiązywania asocjacji z licznikami.

W komunikacji inicjowanej i realizowanej autonomicznie z licznikami koncentrator samodzielnie wybiera właściwą asocjację i rodzaj komunikacji (*broadcast* / *unicast*). Reguły są następujące:

działanie	**asocjacja**	**rodzaj komunikacji**
pobieranie informacji identyfikacyjnych	Public	*unicast*
synchronizacja liczników ramek	Public	*unicast*
realizacja procesu aktualizacji firmware licznika	Firmware Update	*unicast/broadcast*
inne operacje	Management	*unicast*

Tab. 3 Asocjacje i rodzaj komunikacji w przypadku komunikacji realizowanej autonomicznie przez koncentrator

W komunikacji inicjowanej z systemu centralnego, która jest realizowana poprzez komunikację bezpośrednią z licznikami koncentrator wybiera zawsze asocjację zgodnie z ustawieniami obiektu PLC Client ID (Class_id = 1, OBIS = 0-100:32.0.4*255) i rodzaj komunikacji *unicast*.

3.5 Obsługa obiektów liczników ramek urządzeń pomiarowych

W sytuacji kiedy z licznikiem ma być prowadzona komunikacja szyfrowana / uwierzytelniana – zgodnie z [2] w komunikatach są wykorzystywane liczniki ramek. Ponieważ urządzenia pomiarowe mogą nawiązywać komunikację z różnymi koncentratorami w związku zarówno z montażami / demontażami jak i w zależności od aktualnej topologii sieci PLC – sytuacja jest zmienna w czasie. Dlatego też wymagana jest znajomość przez koncentrator aktualnych wartości obiektów liczników ramek urządzeń pomiarowych – czyli następuje konieczność synchronizacji tych wartości między urządzeniami pomiarowymi a koncentratorami.

Funkcjonalność synchronizacji liczników ramek między koncentratorem a urządzeniami pomiarowymi jest realizowana z wykorzystaniem asocjacji Public.

Koncentrator nadaje numery ramek kolejnych komunikatów kierowanych do danego licznika inkrementując je o 1.

W sytuacji przekroczenia połowy zakresu licznika ramek (wartości większe od 0x7FFFFFFF) uważa się, że odpowiadający szyfrujący klucz globalny traci ważność. W związku z tym koncentrator powinien zgłosić błąd DCSAP *EFCLIMITREACHED* i zaprzestać obsługi innych poleceń DLMS niż wymiana właściwego klucza *global unicast/broadcast encryption key* (polecenie ACTION wywołania metody *global_key_transfer(data)*). Wymiana klucza skutkuje również wyzerowaniem odpowiadającego licznika ramek i szyfrowana / uwierzytelniana komunikacja między koncentratorem a urządzeniem pomiarowym wraca do normalnego trybu działania.

3.6 Rozszerzona adresacja DLMS

W warstwie komunikacyjnej protokół DLMS rozszerzono o identyfikator / adres urządzenia, którego dotyczy dane polecenie. Dzięki temu po nawiązaniu sesji z koncentratorem przekazywane mogą być polecenia DLMS dla wszystkich liczników obsługiwanych przez dany koncentrator. Sam koncentrator również posiada dobrze określony identyfikator w tej przestrzeni i może być adresatem poleceń DLMS.

3.7 Opcjonalne zabezpieczenie za pomocą TLS

Szyfrowanie komunikacji może zostać zrealizowane za pomocą protokołu TLS na poziomie połączenia TCP/IP. Nie przewiduje się wykrywania typu połączenia (TLS czy bez) przez serwer na jednym porcie TCP. Zaleca się implementację wg specyfikacji TLS 1.2 lub nowszej.

4 Komunikacja

W komunikacji pomiędzy systemem akwizycji a koncentratorami używane są komunikaty kodowane zgodnie ze standardem A-XDR [3]. Jest to ten sam sposób kodowania co używany w DLMS/COSEM.

Podstawę stanowi komunikat DLMS, do którego dodane zostały identyfikator urządzenia, identyfikator komunikatu oraz rozmiar komunikatu DLMS stający się kodem błędu w przypadku wartości ujemnych. Nagłówek komunikatu ma zawsze stały rozmiar wynoszący 16 bajtów, po którym następują dane komunikatu DLMS/COSEM. Na listingu 1 przedstawiono definicję struktury komunikatu w formacie ASN.1. Użyte typy zdefiniowano w specyfikacji DLMS (Green Book) [2], [5]. (*Unsigned32* i *Integer32* są 4-bajtowymi liczbami całkowitymi).

```
DCSAP-PDU ::= SEQUENCE
{
       device-id      Unsigned32     -- identyfikator urządzenia
       message-id     Unsigned64     -- identyfikator komunikatu
       data-size      Integer32      -- rozmiar danych DLMS lub kod błędu
       dlms-data      xDLMS-APDU     -- dane DLMS (pole obecne tylko gdy data-size > 0)
}
```

Listing 1. Struktura komunikatu protokołu DCSAP

Komunikaty DCSAP-PDU służą zarówno do przesyłania poleceń do koncentratora jak i do przesyłania odpowiedzi oraz notyfikacji o zdarzeniach z koncentratora. Komunikaty przesyłane są asynchronicznie w obu kierunkach.

Identyfikator urządzenia w zleceniu wskazuje na koncentrator lub licznik w nim zarejestrowany, który jest adresatem polecenia. W odpowiedzi lub zgłoszeniu zdarzenia, identyfikator urządzenia wskazuje na nadawcę komunikatu – licznik lub koncentrator. Wartość 0 przypisana jest do koncentratora, natomiast pozostałe wartości są dynamicznie przydzielane przez koncentrator zarejestrowanym licznikom. Politykę przydzielania identyfikatorów pozostawiono do decyzji producenta koncentratora.

Identyfikator komunikatu polecenia musi być użyty przez koncentrator w komunikacie będącym odpowiedzią na niego. Polityka używania identyfikatorów komunikatów zależy wyłącznie od systemu akwizycji . W komunikacie notyfikacji identyfikator musi przyjmować wartość 0.

Rozmiar danych DLMS musi być zgodny z dalszą zawartością komunikatu. Wartość ta służy do separowania komunikatów bez potrzeby analizy struktury danych DLMS. Wartości niedodatnie oznaczają brak danych DLMS, czyli brak pola *dlms-data*. Wartość 0 służy do kontroli połączenia i taki komunikat koncentrator musi odesłać w niezmienionej postaci. Wartości ujemne mogą się pojawić jedynie w odpowiedzi koncentratora dla zgłoszenia problemu z realizacją danego zlecenia. Zwracane kody błędów przedstawiono w Tab. 4. W przypadku błędów w realizacji zleceń na poziomie DLMS, należy używać komunikatów zwrotnych DLMS zawierających odpowiedni kod błędu.

KB	Symbol	Opis
-1	*EUNKNOWN*	Nieznany identyfikator urządzenia.
-4	*EINVALID*	Nieprawidłowe dane (nierozpoznany komunikat DLMS lub błąd jego formatu).
-5	*ETIMEOUT*	Czas oczekiwania na odpowiedź z urządzenia uległ przekroczeniu.
-6	*EINACCESSIBLE*	Urządzenie jest znane ale niedostępne (np. z powodu aktualizacji).
-7	*EARQERROR*	Błąd podczas nawiązywania asocjacji z licznikiem
-8	*EFCLIMITREACHED*	Przekroczono dopuszczalny limit wartości licznika ramek (górna połowa zakresu)

Tab. 4 Kody błędów używane w komunikatach protokołu DCSAP

4.1 Polecenia

W poleceniach DCSAP mogą zostać użyte następujące typy i warianty komunikatów DLMS:

- `get-request [192] IMPLICIT Get-Request`
  - `get-request-normal [1] IMPLICIT Get-Request-Normal`
  - `get-request-with-list [3] IMPLICIT Get-Request-With-List`
- `set-request [193] IMPLICIT Set-Request`
  - `set-request-normal [1] IMPLICIT Set-Request-Normal`
  - `set-request-with-list [4] IMPLICIT Set-Request-With-List`
- `action-request [195] IMPLICIT Action-Request`
  - `action-request-normal [1] IMPLICIT Action-Request-Normal`
  - `action-request-with-list [3] IMPLICIT Action-Request-With-List`

DLMS definiuje dodatkowo warianty komunikatów, które mają zastosowanie w przypadku, gdy ramki w protokole łącza danych mają ograniczony maksymalny rozmiar. DCSAP działa w warstwie wyższej i wykorzystuje strumieniowy protokół TCP, który nie ma takiego ograniczenia. Jednakże w komunikacji koncentratora z licznikami bardzo prawdopodobne jest wystąpienie wspomnianego ograniczenia, dlatego koncentrator musi wówczas „przepakować” żądanie do licznika tak, by mógł je przesłać używanym medium.

Dla zapytań o dane z rejestrów klasy 7 (*profile_generic*, dane z atrybutu 2 - *buffer*) wymagane jest aby koncentrator w komunikacji z licznikiem obsługiwał mechanizm *selective-access* w pełnej postaci, tzn.

```
range_descriptor ::= structure
{
	restricting_object: capture_object_definition
	from_value: CHOICE
	to_value: CHOICE
	selected_values: array capture_object_definition
}
entry_descriptor ::= structure
{
	from_entry: double-long-unsigned
	to_entry: double-long-unsigned
	from_selected_value: long-unsigned
	to_selected_value: long-unsigned
}
```

zgodnie z opisem w dokumentach [1] i [7].

Koncentrator powinien obsługiwać dane typu *date-time* uwzględniając przesunięcie czasu zgodnie z podaną strefą czasową oraz z obowiązującym czasem letnim bądź zimowym. Koncentrator nie powinien zakładać, że otrzymywane dane dotyczące czasu są w tej samej strefie czasowej co zegar koncentratora lub licznika.

Komunikaty DLMS poleceń w polu *invoke-id-and-priority* posiadają bit *priority*, który oznacza żądania o zwiększonym priorytecie. Koncentrator odbierając komunikat z ustawionym bitem *priority* powinien obsłużyć zawarte w nim żądanie przed dotychczas niezrealizowanymi jeszcze żądaniami, które nie mają tego bitu ustawionego.

Pozostałe bity w polu *invoke-id-and-priority* nie są używane przez system akwizycji, dlatego koncentrator w razie konieczności może je dowolnie modyfikować – pole *invoke-id-and-priority* w odpowiedzi nie musi być zgodne z *invoke-id-and-priority* polecenia.

Przykład Get-Request: Zapytanie o liczydło energii czynnej (A+) licznika o identyfikatorze 1

```
DCSAP-PDU
   00 00 00 01                            device-id (= 1)
   00 00 00 00 00 00 01 01                message-id (= 257)
   00 00 00 0D                            data-size (= 13)
   xDLMS-APDU
      C0                                  get-request
      Get-Request
         01                               get-request-normal
         Get-Request-Normal
            00                            invoke-id-and-priority (priority = 0)
            Cosem-Attribute-Descriptor    cosem-attribute-descriptor (= 3/1-0:1.8.0*255/2)
               Cosem-Class-Id             class-id (= 3)
                  00 03
               Cosem-Object-Instance-Id   instance-id (= 1-0:1.8.0*255)
                  01 00 01 08 00 FF
               Cosem-Object-Attribute-Id  attribute-id (= 2)
                  02
            Selective-Access-Descriptor   access-selection (= nothing)
               OPTIONAL
                  00                      (= not present)
```

Przykład Set-Request: Żądanie ustawienia rozmiaru bufora na pomiary zatrzaskiwane w profilu z drugim okresem zatrzaskiwania do licznika o identyfikatorze 11

```
DCSAP-PDU
   00 00 00 0B                            device-id (= 11)
   00 00 00 00 00 01 00 01                message-id (= 65537)
   00 00 00 12                            data-size (= 18)
   xDLMS-APDU
      C1                                  set-request
      Set-Request
         01                               set-request-normal
         Set-Request-Normal
            00                            invoke-id-and-priority (priority = 0)
            Cosem-Attribute-Descriptor    cosem-attribute-descriptor (= 7/1-0:99.2.0*255/8)
               Cosem-Class-Id             class-id (= 7)
                  00 07
               Cosem-Object-Instance-Id   instance-id (= 1-0:99.2.0*255)
                  01 00 63 02 00 FF
               Cosem-Object-Attribute-Id  attribute-id (= 8)
                  08
            Selective-Access-Descriptor   access-selection (= nothing)
               OPTIONAL
                  00                      (= not present)
            Data                          value
               06                         double-long-unsigned (= 200)
                  00 00 00 C8
```

Przykład Action-Request: Żądanie rozłączenia stycznika licznika o identyfikatorze 15

```
DCSAP-PDU
   00 00 00 0F                            device-id (= 15)
   00 00 00 00 00 00 01 02                message-id (= 258)
   00 00 00 0C                            data-size (= 12)
   xDLMS-APDU
      C3                                  action-request
      Action-Request
         01                               action-request-normal
         Action-Request-Normal
            80                            invoke-id-and-priority (priority = 1)
            Cosem-Method-Descriptor       cosem-method-descriptor (= 70/0-0:96.3.10*255/1)
               Cosem-Class-Id             class-id (= 70)
                  00 46
               Cosem-Object-Instance-Id   instance-id (= 0-0:96.3.10*255)
                  00 00 60 03 0A FF
               Cosem-Object-Method-Id     method-id (= 1)
                  01
```

4.2 Odpowiedzi

W odpowiedziach na polecenia DCSAP mogą zostać użyte następujące typy i warianty komunikatów DLMS:

- `get-response [196] IMPLICIT Get-Response`
  - `get-response-normal [1] IMPLICIT Get-Response-Normal`
  - `get-response-with-list [3] IMPLICIT Get-Response-With-List`
- `set-response [197] IMPLICIT Set-Response`
  - `set-response-normal [1] IMPLICIT Set-Response-Normal`
  - `set-response-with-list [5] IMPLICIT Set-Response-With-List`
- `action-response [199] IMPLICIT Action-Response`
  - `action-response-normal [1] IMPLICIT Action-Response-Normal`
  - `action-response-with-list [3] IMPLICIT Action-Response-With-List`

Jeżeli odpowiedź zostanie rozbita przez licznik na wiele ramek przy transmisji do koncentratora, powinna zostać przepakowana w jedną odpowiedź przesyłaną do systemu akwizycji. Na jeden komunikat zlecenia operacji powinien przyjść dokładnie jeden komunikat z odpowiedzią.

Koncentrator powinien przekazywać dane typu date-time nadpisując pole *deviation* zgodnie ze strefą czasową źródła danych w przypadkach gdy w oryginalnych danych pole zawiera wartość UNSPECIFIED.

Przykłady Poniższe przykłady kodowania odpowiedzi korespondują z prezentowanymi wcześniej przykładami zleceń.

Przykład Get-Response: Odpowiedź na zapytanie o wielkość A+ licznika o identyfikatorze 1.

```
DCSAP-PDU
   00 00 00 01                            device-id (= 1)
   00 00 00 00 00 00 01 01                message-id (= 257)
   00 00 00 0D                            data-size (= 13)
   xDLMS-APDU
      C4                                  get-response
      Get-Response
         01                               get-response-normal
         Get-Response-Normal
            00                            invoke-id-and-priority
            Get-Data-Result               result (= 54132)
               00                         data
               Data
                  15                      long64-unsigned
                  00 00 00 00 00 00 D3 74 (= 54132)
```

Przykład: Set-Response: Odpowiedź na żądanie ustawienia rozmiaru bufora na pomiary zatrzaskiwane w profilu z drugim okresem zatrzaskiwania od licznika o identyfikatorze 11

```
DCSAP-PDU
   00 00 00 0B                            device-id (= 11)
   00 00 00 00 00 01 00 01                message-id (= 65537)
   00 00 00 04                            data-size (= 4)
   xDLMS-APDU
      C5                                  set-response
      Set-Response
         01                               set-response-normal
         Set-Response-Normal
            00                            invoke-id-and-priority (priority = 0)
            Data-Access-Result            result (= 3 - read-write-denied)
               03
```

Przykład: Odpowiedź na żądanie rozłączenia stycznika licznika o identyfikatorze 15

```
DCSAP-PDU
   00 00 00 0F                            device-id (= 15)
   00 00 00 00 00 00 01 02                message-id (= 258)
   00 00 00 05                            data-size (= 5)
   xDLMS-APDU
      C7                                  action-response
      Action-Response
         01                               action-response-normal
         Action-Response-Normal
            80                            invoke-id-and-priority
            Action-Response-With-Optional-Data    single-response
               Action-Result              result (= 0 - success)
                  00
               Get-Data-Result            return-parameters (= nothing)
                  OPTIONAL
                     00                   (= not present)
```

4.3 Notyfikacje

Notyfikacje zawierają następujący typ komunikatu DLMS:

- `event-notification-request [194] IMPLICIT EventNotificationRequest`

Mechanizm notyfikacji służy do asynchronicznego powiadamiania systemu akwizycji o zdarzeniach i zmianach zachodzących w koncentratorze oraz licznikach. Jest to mechanizm opcjonalny z punktu widzenia użytkowego (i domyślnie wyłączony w każdej nowej sesji), ale jego obsługa na koncentratorze jest obowiązkowa. W każdej sesji, w której mechanizm notyfikacji został

włączony komunikaty notyfikacji powinny być przekazywane do klienta. System akwizycji może alternatywnie cyklicznie sprawdzać stan odpowiednich obiektów. Takie sprawdzanie jest dużo mniej wydajne, ale może się okazać jedynym rozwiązaniem dla tych koncentratorów, z którymi komunikacja wymaga pełnej kontroli nad transmisją danych w obu kierunkach.

Przykład Event-Notification-Request: Powiadomienie o zdarzeniu zarejestrowanym przez licznik o identyfikatorze 127 w pierwszym dzienniku zdarzeń

```
DCSAP-PDU
   00 00 00 7F                          device-id (= 127)
   00 00 00 00 00 00 00 00              message-id (= 0)
   00 00 00 0C                          data-size (= 12)
   xDLMS-APDU
      C2                                event-notification-request
      Event-Notification-Request
         OCTET STRING                   time (= nothing)
            OPTIONAL
               00                       (= not present)
         Cosem-Attribute-Descriptor     cosem-attribute-descriptor (= 7/0-0:99.98.0*255/2)
            Cosem-Class-Id              class-id (= 7)
               00 07
            Cosem-Object-Instance-Id    instance-id (= 0-0:99.98.0*255)
               00 00 63 62 00 FF
            Cosem-Object-Attribute-Id   attribute-id (= 2)
               02
         Data                           attribute-value
            FF                          dont-care
```

4.4 Przesyłanie znaczników czasu

Koncentrator powinien używać oznaczenia typów Time, Date, DateTime zamiast odpowiadających Octet-String wszędzie tam, gdzie w specyfikacji DLMS/COSEM pozostawiono wybór.

W znacznikach czasu generowanych przez koncentrator (dotyczy także aktualnego czasu koncentratora – obiektu 8/0-0:1.0.0*255/2), pole *deviation* powinno zawierać przesunięcie obowiązującej strefy czasowej w minutach (niedopuszczalne jest sygnalizowanie czasu w strefie lokalnej – wartości 0x8000). Flaga czasu letniego (DST) powinna być ustawiana niezależnie od zmian pola *deviation* przy zmianie czasu. Przykładowo w strefie czasowej Polski:

- czas zimowy: 2014-01-01 (3) 01:23:45.89 +01:00 DST=0
- czas letni: 2014-07-01 (2) 01:23:45.89 +02:00 DST=1

5 Obiekty COSEM

Koncentrator, żeby spełniać swoją funkcję, musi realizować kilka podstawowych mechanizmów. Sterowanie oraz pobieranie wyników działania tych mechanizmów odbywa się poprzez specyficzne atrybuty i metody odpowiednich obiektów COSEM zdefiniowanych dla koncentratora. Te atrybuty i metody są generalizowane za pomocą klas interfejsów COSEM stowarzyszonych z tymi obiektami. System akwizycji korzysta z obiektów dla koncentratora tak samo jak z obiektów przewidzianych dla liczników.

W przypadku implementacji licznika bilansującego jako jednego urządzenia z koncentratorem, licznik powinien być traktowany w ramach DCSAP jako oddzielne urządzenie.

Specyficzne obiekty COSEM koncentratora zostały podzielone na dwie główne kategorie. Pierwsza z nich dotyczy globalnych mechanizmów, pozwala odczytać oraz zmienić aktualny stan koncentratora. Druga grupa to obiekty sesyjne, związane ze stanem aktualnej sesji. Ich zmiany nie propagują się pomiędzy sesjami zarówno tymi nawiązanymi po sobie jak i utrzymywanymi równolegle. Można powiedzieć, że z każdą sesją związane są tymczasowe obiekty, które można odczytywać i używać do sterowania tą konkretną sesją.

Dodatkowo zdefiniowane zostały globalne obiekty liczników realizowane przez koncentrator. Są one realizowane przez oprogramowanie koncentratora, ale istnieją w niezależnych instancjach dla każdego zarejestrowanego licznika. Dostęp do nich z poziomu protokołu DCSAP, poza standardowym podaniem deskryptora atrybutu lub metody COSEM w stosownym typie komunikatu DLMS, wymaga posłużenia się w nagłówku tego komunikatu identyfikatorem urządzenia należącym do licznika.

Wszystkie obiekty realizowane przez koncentrator i zdefiniowane w ramach tej dokumentacji mają następujące cechy:

1. Żądania ich dotyczące i operujące na wielu obiektach (tj. *Get-Request-With-List*, *Set-Request-With-List* lub *Action-Request-With-List*) nie mogą odwoływać się w jednym i tym samym żądaniu do obiektów nierealizowanych przez koncentrator (np. obiektów zaimplementowanych w licznikach).
2. Realizacja żądań odwołujących się do nich jest bezzwłoczna – bez sięgania do zewnętrznych zasobów (tj. znajdujących się poza koncentratorem lub wymagających komunikacji z urządzeniami zewnętrznymi).
3. Dotyczące ich powiadomienia wysyłane są dopiero po zakończeniu zmiany czy aktualizacji, której dotyczą, a nie w jej trakcie bądź też przy rozpoczęciu jej wykonywania.

5.1 Reguły nazywania obiektów specyficznych dla DCSAP w ramach OBIS

Kody OBIS obiektów realizowanych przez koncentrator i wymaganych przy komunikacji za pomocą protokołu DCSAP są definiowane w ramach podgrupy kodów do zastosowań użytkowych (*utility specific*, patrz rozdz. 5 w [1]), gdzie grupa wartości B musi być z zakresu 65–127.

Dla dostawców zewnętrznych chcących wykorzystać zdefiniowany tu schemat, przewidziano osobną pulę obiektów. Zaleca się jej wewnętrzne usystematyzowanie w analogiczny sposób jak dokonano to z pozostałymi pulami.

Pule obiektów	Kod OBIS					
	A	B	C	D	E	F
Globalne obiekty koncentratora	0	100	0-31	d	e	f
Sesyjne obiekty koncentratora	0	100	32-63	d	e	f
Globalne obiekty licznika realizowane przez koncentrator	0	100	64-95	d	e	f
(zarezerwowane)	0	100	96-127	d	e	f
Obiekty definiowane przez dostawców	0	100	128-255	d	e	f

Tab. 5 Reguły doboru kodów OBIS dla nowych obiektów

Przy czym: d, e, f - dowolna wartość niepowodująca kolizji z wcześniej zdefiniowanymi obiektami

5.2 Klasy Interfejsów

Dla potrzeb zdefiniowania niektórych obiektów realizowanych przez koncentrator wprowadzono specyficzne, dedykowane klasy. Wymienione są w Tab. 6 i opisane w dalszej części tego podrozdziału.

Nazwa klasy interfejsu	Numer klasy (class_id)	Kardynalność w koncentratorze	Kardynalność w liczniku
Device firmware	40101	1	1
DLMS Security Setup	40199	1..n	1..n

Tab. 6 Klasy interfejsów wprowadzone na potrzeby protokołu DCSAP

5.2.1 Klasa *Device firmware*

Device firmware		***1***	***class_id=40101, version=0***		
Atrybuty		***Typ danych***	***Min.***	***Max.***	***Def.***
1.	logical_name	octet-string			
2.	version	octet-string			
3.	checksum	octet-string			
4.	last_update_time	date-time			
5.	last_update_id	double-long-unsigned			
6.	last_update_status	integer			
7.	last_update_progress	unsigned			
Specyficzne metody		*obowiązkowe (m) / nie (o)*			
1.	start_update(https_url)	m			
2.	abort_update(update_id)	m			

Tab. 7 Klasa *Device firmware* (*class_id* = 40101)

KSA	Symbol	Opis
7	*DEVREBOOT*	Trwa restartowanie urządzenia po wgraniu obrazu.
6	*DEVRELOAD*	Trwa restartowanie modułów, które uległy aktualizacji. (Urządzenie nie jest w tym wypadku restartowane.)
5	*IMGWRITE*	Trwa wgrywanie obrazu do urządzenia.
4	*IMGBACKUP*	Trwa tworzenie kopii zapasowej modułów poddawanych aktualizacji.
3	*IMGVERIF*	Trwa sprawdzanie obrazu pod kątem poprawności jak i zgodności z urządzeniem.
2	*IMGDOWNLOAD*	Trwa pobieranie obrazu.
1	*CERTVERIF*	Trwa pozyskiwanie i sprawdzanie certyfikatu zwracanego przez serwer HTTPS pod adresem obrazu.
0	*SUCCESS*	Ostatnia aktualizacja zakończyła się pomyślnie.
-1	*EFWABORT*	Aktualizacja została anulowana na żądanie.
-2	*EMAINTAIN*	Aktualizacja została anulowana z powodu wcześniej zainicjowanej i jeszcze niezakończonej czynności utrzymaniowej, np. aktualizacji oprogramowania czy restartu urządzenia.
-3	*EWRONGCERT*	Aktualizacja została anulowana z powodu niepoprawnego certyfikatu, tj. wygasłego lub podpisanego przez nieznany urząd certyfikacji.
-4	*EINVURL*	Aktualizacja została anulowana z powodu złego adresu URL, tj. niepoprawnego lub nieprowadzącego do pliku obrazu.
-5	*EINVCHKSUM*	Aktualizacja została anulowana z powodu niepomyślnej weryfikacji sumy kontrolnej obrazu.
-6	*EUNAVAIL*	Aktualizacja została anulowana z powodu niedostępności urządzenia.
-7	*EFWINVALID*	Aktualizacja została anulowana z powodu wskazania obrazu nieobsługiwanego przez urządzenie.

-8	*EDEVABORT*	Aktualizacja została anulowana przez urządzenie.

Tab. 8 Kody statusowe aktualizacji oprogramowania jakie może przyjmować atrybut *last_update_status* klasy *Device firmware*

Atrybut / metoda	Opis
version	Wersja oprogramowania obecna na urządzeniu. W przypadku wielu niezależnie wersjonowanych modułów, powinna podawać ich poszczególne wersje.
checksum	Suma kontrolna oprogramowania obecnego na urządzeniu. W przypadku wielu niezależnych modułów oprogramowania pole powinno zawierać sumy kontrolne poszczególnych modułów.
last_update_time	Czas ostatniego rozpoczęcia aktualizacji.
last_update_id	Numer sekwencyjny ostatniej aktualizacji.
last_update_status	Status ostatniej aktualizacji. Wartości dodatnie oznaczają, że trwa obecnie aktualizacja i informują o tym, na jakim etapie się ona znajduje. Wartość 0 oznacza, że ostatnia aktualizacja zakończyła się sukcesem. Wartości ujemne oznaczają, że ostatnia aktualizacja zakończyła się niepowodzeniem i informują o powodzie tego niepowodzenia. Opisane kody statusów znajdują się w Tab. 7.
start_update(https_url)	Zlecenie rozpoczęcia aktualizacji oprogramowania urządzenia. Jeżeli trwa w tym czasie inna aktualizacja lub procedura restartowania urządzenia, zwracany jest błąd *temporary-failure*. W przeciwnym wypadku zwiększony zostaje numer sekwencyjny ostatniej aktualizacji i zwracany jest on razem z kodem *success.* Następnie rozpoczyna się właściwy przebieg aktualizacji. Konieczne jest pobranie obrazu oprogramowania wskazanego w argumencie: *https_url ::= octet_string* który zawiera URL pliku dostępnego poprzez protokół HTTPS. Jest ono poprzedzone sprawdzeniem certyfikatu przedstawionego przy nawiązaniu łączności z zadanym serwerem – czy nie wygasł i czy został podpisany przez odpowiedni, znany koncentratorowi, urząd certyfikacji. Po pobraniu obrazu wykonywany jest test jego integralności i zgodności z urządzeniem, po którym obraz jest wgrywany do pamięci urządzenia. W trakcie trwania aktualizacji atrybuty *last_update_status* i *last_update_progress* są na bieżąco uaktualniane. Po zakończeniu aktualizacji powinno zostać wygenerowane powiadomienie od aktualizowanego urządzenia z obiektu implementującego omawianą tu klasę COSEM, o ile aktualizacja nie wymaga rozłączenia sesji koncentratora z systemem akwizycji.

abort_update(update_id)	Przerywa, o ile to możliwe na danym etapie, aktualizację oprogramowania o przekazanym w argumencie numerze sekwencyjnym. *update_id ::= double-long-unsigned* Użycie nr. sekwencyjnego pozwala uniknąć przypadkowego anulowania aktualizacji innej niż uprzednio zlecona (po jej zakończeniu mogła zostać zlecona kolejna aktualizacja, np. z poziomu innej sesji). W przypadku pomyślnego przerwania aktualizacji zwracany jest kod *success*. W przypadku trwania nowszej aktualizacji niż wskazana, zwracany jest kod *object-unavailable*. Gdy nie jest możliwe przerwanie aktualizacji, zwracany jest kod *hardware-fault*. Aktualizacja powinna zostać w pełni przerwana przed wysłaniem odpowiedzi *Action-Response*.

Tab. 9 Opis atrybutów i metod klasy *Device firmware* (class_id = 40101)

5.2.2 Klasa *DLMS Security Setup*

Klasa definiuje parametry bezpieczeństwa komunikacji DLMS związane z odpowiednimi obiektami COSEM obecnymi w licznikach:

DLMS Security Setup		***1..n***	***class_id=40199, version=0***		
Atrybuty		***Typ danych***	***Min.***	***Max.***	***Def.***
1.	logical_name	octet-string			
2.	security_policy	enum	0	3	0
3.	authentication_mechanism_id	unsigned	0	5	1
4.	secret	octet-string			(pusty)
5.	global_unicast_encryption_key	octet-string			(pusty)
6.	global_broadcast_encryption_key	octet-string			(pusty)
7.	global_authentication_key	octet-string			(pusty)

Tab. 10 Klasa *DLMS Security Setup* (class_id = 40199)

Atrybut / metoda	Opis
logical_name	identyfikuje instancję obiektu klasy „DLMS Security setup”
security_policy	określa włączenie mechanizmów szyfrowania i/lub uwierzytelniania transmisji w kontekście Security Suite ID = 0 (Galois/Counter Mode z algorytmem szyfrowania AES-128): enum: 0 - brak 1 - wszystkie wiadomości są uwierzytelniane 2 - wszystkie wiadomości są szyfrowane 3 - wszystkie wiadomości są uwierzytelniane i szyfrowane

authentication_mechanism_id	identyfikator mechanizmu dostępu do danych (autentykacji): unsigned: 0 - COSEM_lowest_level_security_mechanism_name 1 - COSEM_low_level_security_mechanism_name 5 - COSEM_High_Level_Security_Mechanism_Name_Using_GMAC
secret	hasło wykorzystywane w trybie autentykacji LLS z licznikiem
global_unicast_encryption_key	wartość klucza global unicast encryption key - GUEK
global_broadcast_encryption_key	wartość klucza global broadcast encryption key - GBEK
global_authentication_key	wartość klucza (global) authentication key - GAK

Tab. 11 Opis atrybutów i metod klasy DLMS Security Setup (class_id = 40199)

5.3 Obiekty koncentratora

5.3.1 Globalne obiekty koncentratora

Globalne obiekty koncentratora grup: *Informacja o działaniu koncentratora*, *Konfiguracja*, *Dziennik zdarzeń* oraz *Zmiana firmware* są persystentne, tzn. restart koncentratora nie powoduje ich resetowania.

Grupa obiektów: Informacje o działaniu koncentratora

Obiekty udostępniające podstawowe informacje ewidencyjne oraz diagnostyczne na temat funkcjonowania koncentratora. Obiekt *Running firmware version* umożliwia ponadto restart koncentratora.

Obiekt	class_ID COSEM	Kod OBIS
Clock	8	0-0:1.0.0.255
DC logical name	1	0-0:42.0.0.255
Device ID 1	1	0-0:96.1.0.255
DCSAP version	1	0-100:2.0.0.255
Running firmware version	1	0-100:2.0.1.255
Uptime	1	0-100:2.1.0.255
Boot time	1	0-100:2.1.1.255
Boot count	1	0-100:2.1.2.255
DC config ID	1	0-100:2.2.0.255

Tab. 12 Grupa obiektów: *Informacje o działaniu koncentratora*

Grupa obiektów: Konfiguracja

Obiekty umożliwiające skonfigurowanie listy adresów serwerów NTP (aby możliwe było używanie tych samych serwerów czasu w całej infrastrukturze) oraz zdefiniować parametry zabezpieczeń komunikacji TCP (zarówno samego koncentratora jak i serwerów udostępniających pliki firmware urządzeń).

Obiekt	class_ID COSEM	Kod OBIS
NTP server list	1	0-100:0.0.1.255
Trusted firmware certificate list	1	0-100:0.0.3.255
Server certificate chain	1	0-100:0.0.4.255
Server certificate request	1	0-100:0.0.5.255

Tab. 13 Grupa obiektów: *Konfiguracja*

Grupa obiektów: Lista liczników

Najważniejszą funkcją koncentratora jest wykrywanie i rejestrowanie nowych liczników oraz utrzymywanie listy liczników aktualnie działających w jego zasięgu. Obiekty grupy *Lista liczników* definiują struktury danych, dzięki którym koncentrator dzieli się tą listą z systemem akwizycji.

Obiekt	class_ID COSEM	Kod OBIS
Meter list	7	0-100:0.0.0.255
Changed meter ID	1	0-100:1.0.1.255
Changed meter name	1	0-100:1.0.2.255
Changed meter type	1	0-100:1.0.3.255
Changed meter active	1	0-100:1.0.4.255

Tab. 14 Grupa obiektów: *Lista liczników*

Grupa obiektów: Dziennik zdarzeń

Koncentrator, podobnie jak liczniki, jest źródłem zdarzeń, które powinny być w nim składowane z możliwością późniejszego odczytania. W trakcie ich zatrzaśnięcia powinno być wygenerowane powiadomienie do wszystkich sesji z włączoną notyfikacją zdarzeń.

Obiekt	class_ID COSEM	Kod OBIS
DC Event Log	7	0-0:99.98.0.255
DC Event Code	1	0-0:96.11.0.255
DC Event Counter	1	0-0:96.15.0.255
DC Event Comment	1	0-100:0.0.100.255
Meter Event Log	7	0-0:99.98.1.255
Meter Event Code	1	0-0:96.11.1.255
Meter Event Counter	1	0-0:96.15.1.255

Tab. 15 Grupa obiektów: *Dziennik zdarzeń*

Grupa obiektów: Statystyki

Obiekty udostępniające informacje diagnostyczne na temat funkcjonowania komunikacji z punktu widzenia koncentratora.

Obiekt	class_ID COSEM	Kod OBIS
Sessions open	1	0-100:1.0.0.255
Sessions active	1	0-100:1.0.1.255
Bytes received	1	0-100:1.0.10.255
Bytes sent	1	0-100:1.0.11.255
Messages received	1	0-100:1.0.20.255
Messages sent	1	0-100:1.0.21.255
DC requests completed	1	0-100:1.0.30.255
Meter requests completed	1	0-100:1.0.31.255

Tab. 16 Grupa obiektów: *Statystyki*

Grupa obiektów: Wymiana firmware

Wszystkie koncentratory muszą umożliwić aktualizację oprogramowania. Dlatego protokół DCSAP przewiduje konieczność zaimplementowania dedykowanego obiektu w koncentratorze, który umożliwi zlecenie aktualizacji oprogramowania przez system akwizycji.

Obiekt	class_ID COSEM	Kod OBIS
DC firmware	40101	0-100:0.0.1.255

Tab. 17 Grupa obiektów: *Wymiana firmware*

Informacje o działaniu koncentratora

LP	Obiekt/Nazwa atrybutu	Cl	Typ	Wartość	Znaczenie	R/W
	Clock	8	(version=0)	0-0:1.0.0.255		
1	logical_name		octet-string[6]	0000010000FF		R-
2	time		octet-string[12]		aktualny czas i data	RW
3	time_zone		long			RW
4	status		unsigned			R-
5	daylights_savings_begin		octet-string[12]			RW
6	daylights_savings_end		octet-string[12]			RW
7	daylights_savings_deviation		integer			RW
8	daylights_savings_enabled		boolean			RW
9	clock_base		enum			R-
	DC logical name	1	(version=0)	0-0:42.0.0.255		
1	logical_name		octet-string[6]	00002A0000FF		R-
2	value		octet-string[16]		nazwa logiczna koncentratora zgodna z DLMS, przy ograniczeniu do znaków: 0-9, A-Z. Jeśli koncentrator zawiera wbudowany licznik bilansujący, to ich nazwy logiczne mogą być identyczne.	R-
	Device ID 1	1	(version=0)	0-0:96.1.0.255		
1	logical_name		octet-string[6]	0000600100FF		R-
2	value		octet-string[16]		numer seryjny koncentratora	R-
	DCSAP version	1	(version=0)	0-100:2.0.0.255		
1	logical_name		octet-string[6]	0064020000FF		R-
2	value		octet-string[4]		wersja specyfikacji zapisana binarnie x.y.z.b po jednym bajcie na liczbę, np. dla wersji 1.2.3.4: 01 02 03 04	R-
	Running firmware version	1	(version=0)	0-100:2.0.1.255		
1	logical_name		octet-string[6]	0064020001FF		R-
2	value		octet-string		wersja aktualnie działającego firmware	R-
	Uptime	1	(version=0)	0-100:2.1.0.255		

1	logical_name		octet-string[6]	0064020100FF		R-
2	value		long64-unsigned		czas działania koncentratora od ostatniego restartu w milisekundach	R-
M(-1)	restart()				restart koncentratora (metoda o identyfikatorze (-1) z uwagi na to, że niestandardowa dla obiektów class_ID=1)	-W
	Boot time	1	(version=0)	0-100:2.1.1.255		
1	logical_name		octet-string[6]	0064020101FF		R-
2	value		date-time		czas ostatniego restartu koncentratora	R-
	Boot count	1	(version=0)	0-100:2.1.2.255		
1	logical_name		octet-string[6]	0064020102FF		R-
2	value		long64-unsigned		liczba restartów koncentratora od momentu produkcji	R-
	DC config ID	1	(version=0)	0-100:2.2.0.255		
1	logical_name		octet-string[6]	0064020200FF		R-
2	value		octet-string		numer sekwencyjny lub skrót kryptograficzny aktualnej konfiguracji koncentratora.	R-

Tab. 18 Opis atrybutów i metod grupy obiektów: *Informacje o działaniu koncentratora*

Konfiguracja

LP	Obiekt/Nazwa atrybutu	Cl	Typ	Wartość	Znaczenie	R/W
	NTP server list	1	(version=0)	0-100:0.0.1.255		
1	logical_name		octet-string[6]	0064000001FF		R-
2	value		array of octet-string		lista adresów serwerów NTP (IP lub DNS)	RW
	Trusted firmware certificate list	1	(version=0)	0-100:0.0.3.255		
1	logical_name		octet-string[6]	0064000003FF		R-
2	value		array of octet-string		lista zaufanych certyfikatów X509 do weryfikacji serwerów z firmware lub samego firmware (format DER)	RW
	Server certificate chain	1	(version=0)	0-100:0.0.4.255		
1	logical_name		octet-string[6]	0064000004FF		R-
2	value		array of octet-string		łańcuch certyfikatów prezentowany przez server DCSAP/TLS (format DER, w kolejności: CA na końcu)	RW
	Server certificate request	1	(version=0)	0-100:0.0.5.255		
1	logical_name		octet-string[6]	0064000005FF		R-
2	value		octet-string		klucz publiczny w formie certificate-request do podpisania przez CA	RW

Tab. 19 Opis atrybutów i metod grupy obiektów: *Konfiguracja*

Sposób zabezpieczenia przed niepowołaną zmianą konfiguracji pozostawiono do decyzji producenta.

Lista liczników

LP	Obiekt/Nazwa atrybutu	Cl	Typ	Wartość	Znaczenie	R/W
	Meter list	7	(version=1)	0-100:0.0.0.255		
1	logical_name		octet-string[6]	0064000000FF		R-
2	buffer		array		bufor musi być zapisywany w sposób ciągły – nie są dopuszczalne nieregularności w zapisywaniu bufora	R-
3	capture_objects		array	{1,0-0:96.15.1.255,2}; {8,0-0:1.0.0.255,2}; {1,0-100:1.0.1.255,2}; {1,0-100:1.0.2.255,2}; {1,0-100:1.0.3.255,2}; {1,0-100:1.0.4.255,2};	Meter Event Counter Clock Changed meter ID Changed meter name Changed meter type Changed meter active	RW
4	capture_period		double-long-unsigned	0	(zatrzaskiwane asynchronicznie)	R-
5	sort_method		enum	1	FIFO	R-
6	sort_object		capture object definition	none	none	R-
7	entries_in_use		double-long-unsigned			R-
8	profile_entries		double-long-unsigned	2048	maksymalna wielkość listy liczników obsługiwana przez koncentrator	R-
M1	reset(data)					-W
M2	capture(data)					-W
	Changed meter ID	1	(version=0)	0-100:1.0.1.255		
1	logical_name		octet-string[6]	0064010001FF		R-
2	value		double-long-unsigned		adres licznika w DCSAP (device_id) (kopia 0-100:65.0.1.255 zmienionego licznika)	R-
	Changed meter name	1	(version=0)	0-100:1.0.2.255		
1	logical_name		octet-string[6]	0064010002FF		R-
2	value		octet-string[16]		nazwa sieciowa licznika. (kopia 0-0:42.0.0.255 zmienionego licznika)	R-
	Changed meter type	1	(version=0)	0-100:1.0.3.255		
1	logical_name		octet-string[6]	0064010003FF		R-

2	value		octet-string		typ licznika (kopia 0-100:65.0.3.255 zmienionego licznika)	R-
	Changed meter active	1	(version=0)	0-100:1.0.4.255		
1	logical_name		octet-string[6]	0064010004FF		R-
2	value		boolean		flaga oznaczająca czy licznik jest w danym momencie widoczny w sieci (kopia 0-100:65.0.4.255 zmienionego licznika)	R-

Tab. 20 Opis atrybutów i metod grupy obiektów: *Lista liczników*

Wiersze tabeli listy liczników są unikalne względem numeru licznika (kolumny *Changed meter id*) – stare wersje zastępowane są aktualnym stanem dla danego licznika.

Rejestry wirtualne (kolumny 3-6) są chwilowymi kopiami tych samych rejestrów licznika, dla którego dokonywany jest wpis do tabeli. Implementacja dostępu do tych rejestrów dla koncentratora jest opcjonalna – jeśli będzie, powinna zwracać wartości dla ostatnio modyfikowanego licznika.

Obiekt powinien obsługiwać filtrowanie odczytywanych danych z bufora za pomocą *range-descriptor* (*access-selector* 1) przynajmniej dla kolumn: 1,2,3,5,6.

Możliwe jest rozszerzenie tabeli o inne wartości producenta jako dodatkowe kolumny na końcu listy.

Dziennik zdarzeń

LP	Obiekt/Nazwa atrybutu	Cl	Typ	Wartość	Znaczenie	R/W
	DC Event Log	7	(version=1)	0-0:99.98.0.255		
1	logical_name		octet-string[6]	0000636200FF		R-
2	buffer		array		bufor musi być zapisywany w sposób ciągły – nie są dopuszczalne nieregularności w zapisywaniu bufora	R-
3	capture_objects		array	{8,0-0:1.0.0.255,2}; {1,0-0:96.15.0.255,2}; {1,0-0:96.11.0.255,2}; {1,0-100:0.0.100.255,2};	Clock DC Event Counter DC Event Code DC Event Comment	RW
4	capture_period		double-long-unsigned	0	(zatrzaskiwane asynchronicznie)	R-
5	sort_method		enum	1	FIFO	R-
6	sort_object		capture object definition	none	none	R-
7	entries_in_use		double-long-unsigned			R-
8	profile_entries		double-long-unsigned	16384		R-
M1	reset(data)					-W
M2	capture(data)					-W
	DC Event Code	1	(version=0)	0-0:96.11.0.255		
1	logical_name		octet-string[6]	0000600B00FF		R-
2	value		unsigned		kod zdarzenia koncentratora (zestawienie w kolejnej tabeli)	R-
	DC Event Counter	1	(version=0)	0-0:96.15.0.255		
1	logical_name		octet-string[6]	0000600F00FF		R-
2	value		long64-unsigned		nr sekwencyjny ostatniego zdarzenia. Wymagane jest by numer sekwencyjny zdarzenia zachowywał kolejność wystąpienia zdarzeń niezależnie od zmian ustawień zegara koncentratora (zarejestrowane zdarzenie o numerze mniejszym wystąpiło nie później niż zdarzenie o numerze większym).	R-
	DC Event Comment	1	(version=0)	0-100:0.0.100.255		

1	logical_name		octet-string[6]	0064000064FF		R-
2	value		octet-string		dodatkowy komentarz zdarzenia	R-
	Meter Event Log	1	(version=0)	0-0:99.98.1.255		
1	logical_name		octet-string[6]	0000636201FF		R-
2	buffer		array		bufor musi być zapisywany w sposób ciągły – nie są dopuszczalne nieregularności w zapisywaniu bufora	R-
3	capture_objects		array	{8,0-0:1.0.0.255,2}; {1,0-0:96.15.1.255,2}; {1,0-0:96.11.1.255,2}; {1,0-100:1.0.1.255,2}; {1,0-100:1.0.2.255,2};	Clock Meter Event Counter Meter Event Code Changed meter ID Changed meter name	RW
4	capture_period		double-long-unsigned	0	(zatrzaskiwane asynchronicznie)	R-
5	sort_method		enum	1	FIFO	R-
6	sort_object		capture object definition	none	none	R-
7	entries_in_use		double-long-unsigned			R-
8	profile_entries		double-long-unsigned	16384		R-
M1	reset(data)					-W
M2	capture(data)					-W
	Meter Event Code	1	(version=0)	0-0:96.11.1.255		
1	logical_name		octet-string[6]	0000600B01FF		R-
2	value		unsigned		kod zdarzenia licznika (zestawienie w kolejnej tabeli)	R-
	Meter Event Counter	1	(version=0)	0-0:96.15.1.255		
1	logical_name		octet-string[6]	0000600F01FF		R-
2	value		long64-unsigned		nr sekwencyjny ostatniego zdarzenia. Wymagane jest by numer sekwencyjny zdarzenia zachowywał kolejność wystąpienia zdarzeń niezależnie od zmian ustawień zegara koncentratora (zarejestrowane zdarzenie o numerze mniejszym wystąpiło nie później niż zdarzenie o numerze większym).	R-

Tab. 21 Opis atrybutów i metod grupy obiektów: *Dziennik zdarzeń*

KZ	Symbol	Opis
0	*EV_START*	Start systemu koncentratora (boot).
1	*EV_RESTART*	Zlecenie restartowania urządzenia.
2	*EV_FWUPDATEINIT*	Zlecenie aktualizacji oprogramowania urządzenia.
3	*EV_FWUPDATEFINI*	Zakończenie aktualizacji oprogramowania urządzenia.
255	*EV_LOGCLEAR*	Skasowanie zawartości dziennika zdarzeń.

Tab. 22 Kody zdarzeń dotyczące koncentratora (obiekt 0-0:96.11.0.255)

KZ	Symbol	Opis
0	*EV_CLEAR*	Wyczyszczenie listy liczników. (Powinno być zarejestrowane przy starcie jeśli koncentrator nie implementuje persystencji listy pomiędzy restartami).
1	*EV_ADD*	Zarejestrowanie licznika z sieci.
2	*EV_DELETE*	Wyrejestrowanie licznika z sieci.
3	*EV_UPDATE*	Zmiana danych identyfikacyjnych licznika (np. zmiana firmware). Dotyczy wszystkich zmian na liście liczników nie pasujących do innych kodów.
4	*EV_CONFIG*	Wykrycie zmiany konfiguracji licznika.
5	*EV_FWUPDATEINIT*	Zlecenie aktualizacji oprogramowania urządzenia.
6	*EV_FWUPDATEFINI*	Zakończenie aktualizacji oprogramowania urządzenia.
255	*EV_LOGCLEAR*	Skasowanie zawartości dziennika zdarzeń.

Tab. 23 Kody zdarzeń dotyczące liczników (obiekt 0-0:96.11.1.255)

Statystyki

LP	Obiekt/Nazwa atrybutu	Cl	Typ	Wartość	Znaczenie	R/W
	Sessions open	1	(version=0)	0-100:1.0.0.255		
1	logical_name		octet-string[6]	0064010000FF		R-
2	value		long64-unsigned		liczba rozpoczętych sesji DCSAP (zaakceptowanych połączeń)	R-
	Sessions active	1	(version=0)	0-100:1.0.1.255		
1	logical_name		octet-string[6]	0064010001FF		R-
2	value		long64-unsigned		liczba aktywnych sesji DCSAP (aktualnie podłączonych)	R-
	Bytes received	1	(version=0)	0-100:1.0.10.255		
1	logical_name		octet-string[6]	006401000AFF		R-
2	value		long64-unsigned		liczba bajtów odebranych przez koncentrator w połączeniach DCSAP (wewnątrz TCP)	R-
	Bytes sent	1	(version=0)	0-100:1.0.11.255		
1	logical_name		octet-string[6]	006401000BFF		R-
2	value		long64-unsigned		liczba bajtów wysłanych przez koncentrator w połączeniach DCSAP (wewnątrz TCP)	R-
	Messages received	1	(version=0)	0-100:1.0.20.255		
1	logical_name		octet-string[6]	0064010014FF		R-
2	value		long64-unsigned		liczba komunikatów odebranych przez koncentrator w połączeniach DCSAP	R-
	Messages sent	1	(version=0)	0-100:1.0.21.255		
1	logical_name		octet-string[6]	0064010015FF		R-
2	value		long64-unsigned		liczba komunikatów wysłanych przez koncentrator w połączeniach DCSAP	R-
	DC requests completed	1	(version=0)	0-100:1.0.30.255		
1	logical_name		octet-string[6]	006401001EFF		R-
2	value		long64-unsigned		liczba zleceń DSCAP zrealizowanych po stronie koncentratora (bez komunikacji z licznikami)	R-

	Meter requests completed	1	(version=0)	0-100:1.0.31.255		
1	logical_name		octet-string[6]	006401001FFF		R-
2	value		long64-unsigned		liczba zleceń DCSAP zrealizowanych po stronie licznika (wysłanych do licznika)	R-

Tab. 24 Opis atrybutów i metod grupy obiektów: *Statystyki*

Obiekty statystyk mogą być zrealizowane jako liczydła zerowane w momencie restartu koncentratora.

Wymiana firmware

LP	Obiekt/Nazwa atrybutu	Cl	Typ	Wartość	Znaczenie	R/W
	DC firmware	40101	(version=0)	0-100:0.0.1.255		
1	logical_name		octet-string[6]	0064000001FF		R-
2	version		octet-string			R-
3	checksum		octet-string			R-
4	last_update_time		date-time			R-
5	last_update_id		double-long-unsigned			R-
6	last_update_status		integer			R-
7	last_update_progress		unsigned			R-
M1	start_update(https_url)					-W
M2	abort_update(update_id)					-W

Tab. 25 Opis atrybutów i metod grupy obiektów: *Wymiana firmware*

Koncentratory muszą umożliwić aktualizację oprogramowania. W metodzie aktualizacji przekazany zostaje adres URL wskazujący na obraz binarny aktualizacji. Odpowiedzialność za weryfikację poprawności kodu aktualizacji oraz ewentualną detekcję zgodności wersji oprogramowania i sprzętu spoczywa na koncentratorze.

Aktualizacja oprogramowania koncentratora inicjowana metodą *start_update()* powoduje sprawdzenie czy nie trwa obecnie inna aktualizacja bądź też czy nie zainicjowano uprzednio restartowania koncentratora i jeżeli tak jest, zwracany jest błąd *temporary-failure*. W przeciwnym wypadku koncentrator zwiększa numer sekwencyjny ostatniej aktualizacji i przekazuje go w odpowiedzi wraz z kodem *success*.

Jeżeli koncentrator nie wspiera aktualizacji z podtrzymywaniem sesji, a aktualizowany moduł oprogramowania wymaga zaprzestania normalnej pracy koncentratora i jego restartu, dochodzi do zamknięcia wszystkich sesji na czas trwania aktualizacji. Jeżeli aktualizowany moduł nie wymaga zaprzestania normalnej pracy urządzenia, bądź wspierane jest podtrzymywanie sesji, wówczas atrybuty *last_update_status* i *last_update_progress* powinny być na bieżąco aktualizowane w trakcie trwającej aktualizacji.

System akwizycji po ponownym nawiązaniu połączenia z koncentratorem lub po uzyskaniu stosownego powiadomienia jeżeli nie doszło do utraty sesji, będzie mógł wyczytać atrybuty *last_update_id* i *last_update_status*, by sprawdzić czy aktualizacja się powiodła.

Zgodnie z [1] proces aktualizacji oprogramowania liczników obsługiwany jest przez obiekt klasy Image transfer (class_ID=18) i przebiega w 7 krokach:

1. (opcjonalnie) pobranie rozmiaru transferowanego bloku firmware akceptowanego przez licznik
2. inicjalizacja procesu transferu nowego firmware
3. transfer sekwencji bloków nowego firmware
4. weryfikacja kompletności przesłania nowego firmware do licznika
5. weryfikacja firmware przez licznik
6. (opcjonalnie) weryfikacja gotowości do aktywacji nowego firmware w licznku
7. aktywacja nowego firmware przez licznik

Zakłada się, że:

- cała komunikacja związana z realizacją powyższego algorytmu jest realizowana w kontekście asocjacji Firmware Update,
- kroki 1-2 oraz 4-7 wykorzystują komunikację typu unicast,
- krok 3:
    - w pierwszej fazie (realizowanej globalnie w ramach danego koncentratora) jest realizowany za pomocą mechanizmu broadcast (dla zwiększenia skuteczności propagacji emisja danego bloku powinna być ponawiana kilkukrotnie),
    - w drugiej fazie, w wyniku realizacji kroku nr 4 i przy stwierdzeniu niekompletności firmware w danym liczniku – indywidualnie poprzez komunikację unicast do danego licznika są transferowane brakujące bloki.

W sytuacji broadcast'owej transmisji bloków firmware:

- w bajcie Invoke-Id-And-Priority bit nr 6 (service-class) powinien mieć ustawioną wartość = 0 (Unconfirmed),

Jeżeli dodatkowo z ustawień bezpieczeństwa wynika konieczność zastosowania mechanizmów szyfrowania / uwierzytelniania – ponieważ w komunikatach DLSM wykorzystywane są liczniki ramek – wybierana jest maksymalna wartość atrybutu value odpowiedniego obiektu licznika ramek spośród wszystkich układów pomiarowych zarejestrowanych w danym koncentratorze do których jest kierowana transmisja broadcast.

5.3.2 Obiekty sesyjne

Wszystkie sesyjne obiekty koncentratora są tymczasowe i skojarzone z sesją, która zleca do nich dostęp. Obiekty sesyjne w różnych sesjach są od siebie niezależne. Zmiany ustawień sesji nie powinny wpływać na parametry realizacji operacji zleconych wcześniej.

Obiekt	class_ID COSEM	Kod OBIS
Meter data cache enable	1	0-100:32.0.0.255
Event notification enable	1	0-100:32.0.1.255
Command timeout	1	0-100:32.0.2.255
PLC Client ID	1	0-100:32.0.4.255

Tab. 26 Grupa obiektów: *Obiekty sesyjne*

LP	Obiekt/Nazwa atrybutu	Cl	Typ	Wartość	Znaczenie	R/W
	Meter data cache enable	1	(version=0)	0-100:32.0.0.255		
1	logical_name		octet-string[6]	0064200000FF		R-
2	value		boolean	true	zezwolenie na buforowanie zapytań do licznika	RW
	Event notification enable	1	(version=0)	0-100:32.0.1.255		
1	logical_name		octet-string[6]	0064200001FF		R-
2	value		boolean	false	włączenie asynchronicznego przesyłania zdarzeń	RW
	Command timeout	1	(version=0)	0-100:32.0.2.255		
1	logical_name		octet-string[6]	0064200002FF		R-
2	value		double-long-unsigned	300	maksymalny czas w sekundach, który koncentrator powinien oczekiwać na możliwość wysłania zleconej operacji do licznika (np. gdy licznik jest aktualnie niedostępny lub kolejka zadań jest zbyt długa). Po tym czasie od momentu odebrania zlecenia, operacja powinna zostać odrzucona z kodem błędu *ETIMEOUT* na poziomie DCSAP.	RW
	PLC Client ID	1	(version=0)	0-100:32.0.4.255		
1	logical_name		octet-string[6]	0064200004FF		R-
2	value		unsigned	1	identyfikator klienta (asocjacji) przy komunikacji bezpośredniej z licznikiem: 1 – Management Client 2 – Reading Client 3 – Firmware Update Client 4 – HAN Controller Client 16 – Public Client	RW

Tab. 27 Opis atrybutów i metod grupy obiektów: *Obiekty sesyjne*

Obiekt Meter data cache enable

Jedną z podstawowych funkcji koncentratora jest akceleracja dostępu do zatrzaskiwanych w licznikach profili zużycia energii elektrycznej. Realizuje się to poprzez automatyczne dobieranie w tle kolejnych zatrzaśniętych w licznikach wartości i składowaniu ich w koncentratorze. Przy poleceniu odczytania zakresu takiego profilu, koncentrator natychmiast zwraca dane z własnego bufora nie komunikując się z licznikiem. Przyspiesza to realizację tego typu poleceń i dodatkowo zmniejsza obciążenie medium komunikacyjnego pomiędzy koncentratorem a licznikiem, ponieważ dane, które mogą być z koncentratora pobierane wielokrotnie, transmitowane są po tym medium tylko raz. Pozwala to również na efektywne wykorzystanie łącza pomiędzy koncentratorem i licznikami w czasie braku komunikacji od strony systemu akwizycji.

Mechanizmy akceleracji są specyficznym rozwiązaniem zależnym od producenta koncentratora. W najprostszym ujęciu koncentrator może tylko pośredniczyć w pobieraniu wszystkich danych, także profilowych, z liczników energii elektrycznej. W większości przypadków wydajność kanału komunikacyjnego pomiędzy koncentratorem a licznikami nie pozwoli na taką trywialną realizację. W celu spełnienia wymagań użytkownika systemu niezbędne jest wtedy uwzględnienie mechanizmów akceleracji, przynajmniej w stosunku do pobierania danych profilowych. W tym celu producent koncentratora powinien wprowadzić dedykowane obiekty służące do sterowania takim mechanizmem. Ustawienie tych obiektów może wskazywać elementy profilu, które koncentrator będzie pobierał z liczników w celu buforowania. Możliwe jest też uruchamianie buforowania po rozpoznaniu przez koncentrator pierwszego żądania dotyczącego tego typu danych. W takim przypadku dobrym rozwiązaniem jest wyłączenie buforowania po pewnym czasie, kiedy system akwizycji nie pobiera już danego elementu profilu. W tym ostatnim przypadku czas wygaśnięcia buforowania powinien być konfigurowalny za pomocą dedykowanego obiektu.

W pewnych przypadkach, na przykład w celach diagnostycznych, istnieje potrzeba bezpośredniego dostępu do liczników. Wtedy musi istnieć mechanizm dezaktywujący buforowanie odpowiedzi liczników. W tym celu koncentrator musi implementować obiekt, którego odpowiednie ustawienie wymusza bezpośredni dostęp do liczników w ramach danej sesji.

Obiekt Event notification enable

Mechanizm asynchronicznych notyfikacji pozwala zwiększyć wydajność komunikacji pomiędzy systemem akwizycji a koncentratorem. Polega on na asynchronicznym powiadamianiu o zaistniałych zdarzeniach i zmianie stanu koncentratora lub liczników. Asynchroniczne notyfikacje bazują na dedykowanym komunikacie DLMS – więcej szczegółów na jego temat można znaleźć w dokumencie [2], [5].

Mechanizm asynchronicznych notyfikacji nie może być jednak użyty jeżeli system akwizycji jest zmuszony do pełnej kontroli przepływu danych na medium komunikacyjnym łączącym go z koncentratorem. Z tego powodu mechanizm ten jest domyślnie wyłączony po nawiązaniu sesji i jeżeli system akwizycji zechce go użyć musi odpowiednio wysterować specyficzny obiekt

koncentratora. Ustawienie tego mechanizmu dotyczy tylko danej sesji i nie wpływa na inne równoległe lub następujące po niej.

5.3.3 Globalne obiekty liczników realizowane przez koncentrator

Wszystkie globalne obiekty liczników realizowane przez koncentrator są persystentne, tzn. restart koncentratora nie powoduje utraty ich zawartości.

Grupa obiektów: Firmware i identyfikacja licznika

Obiekt	class_ID COSEM	Kod OBIS
Meter firmware	40101	0-100:64.0.1.255
Meter ID	1	0-100:65.0.1.255
Meter type	1	0-100:65.0.5.255
Meter active	1	0-100:65.0.6.255
Meter config ID	1	0-100:65.0.7.255
Meter time synchronization	1	0-100:65.0.8.255
Meter PRIME EUI	1	0-100:65.0.10.255
Meter PRIME address	1	0-100:65.0.11.255

Tab. 28 Grupa obiektów: *Firmware i identyfikacja licznika*

Grupa obiektów: Obiekty sterujące zabezpieczeniami komunikacji z licznikiem

Grupa obiektów określających parametry umożliwiające nawiązanie asocjacji z licznikiem w trybie LLS oraz HLS, a także definiujące ustawienia dla szyfrowania i podpisywania pakietów DLMS.

Obiekt	class_ID COSEM	Kod OBIS
Meter DLMS Security Setup - Management association	40199	0-100:65.0.3.255
Meter DLMS Security Setup - Firmware Update association	40199	0-100:65.0.4.255
Frame counter - Management association	1	0-100:66.0.3.255
Frame counter - Firmware Update association - global unicast	1	0-100:66.0.4.255
Frame counter - Firmware Update association - global broadcast	1	0-100:66.1.4.255

Tab. 29 Grupa obiektów: *Obiekty sterujące zabezpieczeniami komunikacji z licznikiem*

Firmware i identyfikacja licznika

LP	Obiekt/Nazwa atrybutu	Cl	Typ	Wartość	Znaczenie	R/W
	Meter firmware	40101	(version=0)	0-100:64.0.1.255		
1	logical_name		octet-string[6]	0064400001FF		R-
2	version		octet-string			R-
3	checksum		octet-string			R-
4	last_update_time		date-time			R-
5	last_update_id		double-long-unsigned			R-
6	last_update_status		integer			R-
7	last_update_progress		unsigned			R-
M1	start_update(https_url)					-W
M2	abort_update(update_id)					-W
	Meter ID	1	(version=0)	0-100:65.0.1.255		
1	logical_name		octet-string[6]	0064410001FF		R-
2	value		long64-unsigned		adres licznika w DCSAP (device_id)	R-
	Meter type	1	(version=0)	0-100:65.0.5.255		
1	logical_name		octet-string[6]	0064410005FF		R-
2	value		octet-string		typ licznika – wartość definiowana przez producenta, unikalna dla każdej kombinacji {model fizyczny licznika, model obiektowy DLMS}	R-
	Meter active	1	(version=0)	0-100:65.0.6.255		
1	logical_name		octet-string[6]	0064410006FF		R-
2	value		boolean		flaga oznaczająca czy licznik jest w danym momencie widoczny w sieci (np. widoczny w topologii PRIME)	R-
	Meter config ID	1	(version=0)	0-100:65.0.7.255		
1	logical_name		octet-string[6]	0064410007FF		R-
2	value		octet-string		numer sekwencyjny lub skrót kryptograficzny aktualnej konfiguracji licznika (zmieniany w momencie dowolnej zmiany konfiguracji licznika)	R-

	Meter time synchronization	1	(version=0)	0-100:65.0.8.255		
1	logical_name		octet-string[6]	0064410008FF		R-

2	value		structure
			{
			long,
			long-unsigned
			}
					Stan synchronizacji czasu licznika po wywołaniu metody Sync_time() – para: czas licznika minus czas koncentratora [s], dokładność synchronizacji czasu dla tego licznika [s] (RTT/2 zaokrąglone w górę). Wartości większe (mniejsze) niż możliwe do przedstawienia powinny zostać zastąpione wartością maksymalną (minimalną) dla danego typu.	R-

M(-2)	sync_time()				synchronizuje czas licznika z czasem koncentratora (metoda o identyfikatorze (-2) z uwagi na to, że niestandardowa dla obiektów class_ID=1)	-W
	Meter PRIME EUI	1	(version=0)	0-100:65.0.10.255		
1	logical_name		octet-string[6]	006441000AFF		R-
2	value		octet-string[6]		MAC EUI modemu licznika	R-
	Meter PRIME address	1	(version=0)	0-100:65.0.11.255		
1	logical_name		octet-string[6]	006441000BFF		R-

2	value		structure
			{
			unsigned,
			long-unsigned
			}
					adres licznika w sieci PRIME: para SID, LNID. (null zamiast struktury jeśli licznik nie jest zarejestrowany)	R-

Tab. 30 Opis atrybutów i metod grupy obiektów: *Firmware i identyfikacja licznika*

Liczniki muszą umożliwiać aktualizację oprogramowania. Koncentratory muszą realizować obiekty adresowane identyfikatorami zarejestrowanych liczników, które umożliwią zlecanie aktualizacji oprogramowania przez system akwizycji. Zakładamy, że w metodzie aktualizacji przekazany zostaje adres URL wskazujący na obraz binarny aktualizacji. Odpowiedzialność za weryfikację poprawności kodu aktualizacji, ewentualną detekcję zgodności wersji oprogramowania i sprzętu spoczywa na koncentratorze i licznikach.

Oprogramowanie urządzeń często składa się z wielu modułów niezależnie wersjonowanych. Ewentualne wsparcie aktualizacji częściowych (tj. pokrywających jedynie wybrane moduły) spoczywa również na koncentratorze i licznikach, tzn. używany format obrazu powinien zawierać informacje o modułach, które przewidziane są w nim do aktualizacji. Zalecane jest wówczas, by atrybut version był połączeniem numerów wersji wszystkich modułów oddzielonych wybranym separatorem, np. średnikiem.

Jeżeli system akwizycji zleci wiele kolejnych poleceń aktualizacji oprogramowania zaadresowanych do różnych liczników, ale z użyciem tego samego adresu URL, mogą one zostać obsłużone jednokrotnie pobranym i zbuforowanym obrazem oprogramowania. Powinno to nieznacznie przyspieszyć masowe aktualizacje liczników.

	Meter DLMS Security Setup - Management association	40199	(version=0)	0-100:65.0.3.255		
1	logical_name		octet-string[6]	0064410003FF		R-
2	security_policy		enum	0	(brak bezpieczeństwa)	RW
3	authentication_mechanism_id		unsigned	1	LLS	RW
4	secret		octet-string[8]			-W
5	global_unicast_encryption_key		octet-string[16]			-W
6	global_broadcast_encryption_key		octet-string[16]			-W
7	global_authentication_key		octet-string[16]			-W
	Meter DLMS Security Setup - Firmware Update association	40199	(version=0)	0-100:65.0.4.255		
1	logical_name		octet-string[6]	0064410004FF		R-
2	security_policy		enum	0	(brak bezpieczeństwa)	RW
3	authentication_mechanism_id		unsigned	1	LLS	RW
4	secret		octet-string[8]			-W
5	global_unicast_encryption_key		octet-string[16]			-W
6	global_broadcast_encryption_key		octet-string[16]			-W
7	global_authentication_key		octet-string[16]			-W
	Frame counter - Management association	1	(version=0)	0-100:66.0.3.255	global unicast	
1	logical_name		octet-string[6]	0064420003FF		R-
2	value		double-long-unsigned			R-
	Frame counter - Firmware Update association - global unicast	1	(version=0)	0-100:66.0.4.255	global unicast	
1	logical_name		octet-string[6]	0064420004FF		R-
2	value		double-long-unsigned			R-
	Frame counter - Firmware Update association - global broadcast	1	(version=0)	0-100:66.1.4.255	global broadcast	
1	logical_name		octet-string[6]	0064420104FF		R-
2	value		double-long-unsigned			R-

Tab. 31 Opis atrybutów i metod grupy obiektów: *Obiekty sterujące zabezpieczeniami komunikacji z licznikiem*

6 Wykorzystanie protokołu DCSAP

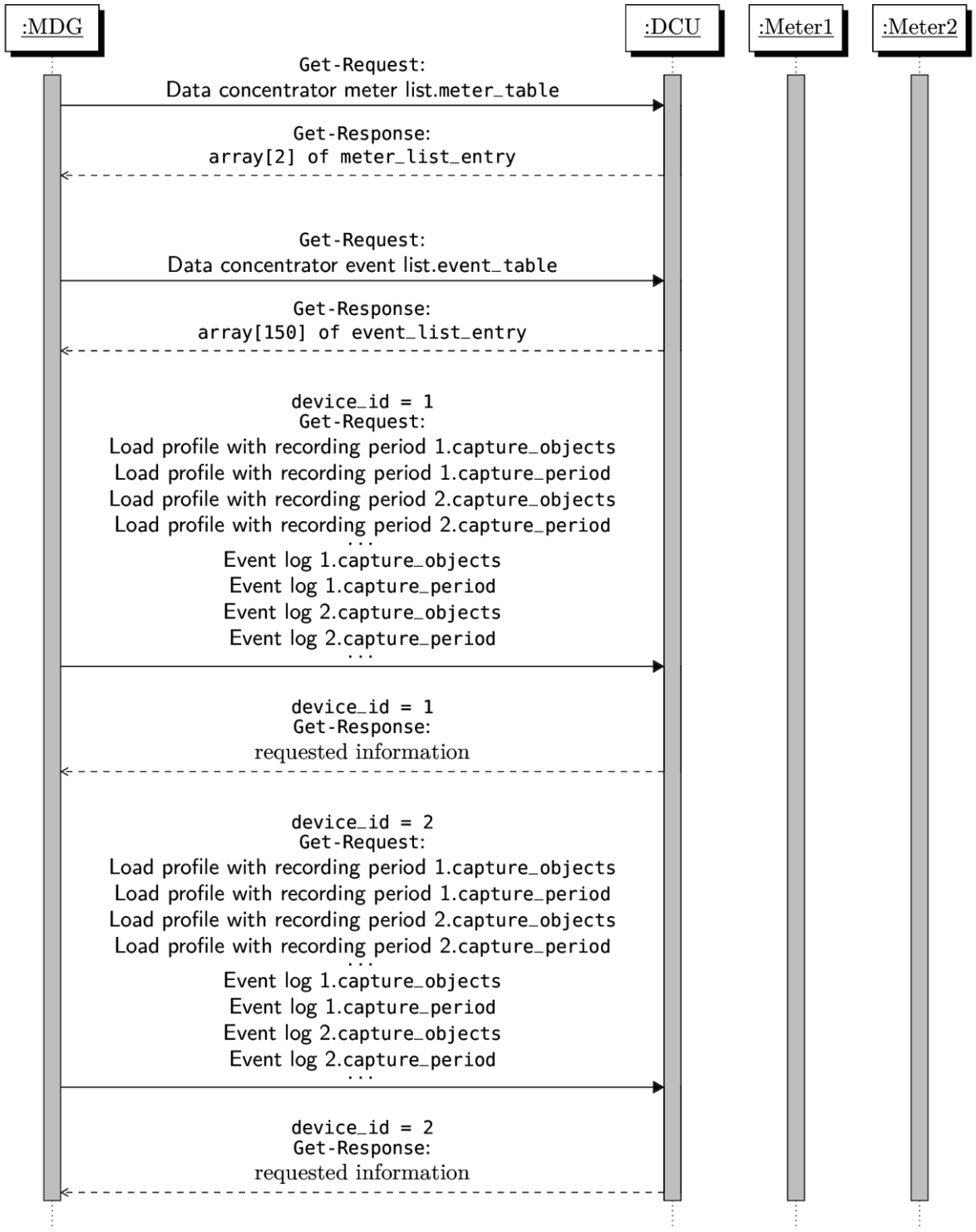
W tym rozdziale przedstawiono najbardziej typowe przykłady wykorzystania protokołu DCSAP w procesie prowadzenia akwizycji.

6.1 Rozpoczęcie sesji DCSAP z koncentratorem

System akwizycji w trakcie swojej pracy otwiera sesje DCSAP ze wszystkimi koncentratorami. Rozpoczęcie sesji polega na nawiązaniu połączenia TCP z koncentratorem. Po nawiązaniu połączenia system akwizycji pobiera listę liczników i odczytuje ich konfigurację (patrz Rys. 3). Jeżeli jest to kolejna sesja do danego koncentratora, system akwizycji może uzupełnić przyrostowo listę liczników tylko na tych pozycjach, w których wystąpiła zmiana (patrz Rys. 4). Jeżeli system akwizycji chce skorzystać z mechanizmu notyfikacji w trakcie trwania danej sesji, powinien włączyć ten mechanizm zaraz po nawiązaniu połączenia.

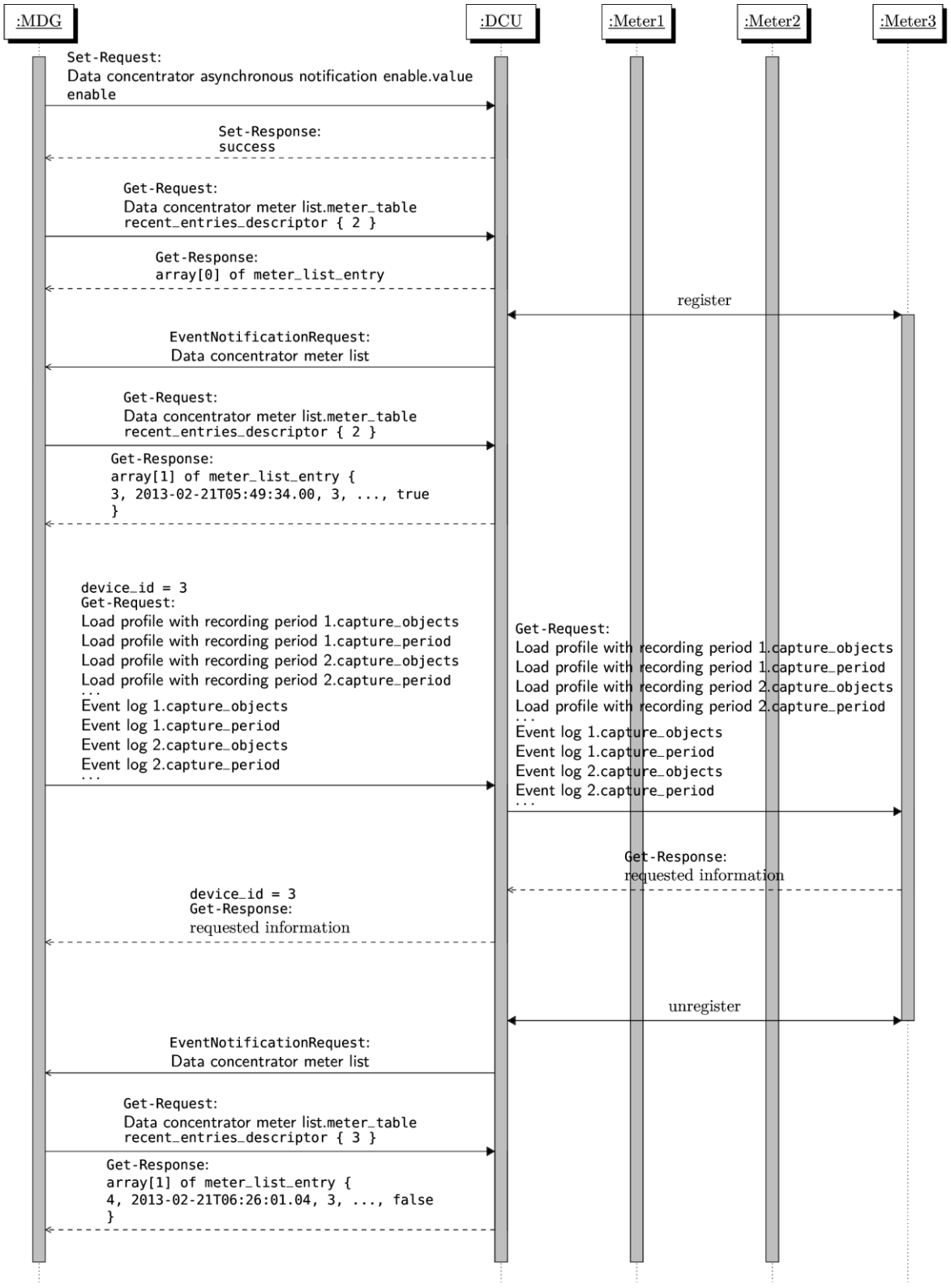
Jeżeli przez 5 minut nie są wysyłane polecenia do koncentratora, system akwizycji powinien wysłać pusty komunikat i oczekiwać na odpowiedź (patrz Rys. 5, Rys. 6). W ten sposób wykryte zostanie ewentualne zerwanie połączenia spowodowane nieplanowanym sprzętowym restartem koncentratora lub, w przypadku długotrwałego braku odpowiedzi, wstrzymanie połączenia spowodowane problemami natury telekomunikacyjnej (patrz Rys. 6). Brak odpowiedzi na pusty komunikat przez 5 minut skutkuje zamknięciem połączenia. Po zamknięciu lub zerwaniu połączenia system akwizycji próbuje nawiązać nowe w odstępach 3 minutowych.

Koncentrator powinien wykrywać długi czas bezczynności sesji (brak komunikatów z systemu akwizycji przez 10 minut lub dłużej) i zamykać ją zwalniając zasoby (patrz Rys. 6).



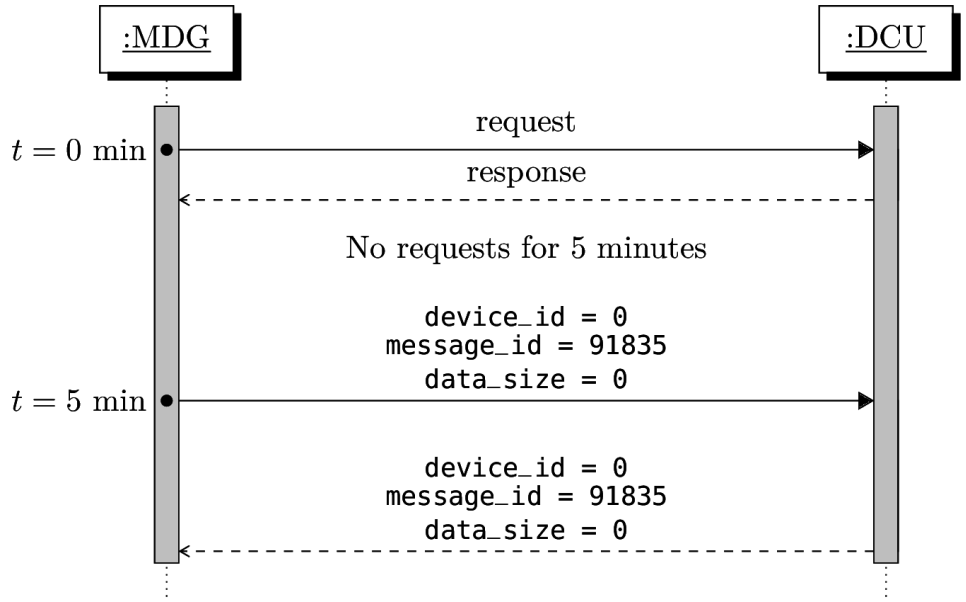
Rys. 3 Diagram sekwencji operacji wykonywanych przez system akwizycji po rozpoczęciu pierwszej sesji DCSAP z koncentratorem

Założono, że koncentrator wspiera akceleracją dostępu do konfiguracji liczników oraz że system akwizycji nie korzysta z mechanizmu notyfikacji.

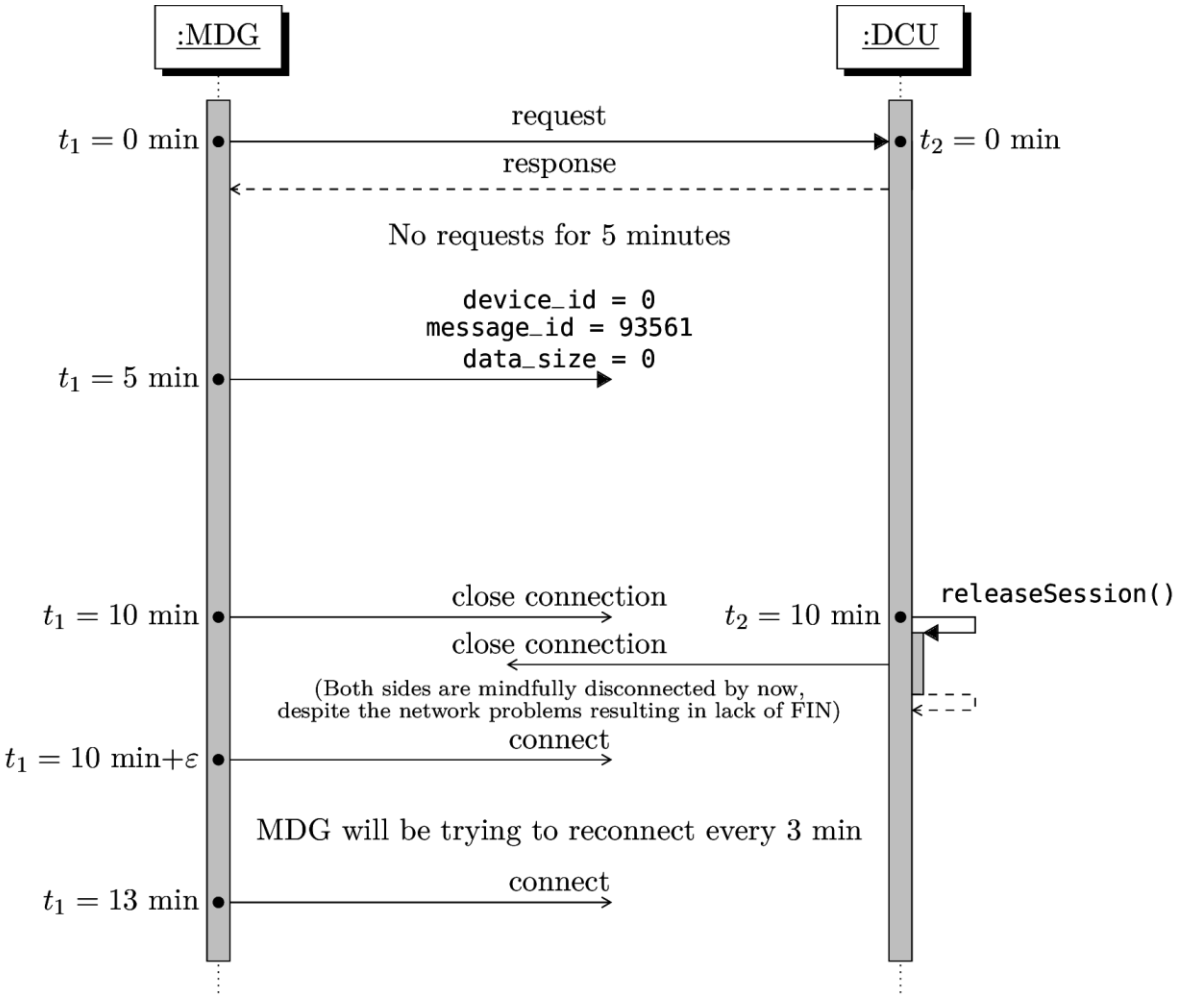


Rys. 4 Diagram sekwencji operacji wykonywanych przez system akwizycji po rozpoczęciu sesji DCSAP z koncentratorem

Założono, że koncentrator komunikuje się z licznikami w warstwie aplikacji za pomocą protokołu DLMS.



Rys. 5 Diagram sekwencji podtrzymywania połączenia z koncentratorem



Rys. 6 Diagram sekwencji próby podtrzymywania połączenia z koncentratorem i bezczynności sesji w przypadku problemów z łączem

Założono, że w trakcie braku komunikacji parametry łączności uległy pogorszeniu do poziomu uniemożliwiającego komunikację.

6.2 Zamknięcie sesji DCSAP

Zamknięcie sesji następuje po zamknięciu połączenia TCP.

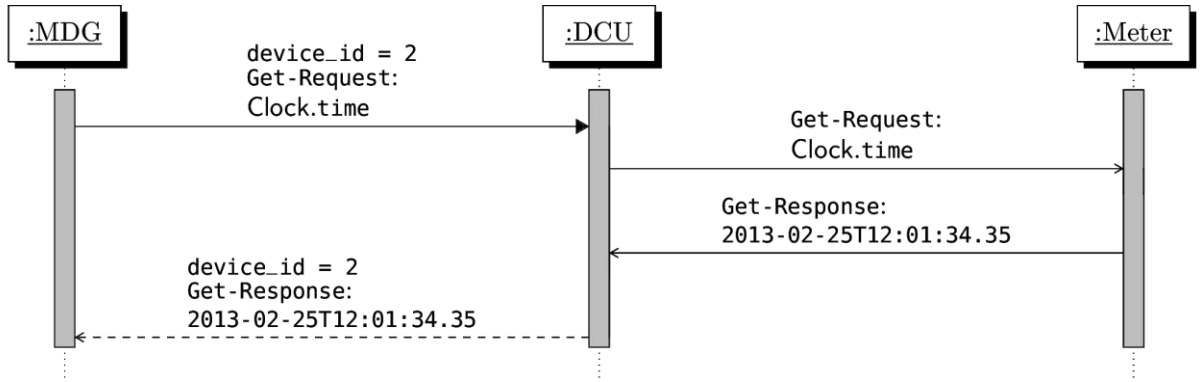
6.3 Synchronizacja topologii

W trakcie trwania sesji system akwizycji synchronizuje topologię poprzez wysłanie polecenia pobrania listy liczników. W celu minimalizacji przesyłanych danych, system akwizycji stosuje wybór selektywny podając najwyższy znany numer sekwencyjny zmiany rekordu. Dzięki temu koncentrator odpowie tylko takim podzbiorem rekordów, które uległy zmianie (tj. rekordów o numerze sekwencyjnym ostatniej zmiany większym od zadanego).

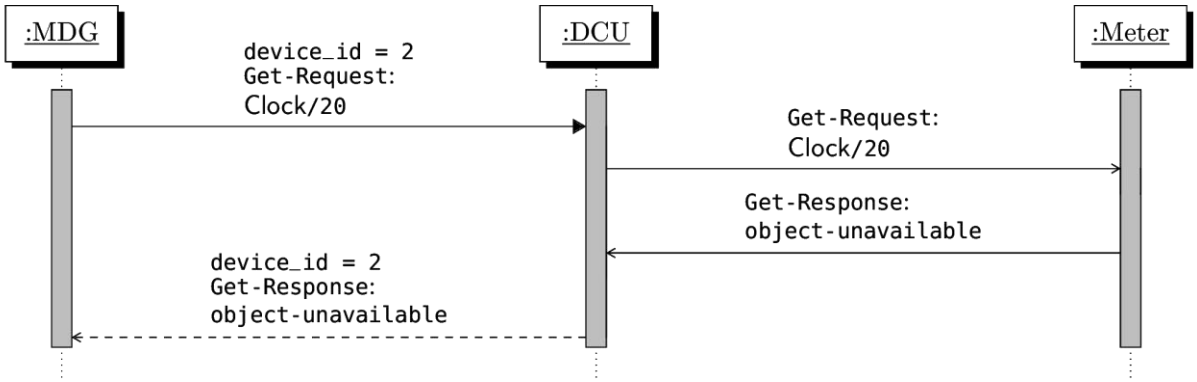
Istnieją dwie metody wyzwalania aktualizacji topologii. System akwizycji może cyklicznie sprawdzać czy są zaktualizowane rekordy lub reagować na asynchroniczną notyfikację (patrz Rys. 4).

6.4 Odczyt rejestru układu

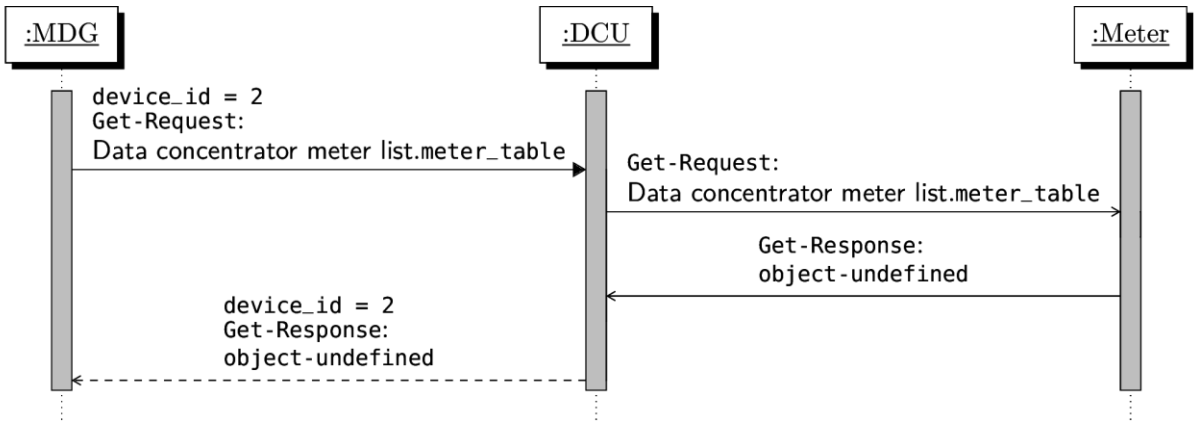
Po nawiązaniu sesji z koncentratorem i pobraniu listy liczników system akwizycji może kierować polecenia do liczników podłączonych do danego koncentratora. W tym celu przygotowuje komunikat polecenia. Ustawia identyfikator licznika odczytany z listy, unikalny identyfikator komunikatu oraz wypełnia dane DLMS poleceniem *Get-Request* ze wskazaniem na konkretny rejestr układu. Tak przygotowany komunikat przesyła do koncentratora, po czym oczekuje na odpowiedź (patrz Rys. 7, Rys. 8, Rys. 9, Rys. 10, Rys. 13).



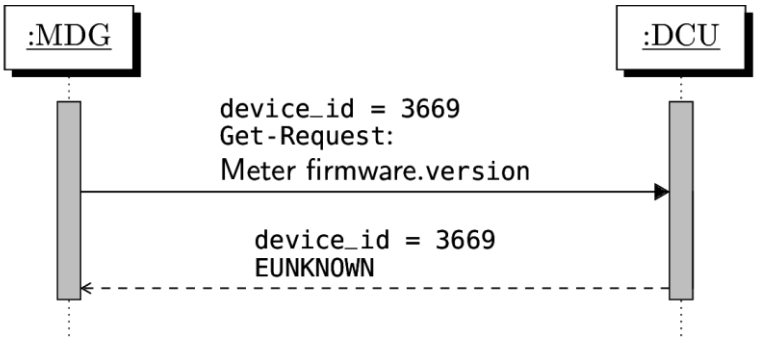
Rys. 7 Diagram sekwencji odczytu z licznika – wariant pomyślny



Rys. 8 Diagram sekwencji odczytu z licznika – wariant nieprawidłowego atrybutu



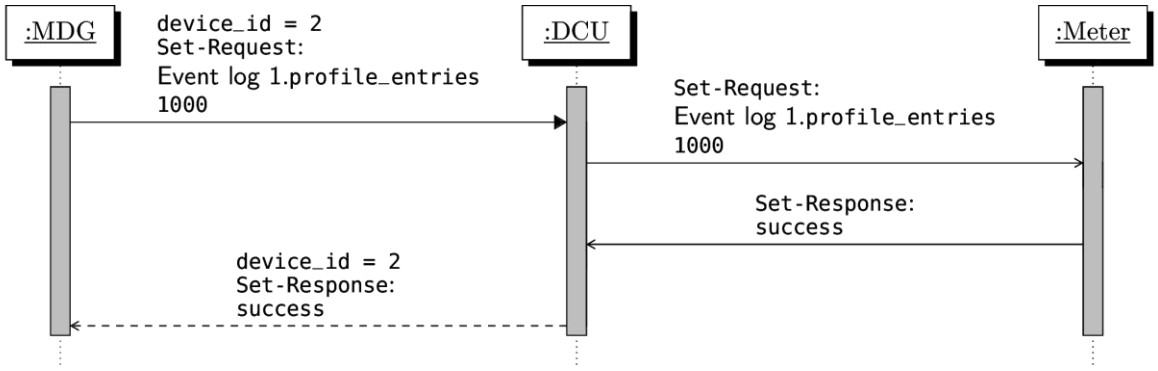
Rys. 9 Diagram sekwencji odczytu z licznika – wariant nieistniejącego obiektu



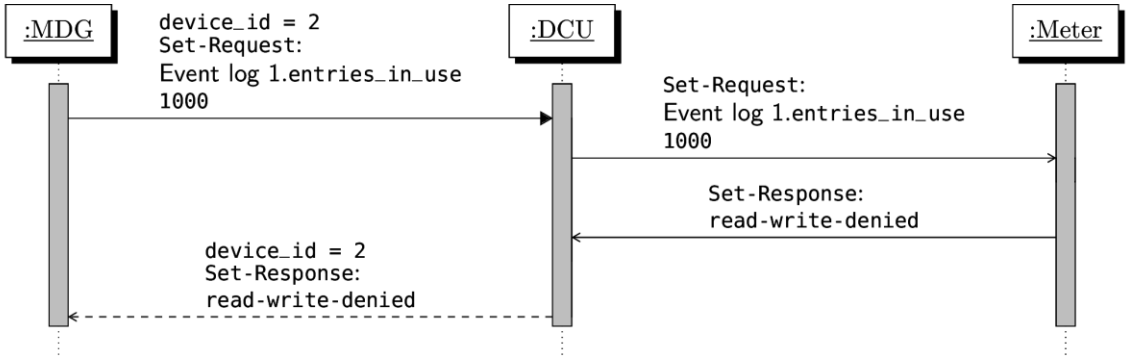
Rys. 10 Diagram sekwencji odczytu z licznika – wariant nieznanego identyfikatora licznika

6.5 Zapis rejestru układu

Zapis realizowany jest podobnie jak odczyt. Wysyłany jest komunikat z poleceniem *Set-Request* w danych DLMS. Przesłana odpowiedź potwierdzi wykonanie operacji (patrz Rys. 11) lub nie (patrz Rys. 12).



Rys. 11 Diagram sekwencji zapisu do licznika – wariant pomyślny



Rys. 12 Diagram sekwencji zapisu do licznika – wariant niepomyślny

6.6 Konfiguracja układu

Konfiguracja układu polega na wysłaniu zestawu poleceń zapisu kompletu rejestrów danego układu. W przypadku niekompletnego potwierdzenia lub selektywnego niepowodzenia można ponowić cały zestaw operacji lub w celu optymalizacji wyłączyć z tego zestawu te operacje, na które przyszło potwierdzenie.

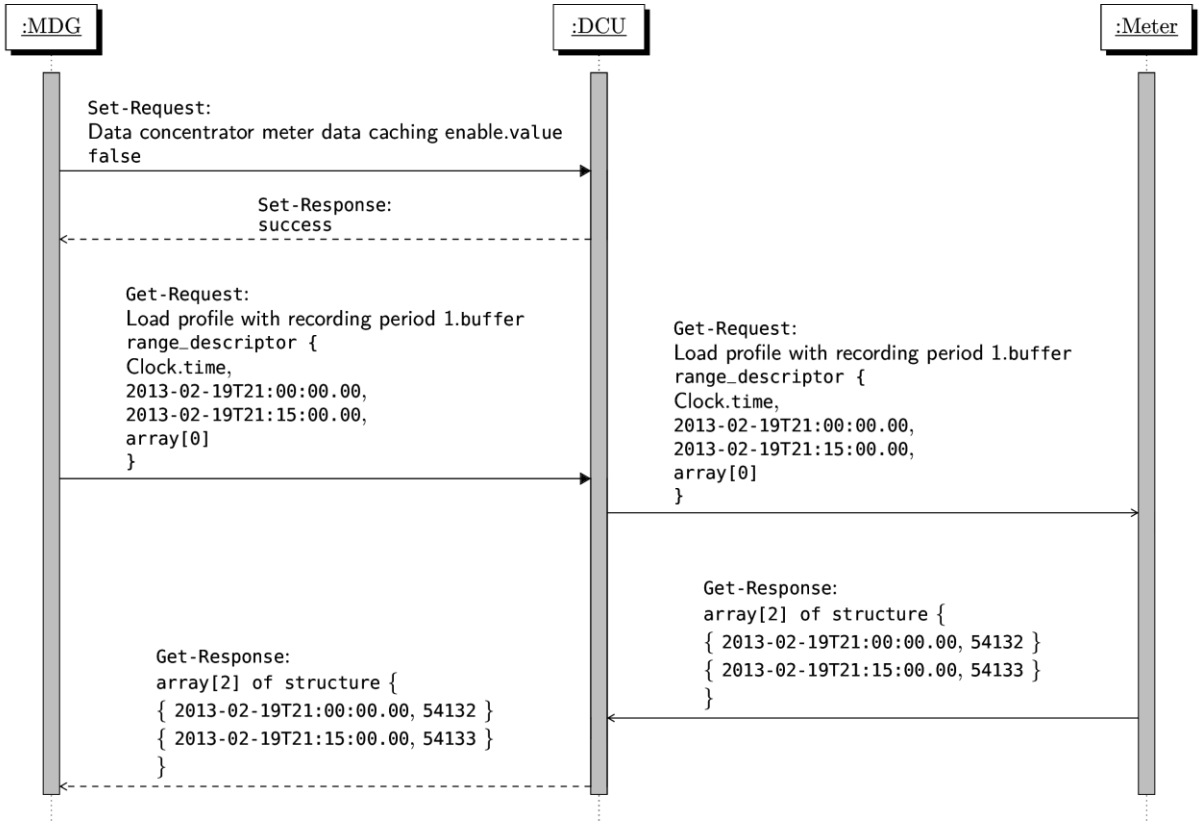
6.7 Pobieranie parametrów konfiguracyjnych układu

Pobieranie parametrów konfiguracyjnych polega na wywołaniu sekwencji odczytów kolejnych rejestrów stanowiących elementy konfiguracji układu. (patrz Rys. 3 dla przypadku

akcelerowanego dostępu do licznika lub Rys. 4 dla przypadku gdy koncentrator jeszcze sam nie pobrał konfiguracji z licznika).

6.8 Bezpośrednia komunikacja z układem

Pomimo zastosowania w koncentratorach różnych technik buforowania odpowiedzi od liczników, możliwe jest wymuszenie (np. w celach diagnostycznych) bezpośredniej komunikacji z licznikiem, z pominięciem tych mechanizmów. W tym celu należy poleceniem *Set-Request* przestawić atrybut *value* obiektu *Meter data caching enable* na *false*, wyłączając tym samym buforowanie odpowiedzi liczników. Odpowiedzi na polecenia adresowane do konkretnych liczników będą pochodzić teraz z liczników (patrz Rys. 13). Wyłączenie buforowania obowiązuje tylko w ramach tej sesji, do jej zakończenia lub ponownego włączenia buforowania.



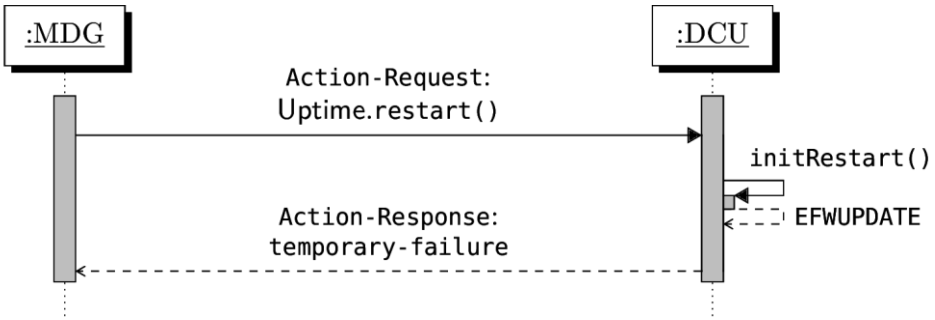
Rys. 13 Diagram sekwencji zapisu do koncentratora i odczytu bezpośredniego z układu

Założono, że obiekt *Load profile with recording period 1* zatrzaskuje tylko czas i wartość A+.

6.9 Restart koncentratora

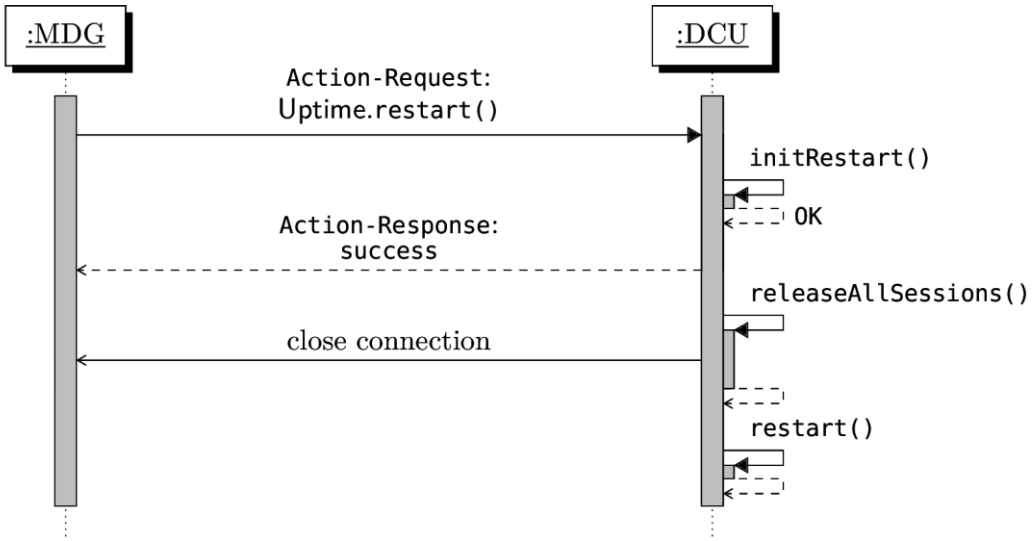
Procedurę restartu koncentratora inicjuje się poprzez wywołanie metody *restart()* obiektu *Uptime* (0-100:2.1.0*255). Polecenie to zwraca błąd jeżeli trwa obecnie aktualizacja koncentratora

lub licznika (patrz Rys. 14). W przeciwnym razie zwracany jest kod sukcesu, a koncentrator zwalnia wszystkie sesje i restartuje się (patrz Rys. 15).



Rys. 14 Diagram sekwencji restartowania koncentratora – wariant niepomyślny

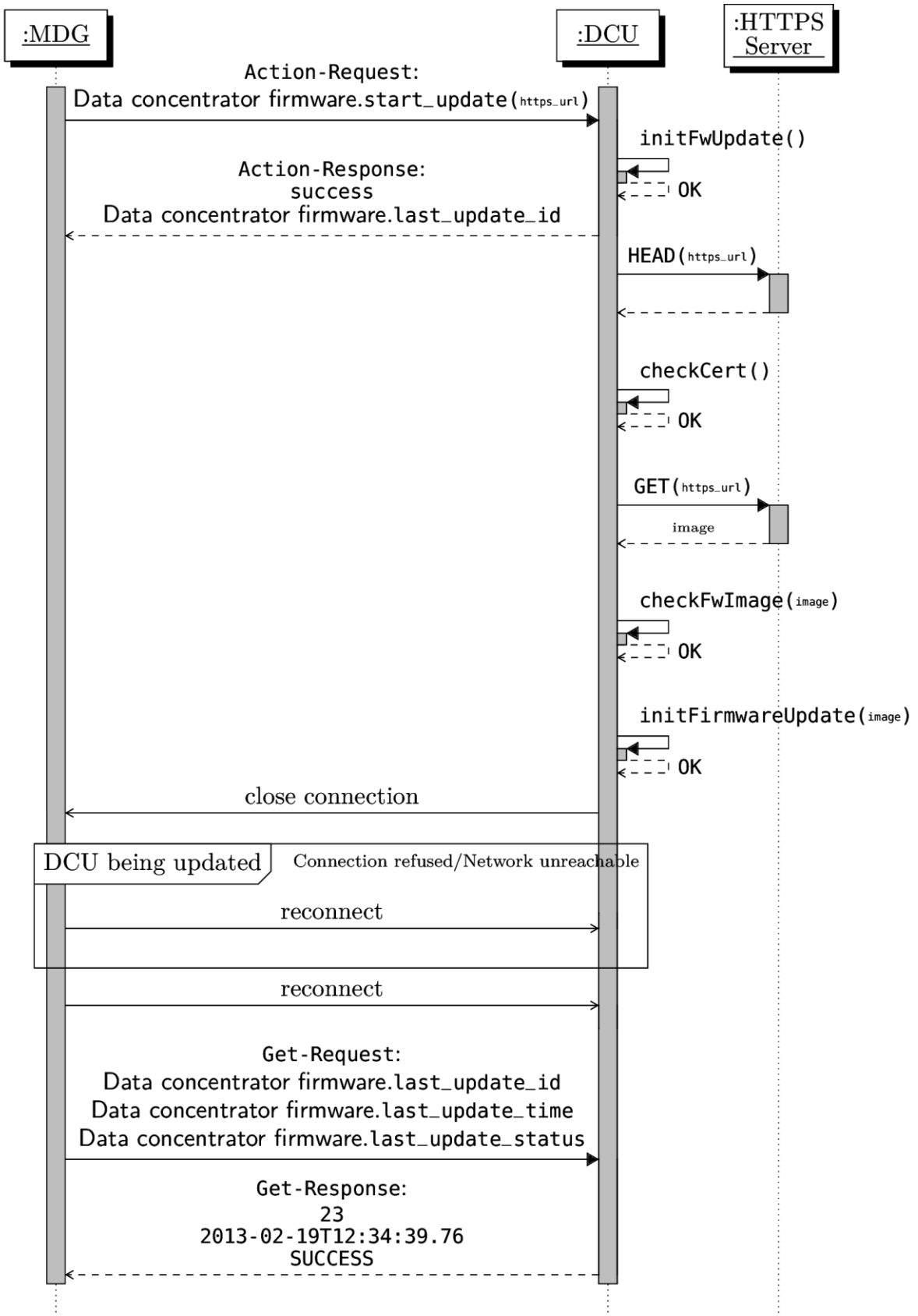
Założono, że przed próbą restartu koncentratora zlecono operację aktualizacji oprogramowania, która jeszcze się nie zakończyła.



Rys. 15 Diagram sekwencji restartowania koncentratora – wariant pomyślny

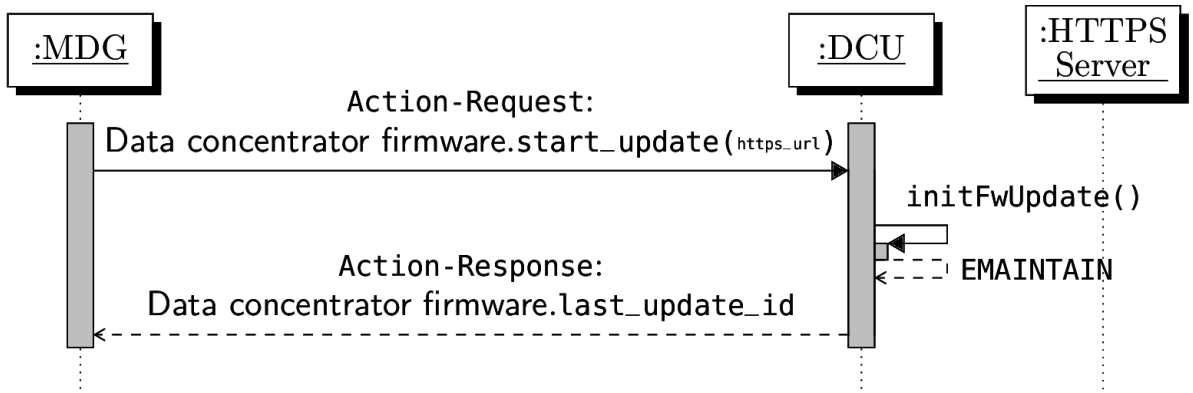
6.10Aktualizacja oprogramowania koncentratora

W celu zaktualizowania oprogramowania na koncentratorze system akwizycji wysyła komunikat z poleceniem wykonania metody *start_update()* obiektu *Data concentrator firmware*, przekazując adres URL, z którego koncentrator pobiera obraz aktualizacji protokołem HTTPS. Po pobraniu obrazu, koncentrator sprawdza jego poprawność. Następnie rozpoczyna aktualizację, a po jej zakończeniu – restartuje się. System akwizycji po odebraniu komunikatu o pomyślnej inicjacji aktualizacji, wykrywa zamknięcie połączenia, po czym otwiera nową sesję, z poziomu której sprawdza datę rozpoczęcia ostatniej aktualizacji i jej status. Cały proces i możliwe niepowodzenia obrazują diagramy sekwencji przedstawione na Rys. 16, Rys. 17, Rys. 18, Rys. 19, Rys. 20, Rys. 21, Rys. 22.



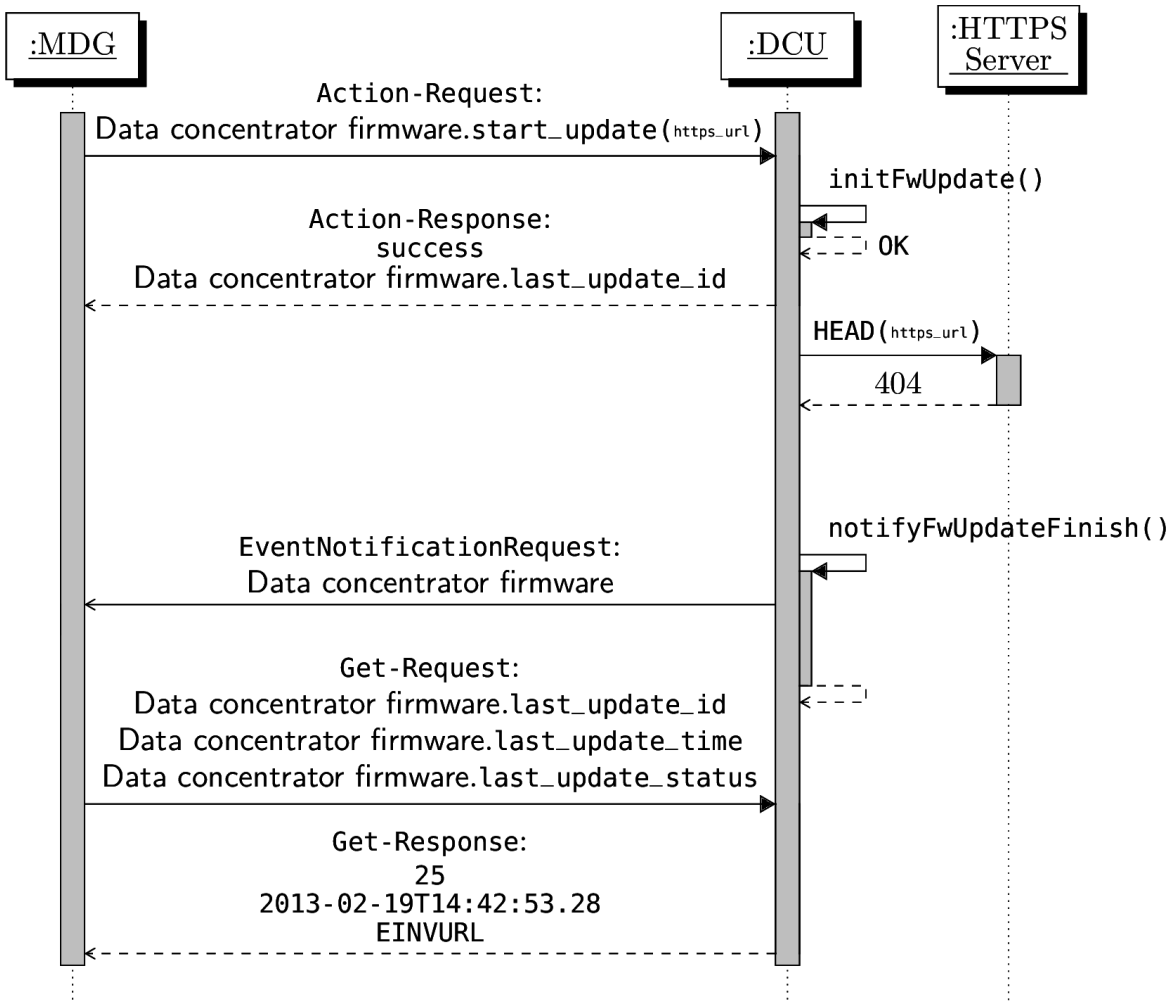
Rys. 16 Diagram sekwencji aktualizacji oprogramowania koncentratora – wariant pomyślny

Założono, że koncentrator nie wspiera podtrzymywania sesji w trakcie aktualizacji i wszystkie kroki przebiegły pomyślnie.



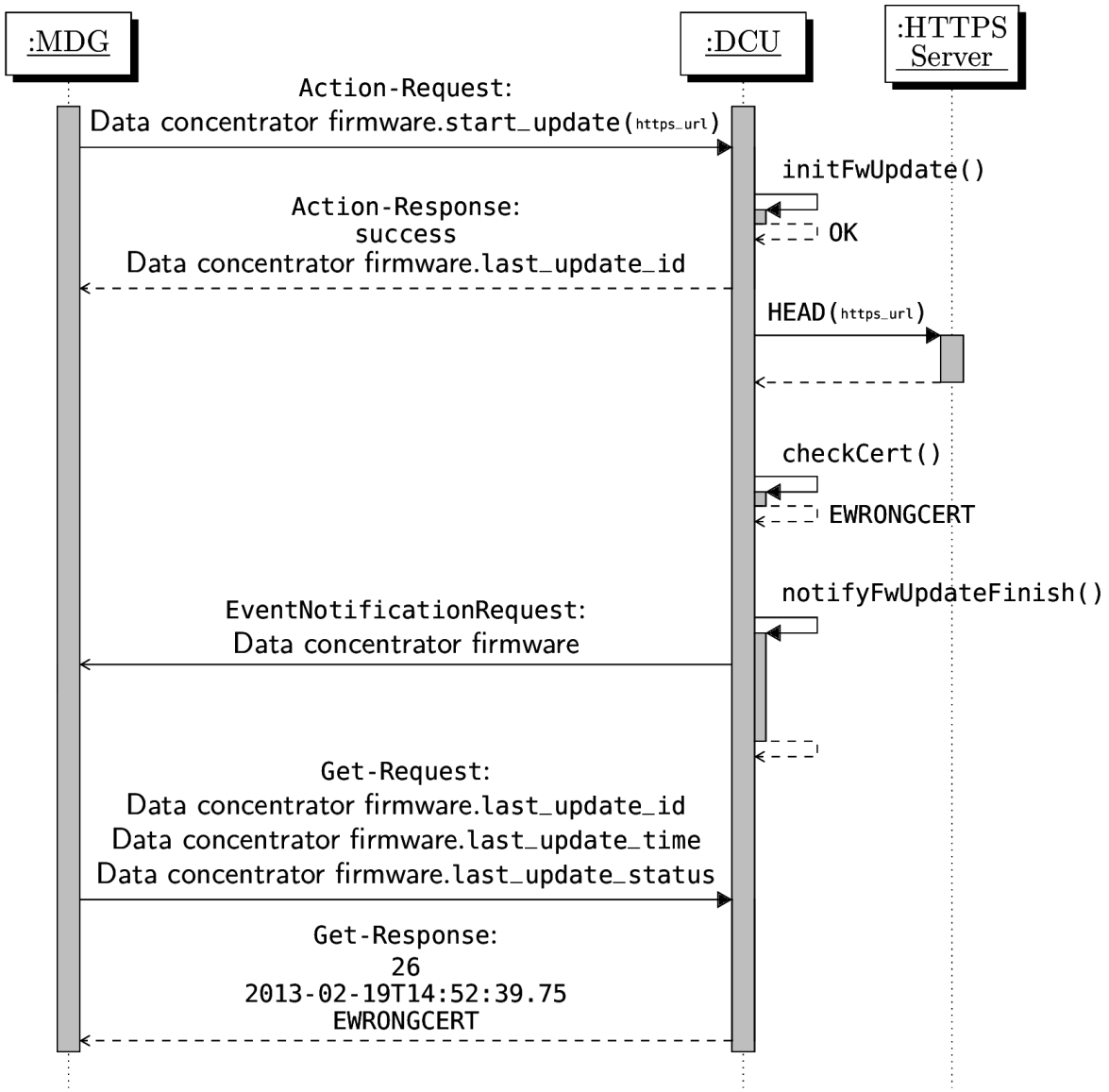
Rys. 17 Diagram sekwencji aktualizacji oprogramowania koncentratora – 1. wariant niepomyślny

Założono, że rozpoczęto wcześniej już inną aktualizację, która jeszcze się nie zakończyła.



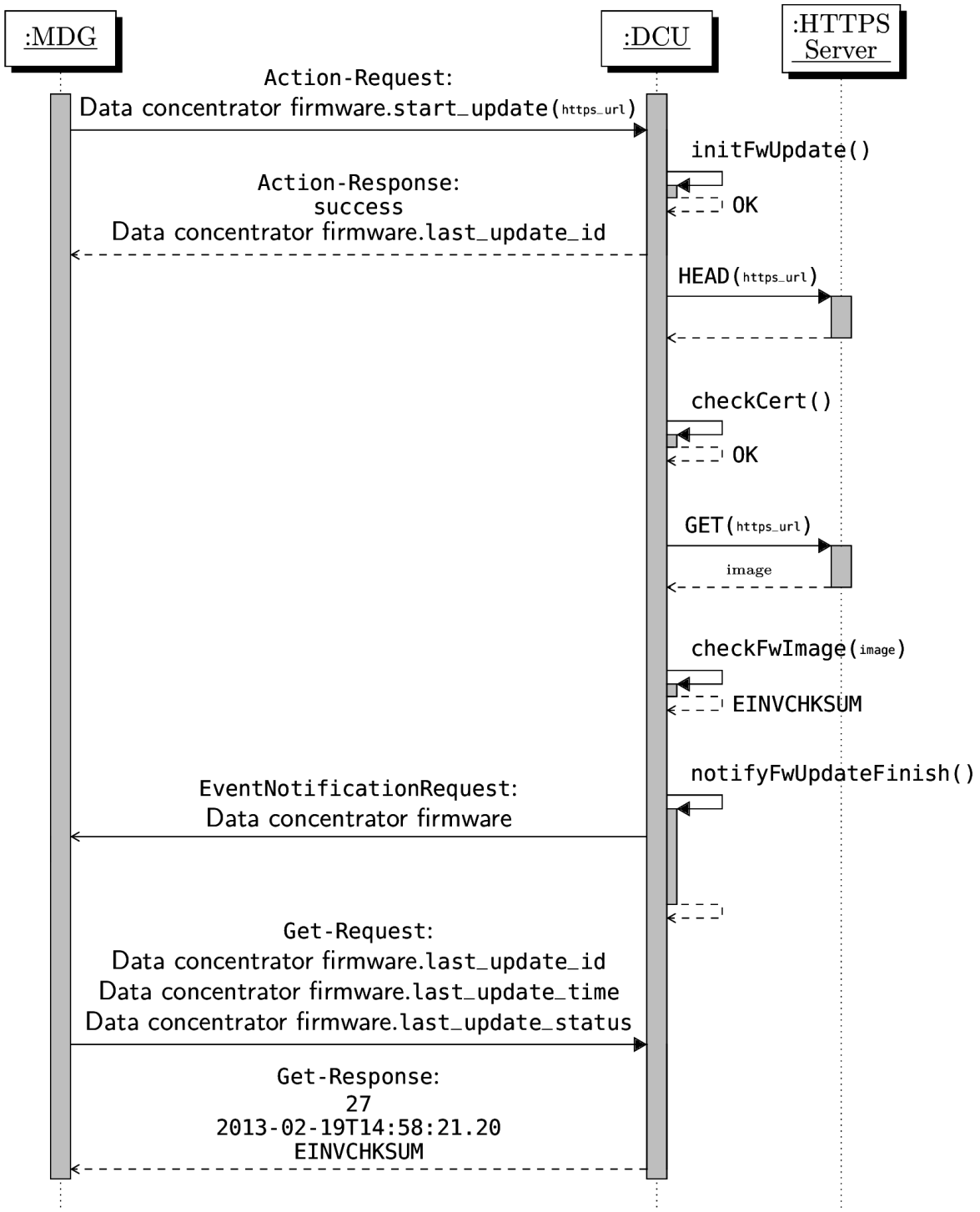
Rys. 18 Diagram sekwencji aktualizacji oprogramowania koncentratora – 2. wariant niepomyślny

Założono, że podany adres obrazu jest nieprawidłowy.



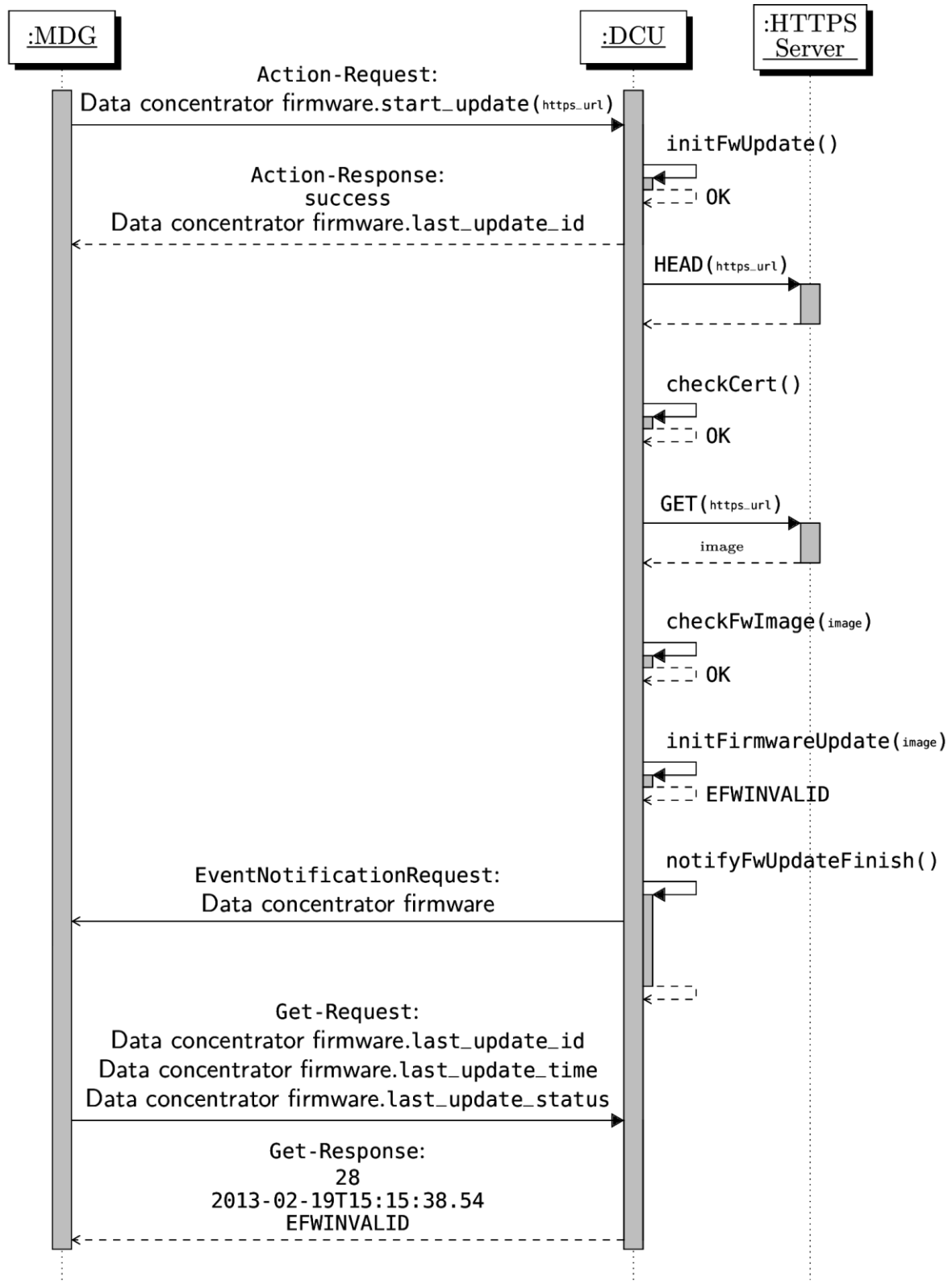
Rys. 19 Diagram sekwencji aktualizacji oprogramowania koncentratora – 3. wariant niepomyślny

Założono, że certyfikat, który zwraca serwer HTTPS, jest nieprawidłowy.



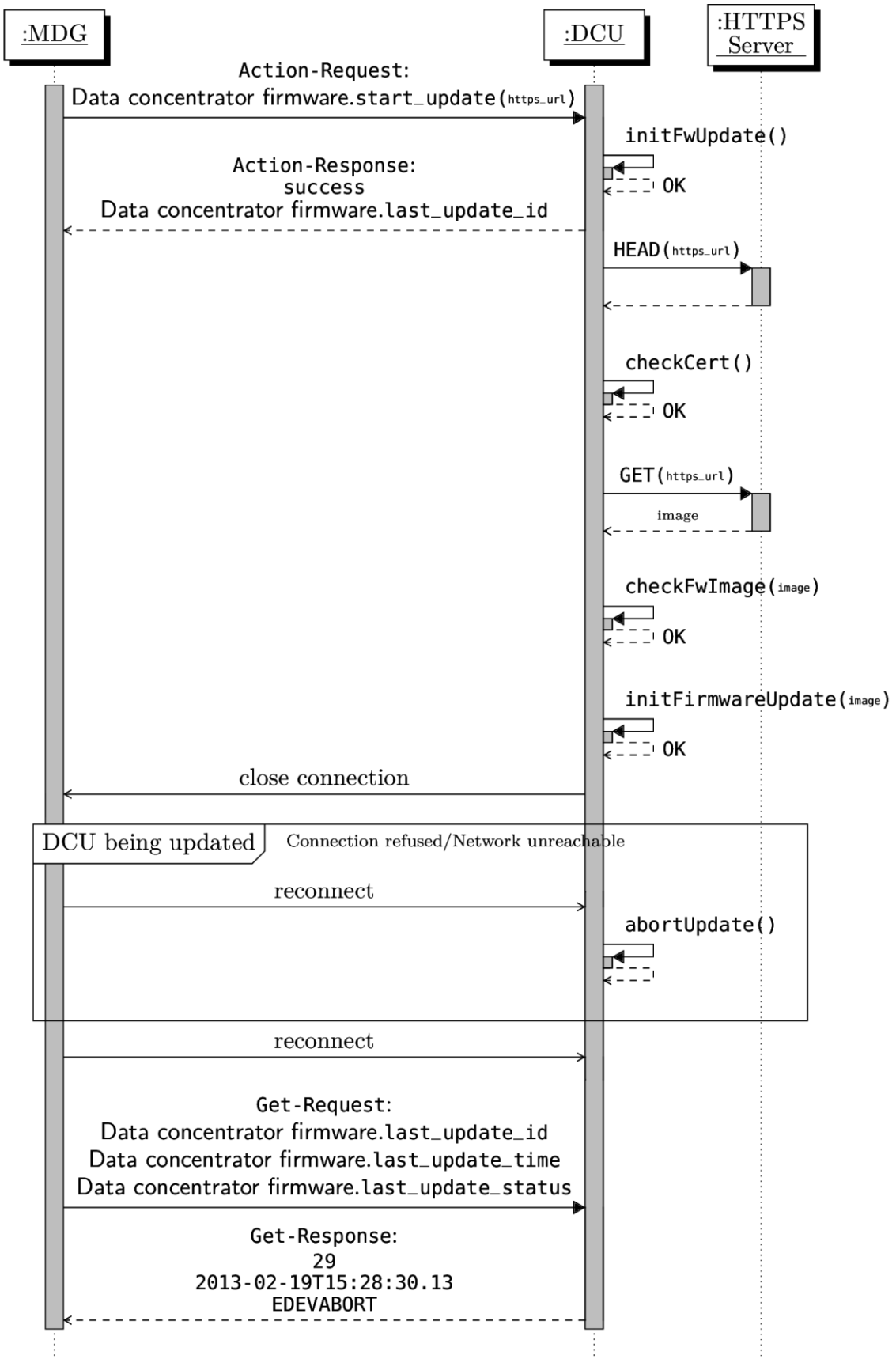
Rys. 20 Diagram sekwencji aktualizacji oprogramowania koncentratora – 4. wariant niepomyślny

Założono, że plik obrazu pobrany z serwera HTTPS jest uszkodzony (ma nieprawidłową sumę kontrolną).



Rys. 21 Diagram sekwencji aktualizacji oprogramowania koncentratora – 5. wariant niepomyślny

Założono, że wskazany obraz nie jest obsługiwany przez koncentrator.

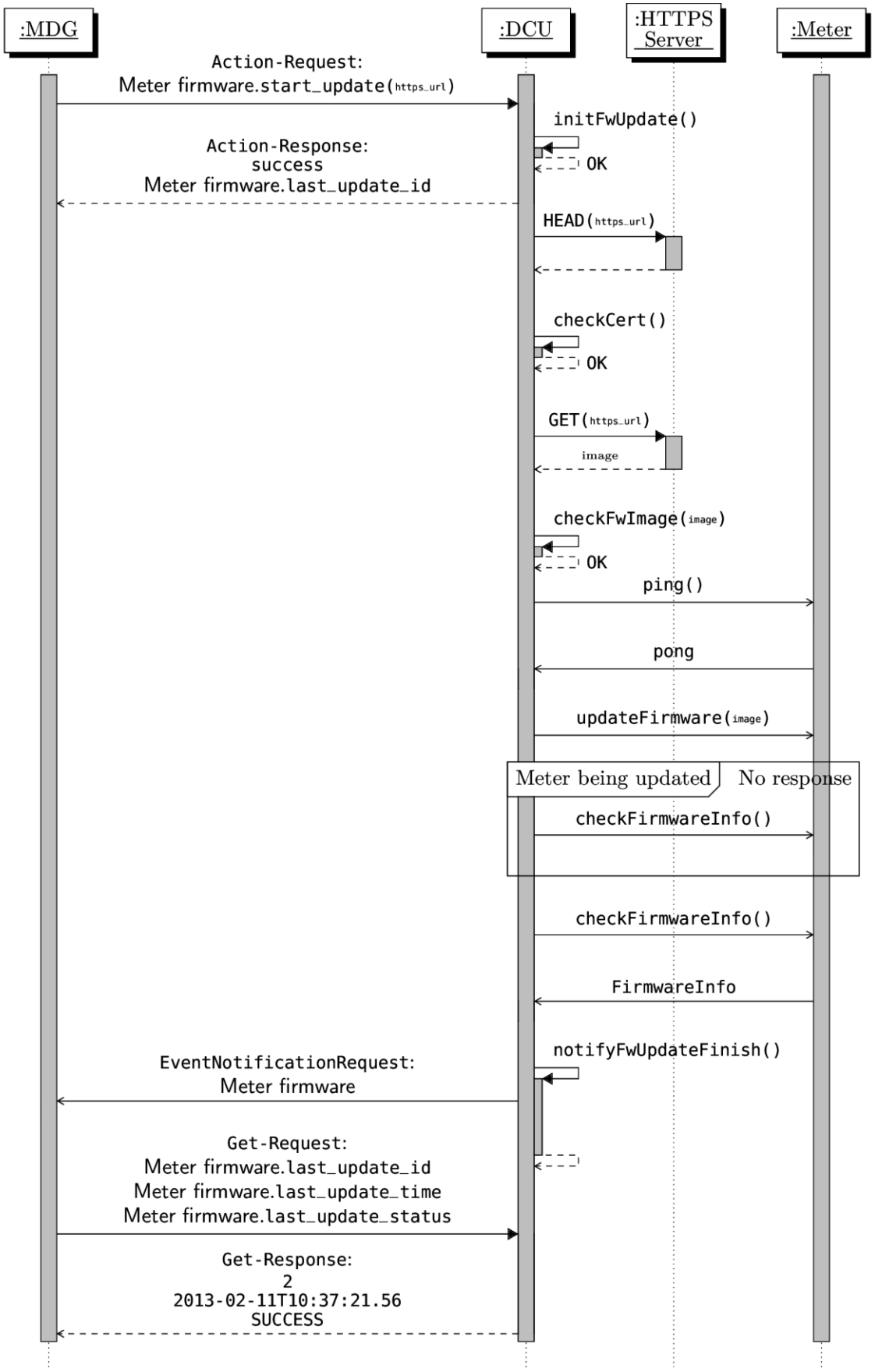


Rys. 22 Diagram sekwencji aktualizacji oprogramowania koncentratora – 6. wariant niepomyślny

Założono, że koncentrator nie wspiera podtrzymywania sesji w trakcie aktualizacji oraz że przerwał samą aktualizację.

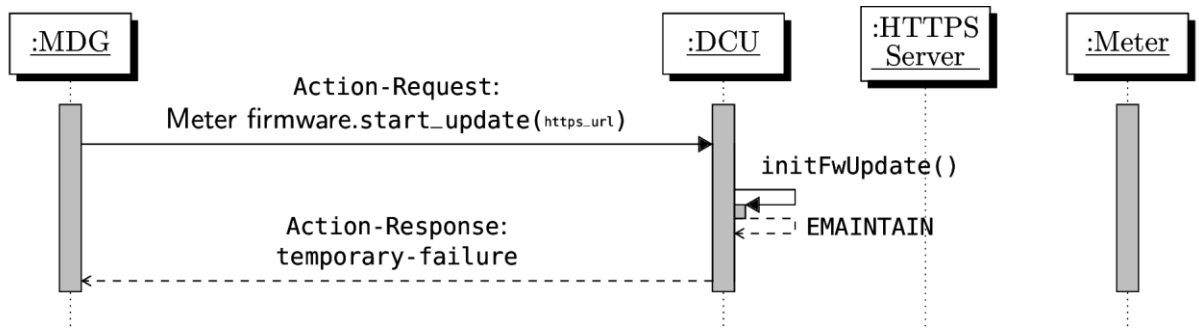
6.11 Aktualizacja oprogramowania licznika

Aktualizacja oprogramowania licznika (patrz Rys. 23, Rys. 24, Rys. 25, Rys. 26, Rys. 27, Rys. 28, Rys. 29, Rys. 30) odbywa się analogicznie do aktualizacji oprogramowania koncentratora, z tą różnicą, że polecenie aktualizacji adresowane jest do licznika i obiektu *Meter firmware*. Obraz aktualizacji musi w takim przypadku pasować do wskazanego licznika. Zadaniem koncentratora jest odpowiedzenie systemowi akwizycji sukcesem i nowym (już zinkrementowanym) identyfikatorem ostatniej (czyli właśnie trwającej) aktualizacji, jeżeli nie trwa obecnie żadna inna aktualizacja. Następnie odbywa się pobranie tego obrazu, sprawdzenie jego poprawności (o ile producent koncentratora jest w stanie je przeprowadzić), przekazanie obrazu do licznika i właściwe wyzwolenie procesu aktualizacji. Na koniec wysyłane jest powiadomienie do systemu akwizycji informujące o zakończeniu aktualizacji, po otrzymaniu którego system akwizycji sprawdza status ostatniej aktualizacji.



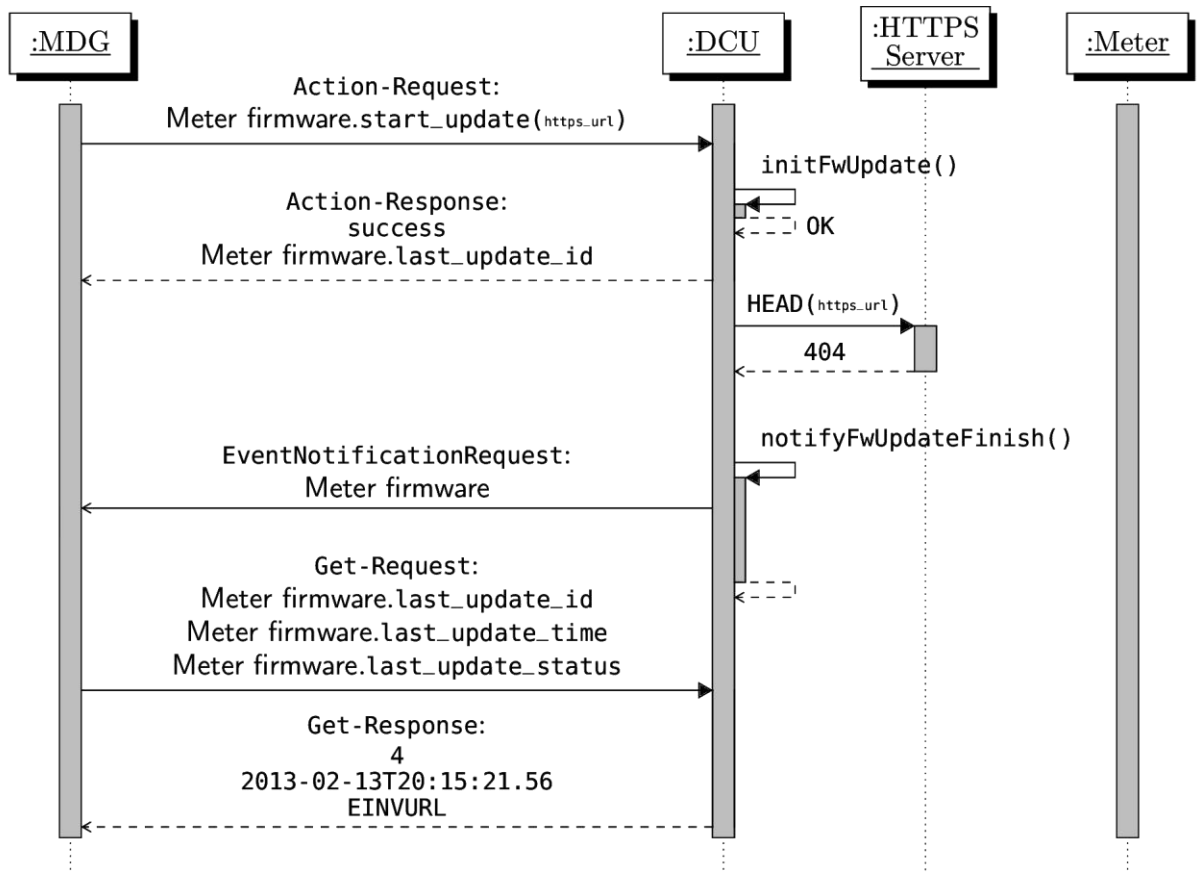
Rys. 23 Diagram sekwencji aktualizacji oprogramowania licznika – wariant pomyślny

Założono, że wszystkie kroki przebiegły pomyślnie.



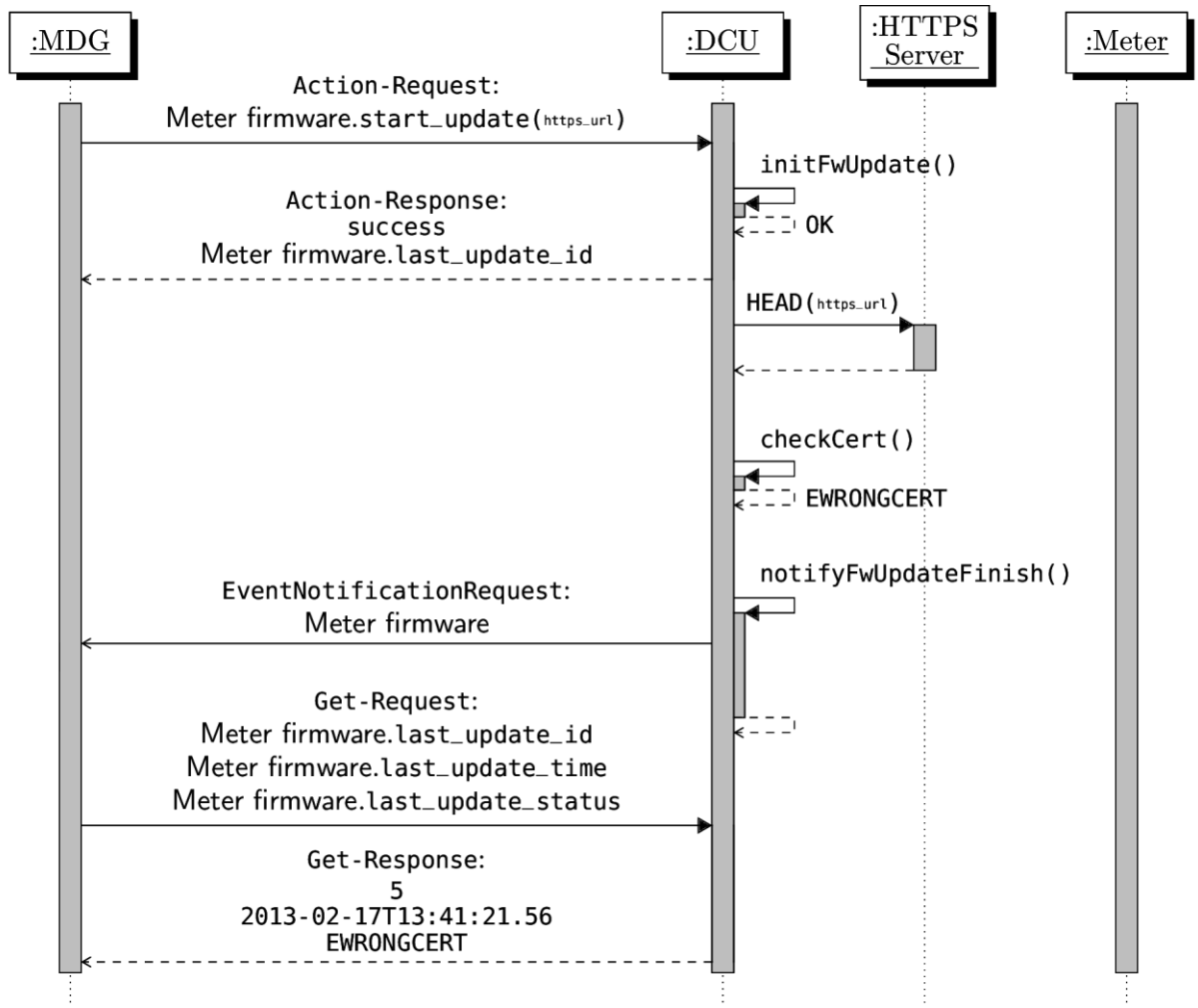
Rys. 24 Diagram sekwencji aktualizacji oprogramowania licznika – 1. wariant niepomyślny

Założono, że rozpoczęto wcześniej już inną aktualizację, która jeszcze się nie zakończyła.



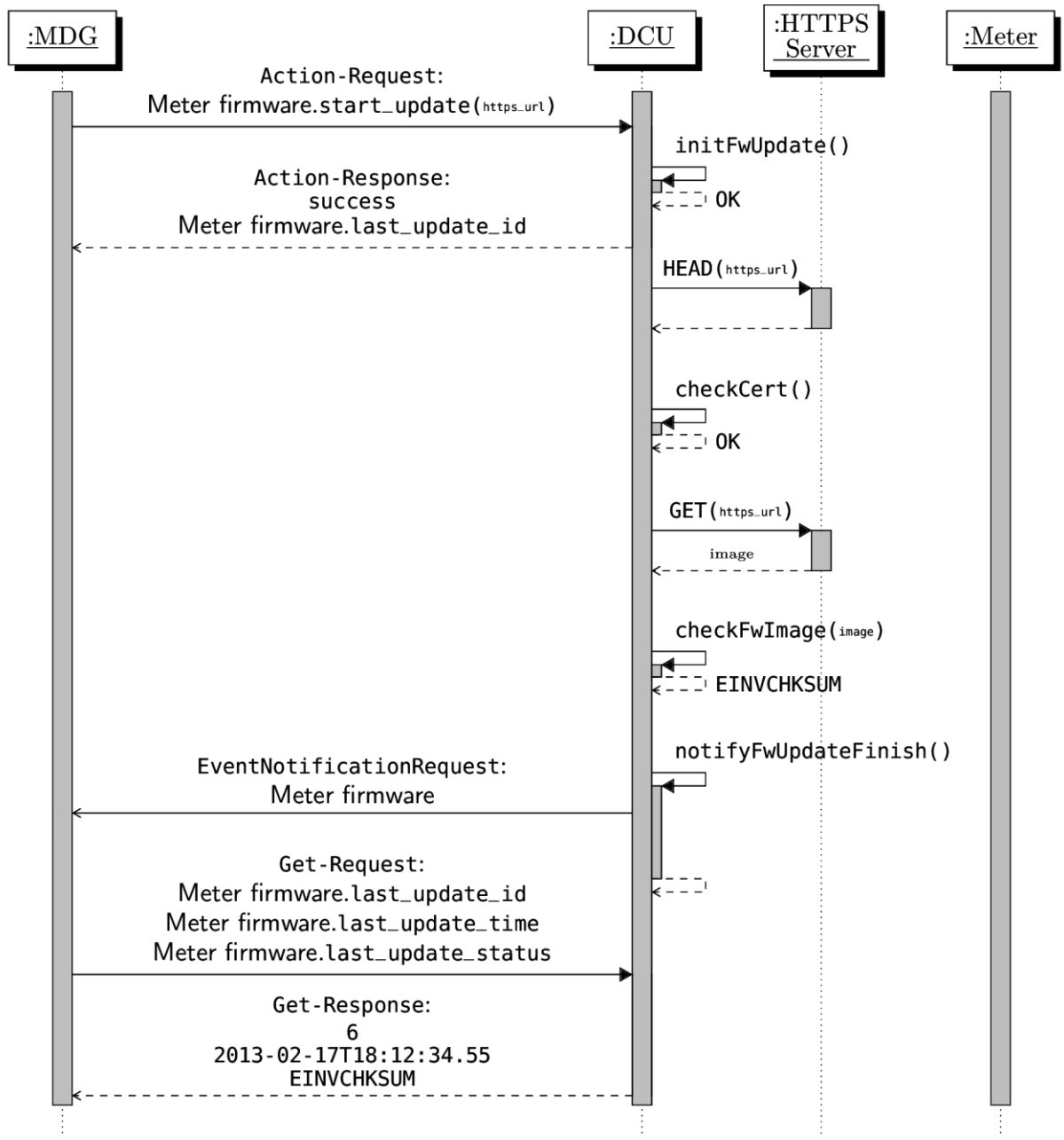
Rys. 25 Diagram sekwencji aktualizacji oprogramowania licznika – 2. wariant niepomyślny

Założono, że podany adres obrazu jest nieprawidłowy.



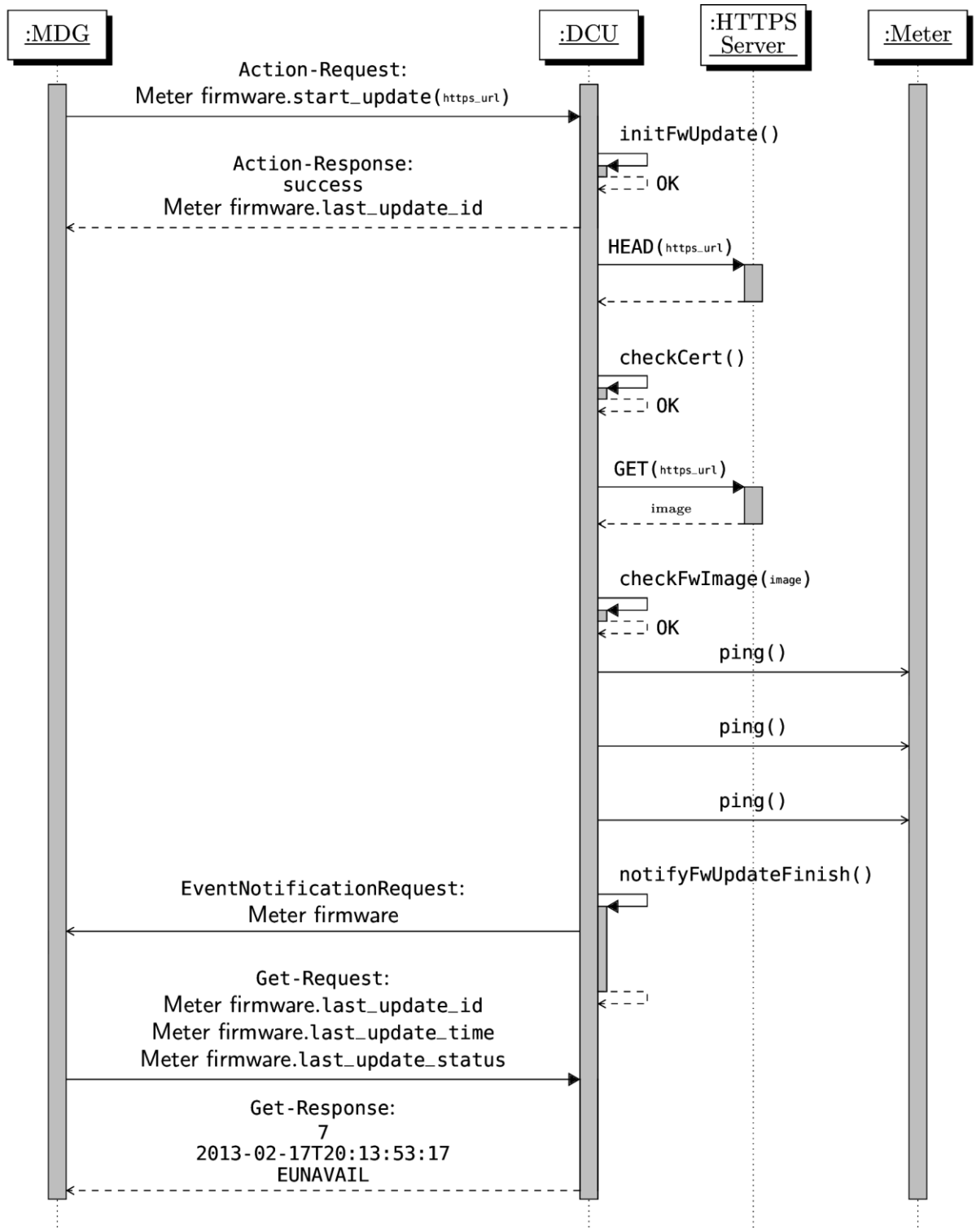
Rys. 26 Diagram sekwencji aktualizacji oprogramowania licznika – 3. wariant niepomyślny

Założono, że certyfikat, który zwraca serwer HTTPS, jest nieprawidłowy.



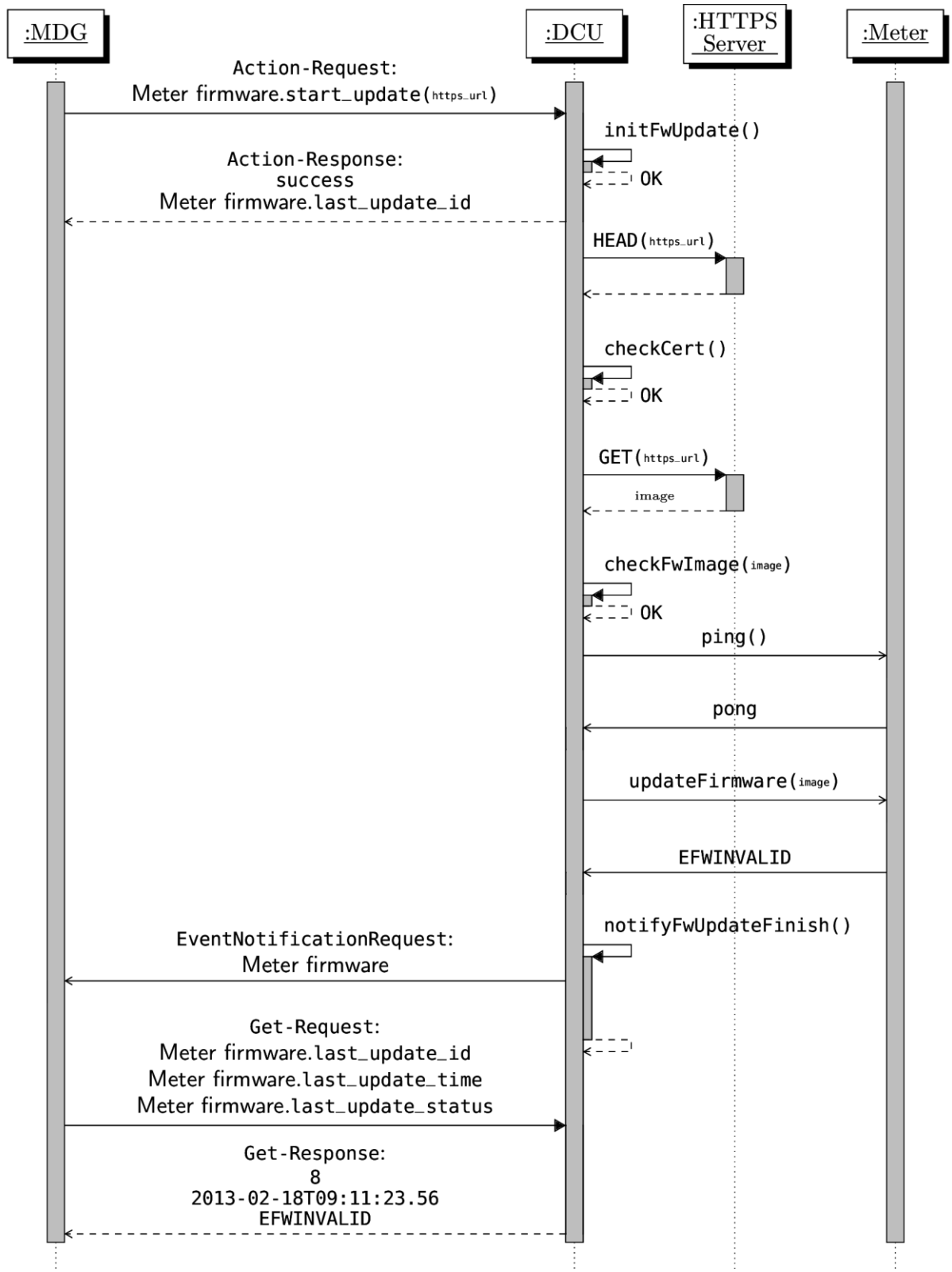
Rys. 27 Diagram sekwencji aktualizacji oprogramowania licznika – 4. wariant niepomyślny

Założono, że plik obrazu pobrany z serwera HTTPS jest uszkodzony (ma nieprawidłową sumę kontrolną).



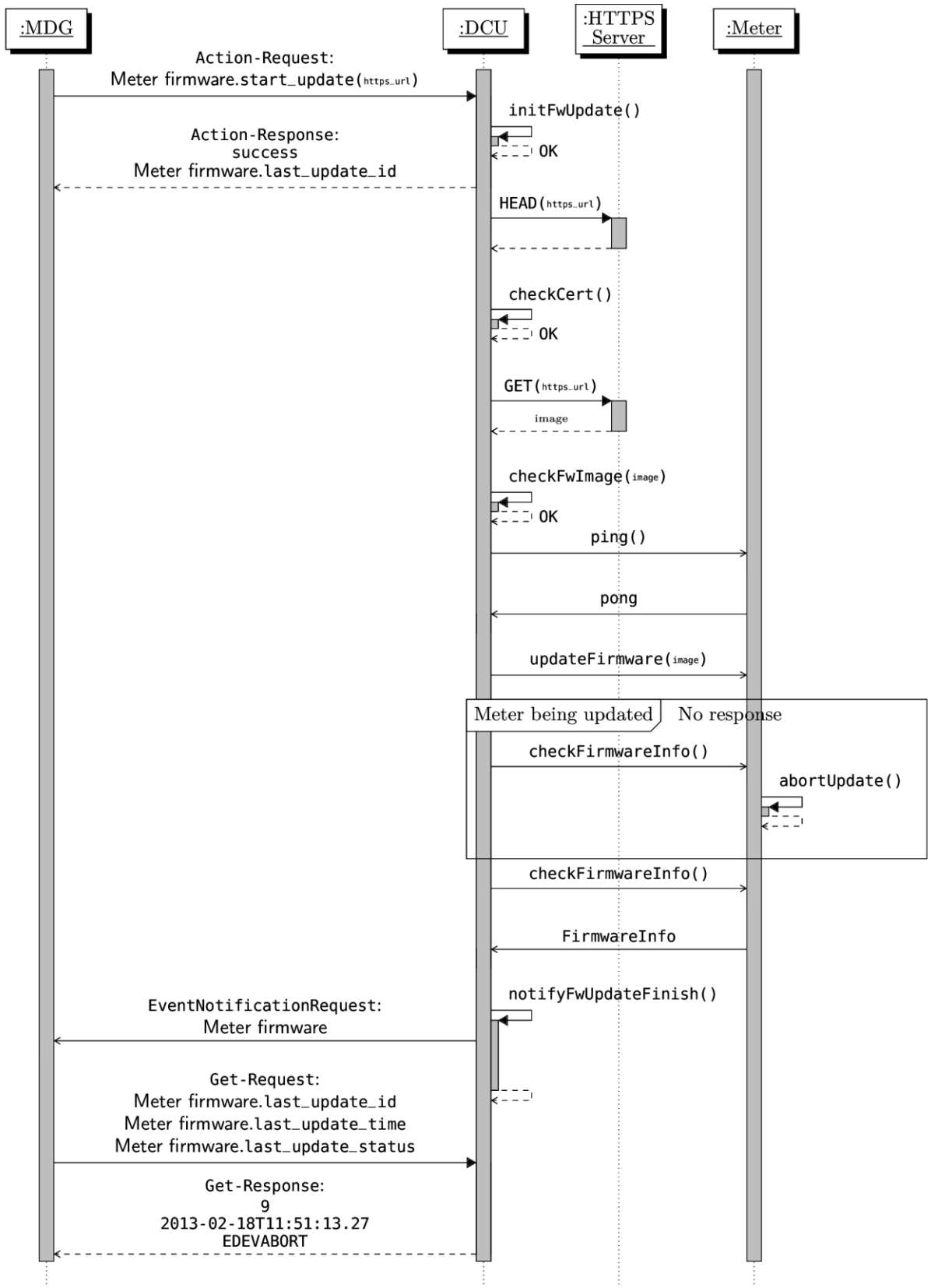
Rys. 28 Diagram sekwencji aktualizacji oprogramowania licznika – 5. wariant niepomyślny

Założono, że licznik nie jest dostępny w trakcie próby przeprowadzenia aktualizacji.



Rys. 29 Diagram sekwencji aktualizacji oprogramowania licznika – 6. wariant niepomyślny

Założono, że przekazany do licznika obraz nie jest przez niego obsługiwany.

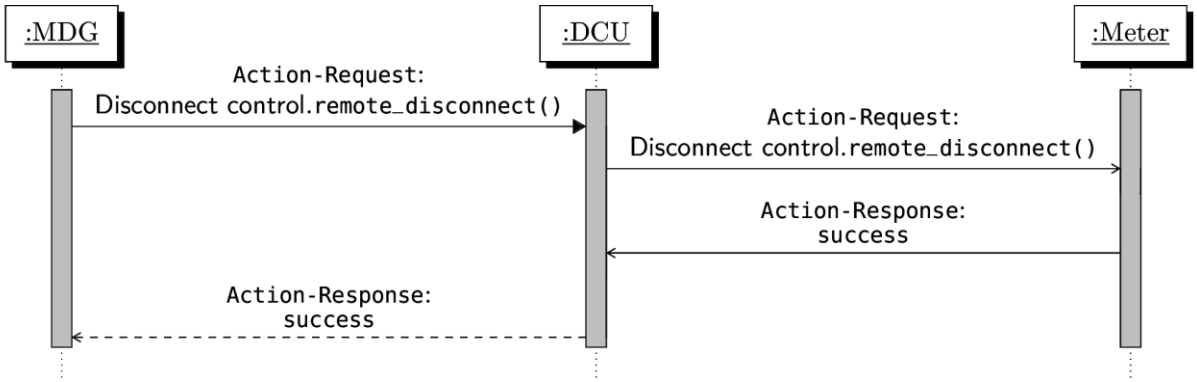


Rys. 30 Diagram sekwencji aktualizacji oprogramowania licznika – 7. wariant niepomyślny

Założono, że licznik przerwał aktualizację obrazu.

6.12 Wyłączenie stycznika

Wyłączenie stycznika w liczniku (patrz Rys. 31) polega na wysłaniu polecenia *Action-Request* wywołującego metodę *remote_disconnect()* obiektu *Disconnect control* wskazanego licznika. Wykonana operacja potwierdzana jest systemowi akwizycji.

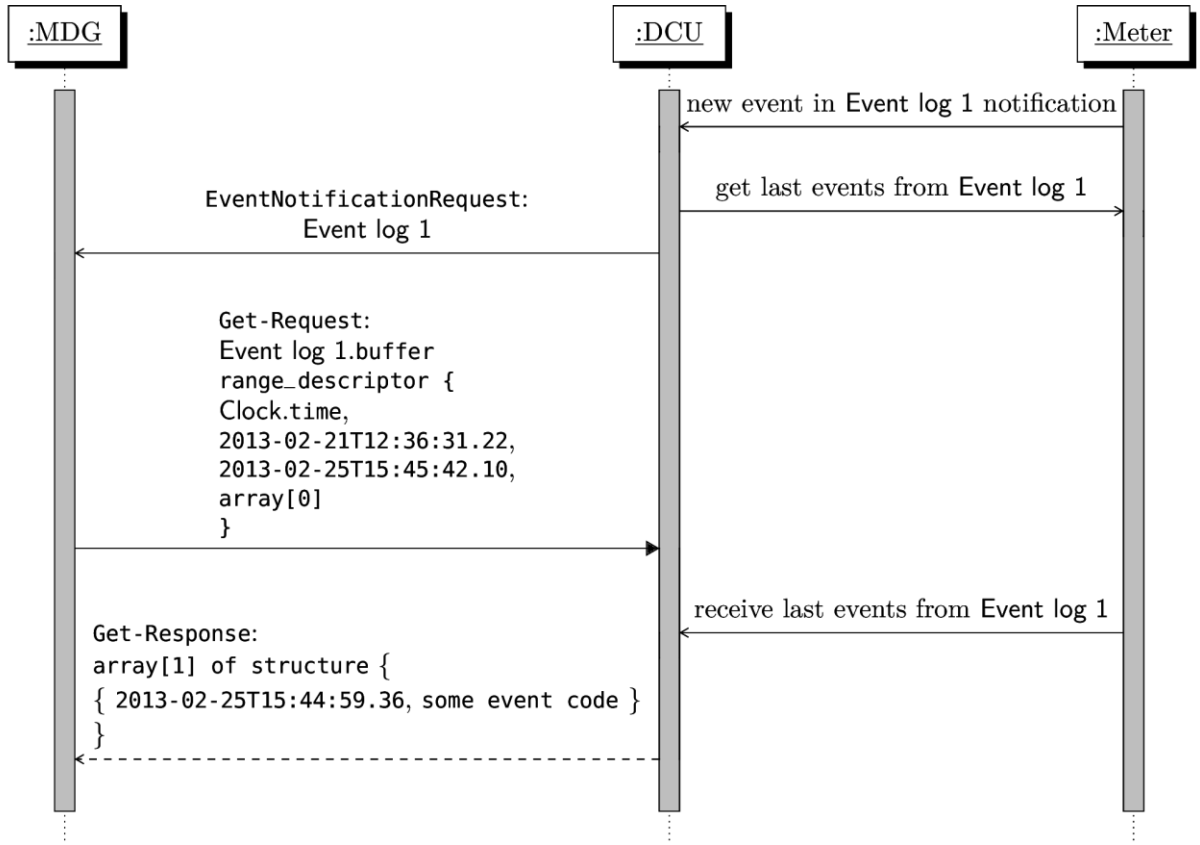


Rys. 31 Diagram sekwencji rozłączenia stycznika w liczniku

6.13 Odbieranie zdarzeń układu

Zdarzenia takie jak otwarcie pokrywy licznika i inne odkładane są do dziennika zdarzeń (dzienników zdarzeń może być wiele). System akwizycji odpytuje bufor takiego obiektu profilowego stosując wybór selektywny. Wykorzystywany jest tu deskryptor zakresowy (struktura *range_descriptor*; przedstawiona w [1] oraz [7]) sterowany czasem (w polu *restricting_object* należy podać deskryptor COSEM 2. atrybutu obiektu zegara reprezentującego czas – zdarzenia zawsze są zapisywane z czasem ich wystąpienia). Niestety zakres wyrażany tym deskryptorem (pola *from_value* i *to_value*) jest obustronnie domknięty, dlatego system akwizycji jako dolną granicę podaje czas z ostatnio odebranego rekordu z danego dziennika zdarzeń powiększony o epsilon, co w przypadku typu *date-time* oznacza inkrementację o jedną setną sekundy. Górny zakres to czas bieżący. Dzięki temu w odpowiedzi z koncentratora przesyłane są tylko najnowsze wpisy dziennika i unikamy pobierania rekordów już posiadanych.

System akwizycji może cyklicznie odpytywać koncentrator o nowe rekordy dziennika zdarzeń lub może czekać na wyzwolenie komunikatem notyfikacji (patrz Rys. 32). W przypadku korzystania z mechanizmu powiadomień, należy włączyć ten mechanizm przed żądaniem zwrócenia rekordów nowszych niż ostatni dotychczas odebrany, ponieważ w ten sposób nie utracimy żadnego powiadomienia. W najgorszym wypadku dostaniemy powiadomienie w trakcie oczekiwania na rekordy lub ich wyczytywania. Może ono dotyczyć zdarzenia zawierającego się w zleconym zakresie lub nie. Dlatego po otrzymaniu takiej notyfikacji, system akwizycji musi ponownie zlecić koncentratorowi zwrócenie ostatnich rekordów (tym razem już z nowym zakresem), a w odpowiedzi może dostać pustą tablicę.



Rys. 32 Diagram sekwencji odbierania nowych zdarzeń z licznika

Założono, że w koncentratorze są aktywne akceleracja dostępu do danych z liczników i powiadomienia oraz że atrybut *capture_objects* obiektu *Event log 1* zawiera czas i *Event code object #1*.

7 Rekomendacje dla implementacji serwera protokołu DCSAP

7.1 Port TCP

Rekomendowanym numerem portu dla protokołu DCSAP jest port nr 4069, dla protokołu DCSAP/TLS – port nr 4169.

7.2 Ilość obsługiwanych, równoległych sesji DCSAP

W protokole nie zdefiniowano ograniczenia dla ilości sesji komunikacyjnych. Rekomendowana liczba sesji otwieranych przez system akwizycji przy wielosesyjnej komunikacji z koncentratorem to 3 – stale utrzymywana sesja dla strumienia instrukcji, stale utrzymywana sesja do odbierania zdarzeń oraz powoływana w razie potrzeby sesja diagnostyczna.

W większości przypadków komunikacja z DCU będzie obsługiwana za pomocą pojedynczej sesji DCSAP.

7.3 Rozmiar listy liczników

Dopuszczalna liczba wpisów w tablicy przechowującej listę liczników (reprezentowana przez atrybut *max_entries* obiektu *Data concentrator meter list*), z którymi koncentrator ma lub miał łączność, powinna być kilkukrotnie większa niż liczba liczników na stacjach. Zalecana minimalna wartość to 2048.

7.4 Rozmiar dziennika zdarzeń

Rozmiar dziennika zdarzeń realizowanego za pomocą obiektu *Data concentrator event list* powinien być na tyle duży aby w praktyce, po ewentualnej utracie komunikacji z koncentratorem pozwolić na odzyskanie historii zdarzeń. Zalecana minimalna wartość to 16384.

7.5 Bezpieczeństwo sesji DCSAP

Rekomendowane jest wykorzystanie DCSAP wewnątrz TLS 1.2 z uwierzytelnieniem obu stron połączenia za pomocą certyfikatów. Zaleca się, żeby obsługa certyfikatów w serwerze DCSAP umożliwiała rozszerzenie lub ograniczenie dostępu do grup obiektów koncentratora i liczników na podstawie dodatkowych podpisanych pól certyfikatu.

7.6 Synchronizacja czasu

Protokół DCSAP zakłada synchronizację czasu pomiędzy DCU i systemem akwizycji. Mechanizm synchronizacji czasu jest jednak poza specyfikacją protokołu. Zalecaną metodą synchronizacji czasu jest protokół NTP [9] i używanie tych samych serwerów czasu, co Aplikacja AMI. Lista serwerów NTP realizowana przez obiekt koncentratora *Data concentrator NTP server list* powinna pozwolić na zdefiniowanie co najmniej 4 serwerów.

7.7 Przepływność pojedynczej sesji DCSAP

Dla potrzeb implementacji należy założyć, że średnia przepływność łącza do pojedynczego koncentratora wynosi 64 kbit/s i z taką prędkością będą przekazywane dane liczone zbiorczo w ramach wszystkich sesji. Powiadomienia są lekkimi komunikatami a sesje diagnostyczne będą powoływane sporadycznie, dlatego przy wielosesyjnej komunikacji z koncentratorem można przyjąć, że równoczesne wykorzystanie łącza przez wiele sesji, choć musi być wspierane, będzie zachodzić na tyle rzadko, że poszczególne sesje będą mogły w całości wykorzystać dostępną przepływność.

7.8 Sterowanie przepływem w połączeniu TCP

Koncentrator odbiera polecenia wysyłane przez system akwizycji. Może się zdarzyć, że w krótkim okresie czasu nie będzie w stanie przetworzyć wysyłanych poleceń ze względu na ograniczone zasoby (CPU, pamięć). W takim przypadku koncentrator może posiłkować się zaprzestaniem odbierania kolejnych bajtów z gniazda, co spowoduje wstrzymanie połączenia TCP w kierunku do koncentratora. Rozwiązanie takie jest niekorzystne ze względu na brak możliwości przesyłania w takiej sytuacji poleceń o wysokim priorytecie. Zakłada się jednak, że aby akwizycja była skuteczna koncentrator musi być w stanie nadążać z realizacją poleceń, a krótkotrwałe wstrzymania przepływu, spowodowane chwilowym przeciążeniem koncentratora, są akceptowalne. Jeżeli praktyka pokaże konieczność zastosowania dodatkowego mechanizmu, to zostanie on wprowadzony w kolejnych wersjach protokołu jako opcjonalny.

8 Opis implementacji referencyjnej

Implementacja referencyjna protokołu DCSAP jest przykładem jego realizacji w języku C. Ma ona na celu zademonstrować działanie protokołu, pokazać szczegóły interakcji pomiędzy stronami oraz zaproponować prostą i przejrzystą dekompozycję. Wprowadzono podział na podsystemy odpowiedzialne za określoną część funkcjonalności całego protokołu.

Poza realizacją podstawowych funkcji protokołu dołączono też prosty symulator wirtualnego koncentratora z możliwością dodawania i usuwania wirtualnych liczników energii elektrycznej.

Poniżej przedstawiono opis każdego podsystemu oraz ich funkcje.

8.1 Kodowanie A-XDR

Jest to zbiór funkcji pozwalających na kodowanie i dekodowanie komunikatów zapisanych zgodnie z regułami standardu A-XDR. Wszystkie funkcje przekształcają wartości ze zmiennej, która jest argumentem wywołania, do postaci uszeregowanej w zaalokowanym buforze lub na odwrót. Argumentami wywołania zawsze są podwójny wskaźnik na bufor oraz wskaźnik na jego rozmiar. W wyniku poprawnego wykonania operacji zarówno wskaźnik na bufor jak i jego rozmiar są aktualizowane w taki sposób aby wskazywać na kolejny element. Wartości zwracane informują o błędzie lub, w przypadku wartości dodatniej, o wymaganym minimalnym rozmiarze bufora niezbędnym do poprawnego wykonania operacji.

```
int32 axdr_read_choice(uint8 **buff, uint32 *bufflen, uint8 *val);
int32 axdr_read_enum(uint8 **buff, uint32 *bufflen, uint8 *val);
```

Odczytanie z bufora elementu *CHOICE* lub *ENUMERATED*. Konsumowany jest jeden bajt z bufora. Przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor z zakodowaną postacią danej.
- *bufflen* – wskaźnik na rozmiar bufora.
- *val* – wskaźnik na zmienną wypełnianą zdekodowaną wartością.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- Wartość 1 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.

```
int32 axdr_read_optional(uint8 **buff, uint32 *bufflen, int *val);
int32 axdr_read_bool(uint8 **buff, uint32 *bufflen, int *val);
```

Odczytanie z bufora elementu *OPTIONAL* lub *BOOLEAN*. Konsumowany jest jeden bajt z bufora. Przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor z zakodowaną postacią danej.
- *bufflen* – wskaźnik na rozmiar bufora.
- *val* – wskaźnik na zmienną wypełnianą zdekodowaną wartością.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- Wartość 1 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.

```
int32 axdr_read_int32(uint8 **buff, uint32 *bufflen, uint32 len, int32 *val);
int32 axdr_read_uint32(uint8 **buff, uint32 *bufflen, uint32 len, uint32 *val);
int32 axdr_read_int64(uint8 **buff, uint32 *bufflen, uint32 len, int64 *val);
int32 axdr_read_uint64(uint8 **buff, uint32 *bufflen, uint32 len, uint64 *val);
```

Odczytanie z bufora elementu wartości *INTEGER*, 32 lub 64 bitowej ze znakiem lub bez. Konsumowana jest wskazana liczba bajtów z bufora. Przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor z zakodowaną postacią danej.
- *bufflen* – wskaźnik na rozmiar bufora.
- *len* – rozmiar zakodowanej postaci (wartość z przedziału 1–4 dla wariantów 32-bitowych lub 1–8 dla wariantów 64-bitowych).
- *val* – wskaźnik na zmienną wypełnianą zdekodowaną wartością.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- > 0 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.
- < 0 błędna wartość argumentu *len*.

```
int32 axdr_read_int32var(uint8 **buff, uint32 *bufflen, int32 *val);
int32 axdr_read_uint32var(uint8 **buff, uint32 *bufflen, uint32 *val);
int32 axdr_read_int64var(uint8 **buff, uint32 *bufflen, int64 *val);
int32 axdr_read_uint64var(uint8 **buff, uint32 *bufflen, uint64 *val);
```

Odczytanie z bufora elementu wartości *INTEGER*, 32 lub 64 bitowej ze znakiem lub bez, zakodowanej w postaci o zmiennej długości. Jeżeli bufor nie zawiera całego elementu operacja się nie wykona, a zwrócona wartość wskaże minimalną potrzebną wielkość wypełnionego bufora. W przypadku powodzenia przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor z zakodowaną postacią danej.
- *bufflen* – wskaźnik na rozmiar bufora.
- *val* – wskaźnik na zmienną wypełnianą zdekodowaną wartością.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- > 0 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.
- < 0 wartość poza zakresem 32 lub 64 bitów.

```
int32 axdr_read_len(uint8 **buff, uint32 *bufflen, uint32 *val);
```

Odczytanie z bufora długości następnego elementu. Długość sama kodowana jest w postaci o zmiennej długości. Jeżeli bufor nie zawiera całego elementu operacja się nie wykona, a zwrócona wartość wskaże minimalną potrzebną wielkość wypełnionego bufora. W przypadku powodzenia przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor z zakodowaną postacią danej.
- *bufflen* – wskaźnik na rozmiar bufora.
- *val* – wskaźnik na zmienną wypełnianą zdekodowaną wartością.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- > 0 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.
- < 0 wartość poza zakresem 32 bitów.

```
int32 axdr_read_string(uint8 **buff, uint32 *bufflen, uint32 len, void *val);
int32 axdr_read_bitstring(uint8 **buff, uint32 *bufflen, uint32 len, void *val);
```

Odczytanie z bufora ciągu bajtów lub bitów o określonej długości. Konsumowana jest wskazana liczba bajtów z bufora. W przypadku odczytywania bitów skonsumowana zostanie minimalna liczba bajtów zawierająca wskazaną liczbę bitów. Przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor z zakodowaną postacią danej.
- *bufflen* – wskaźnik na rozmiar bufora.
- *len* – rozmiar ciągu bajtów lub bitów. Musi być większy od 0.
- *val* – wskaźnik na miejsce docelowe odczytywanego ciągu.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- > 0 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.

• < 0 niepoprawny rozmiar ciągu.

```
int32 axdr_write_choice(uint8 **buff, uint32 *bufflen, uint8 val);
int32 axdr_write_enum(uint8 **buff, uint32 *bufflen, uint8 val);
```

Zapisanie do bufora elementu *CHOICE* lub *ENUMERATED*. Konsumowany jest jeden bajt bufora. Przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

• *buff* – podwójny wskaźnik na bufor docelowy.

• *bufflen* – wskaźnik na rozmiar bufora.

• *val* – kodowana wartość.

Wynik:

• Wartość 0 oznaczająca poprawne wykonanie.

• Wartość 1 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.

```
int32 axdr_write_optional(uint8 **buff, uint32 *bufflen, int val);
int32 axdr_write_bool(uint8 **buff, uint32 *bufflen, int val);
```

Zapisanie do bufora elementu *OPTIONAL* lub *BOOLEAN*. Konsumowany jest jeden bajt bufora. Przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

• *buff* – podwójny wskaźnik na bufor docelowy.

• *bufflen* – wskaźnik na rozmiar bufora.

• *val* – kodowana wartość.

Wynik:

• Wartość 0 oznaczająca poprawne wykonanie.

• Wartość 1 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.

```
int32 axdr_write_int32(uint8 **buff, uint32 *bufflen, uint32 len, int32 val);
int32 axdr_write_uint32(uint8 **buff, uint32 *bufflen, uint32 len, uint32 val);
int32 axdr_write_int64(uint8 **buff, uint32 *bufflen, uint32 len, int64 val);
int32 axdr_write_uint64(uint8 **buff, uint32 *bufflen, uint32 len, uint64 val);
```

Zapisanie do bufora elementu wartości *INTEGER*, 32 lub 64 bitowej ze znakiem lub bez. Konsumowana jest wskazana liczba bajtów bufora. Przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor docelowy.
- *bufflen* – wskaźnik na rozmiar bufora.
- *len* – rozmiar zakodowanej postaci (wartość z przedziału 1–4 dla wariantów 32-bitowych lub 1–8 dla wariantów 64-bitowych).
- *val* – kodowana wartość.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- > 0 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.
- < 0 błędna wartość argumentu *len*.

```
int32 axdr_write_int32var(uint8 **buff, uint32 *bufflen, int32 val);
int32 axdr_write_uint32var(uint8 **buff, uint32 *bufflen, uint32 val);
int32 axdr_write_int64var(uint8 **buff, uint32 *bufflen, int64 val);
int32 axdr_write_uint64var(uint8 **buff, uint32 *bufflen, uint64 val);
```

Zapisanie do bufora elementu wartości *INTEGER*, 32 lub 64 bitowej ze znakiem lub bez, zakodowanej w postaci o zmiennej długości. Jeżeli bufor nie zmieści całego elementu operacja się nie wykona, a zwrócona wartość wskaże minimalną potrzebną wielkość bufora. W przypadku powodzenia przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor docelowy.
- *bufflen* – wskaźnik na rozmiar bufora.
- *val* – kodowana wartość.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- > 0 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.

```
int32 axdr_write_len(uint8 **buff, uint32 *bufflen, uint32 val);
```

Zapisanie do bufora długości następnego elementu. Długość sama kodowana jest w postaci o zmiennej długości. Jeżeli bufor nie zmieści całego elementu operacja się nie wykona, a zwrócona wartość wskaże minimalną potrzebną wielkość bufora. W przypadku powodzenia przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor docelowy.
- *bufflen* – wskaźnik na rozmiar bufora.
- *val* – kodowana wartość.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- > 0 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.

```
int32 axdr_write_string(uint8 **buff, uint32 *bufflen, uint32 len, void *val);
int32 axdr_write_bitstring(uint8 **buff, uint32 *bufflen, uint32 len, void *val);
```

Zapisanie do bufora ciągu bajtów lub bitów o określonej długości. Zapisywana jest wskazana liczba bajtów do bufora. W przypadku zapisywania bitów skonsumowana zostanie minimalna liczba bajtów zawierająca wskazaną liczbę bitów. Przesuwany jest wskaźnik bufora oraz zmniejszany jego rozmiar.

Argumenty:

- *buff* – podwójny wskaźnik na bufor docelowy.
- *bufflen* – wskaźnik na rozmiar bufora.
- *len* – rozmiar ciągu bajtów lub bitów. Musi być większy od 0.
- *val* – wskaźnik na ciąg bajtów lub bitów.

Wynik:

- Wartość 0 oznaczająca poprawne wykonanie.
- > 0 oznaczająca zbyt mały bufor i określa jego minimalny rozmiar.
- < 0 niepoprawny rozmiar ciągu.

8.2 Obsługa gniazd TCP

Zestaw funkcji opakowujących gniazda TCP w trybie nieblokującym. Pozwala na łatwe tworzenie gniazd klienta i serwera, nawiązywanie połączeń lub ich akceptowanie oraz wysyłanie i odbieranie danych.

```
int tcpsocket_open(char *host, uint16 port, int32 timeout);
```

Tworzy gniazdo klienta oraz nawiązuje połączenie z serwerem. Jeżeli operacja nie powiedzie się w zadanym czasie zwracany jest błąd.

Argumenty:

- *host* – nazwa domenowa serwera lub jego adres IP.
- *port* – port serwera.
- *timeout* – czas na wykonanie operacji w milisekundach.

Wynik:

- >= 0 deskryptor gniazda.
- < 0 niepowodzenie.

```
int tcpsocket_close(int sockfd);
```

Zamyka połączenie oraz gniazdo klienta.

Argumenty:

- *sockfd* – deskryptor gniazda.

Wynik:

- 0 sukces

```
int tcpsocket_server(uint16 port);
```

Tworzy gniazdo serwera nasłuchujące na wskazanym porcie.

Argumenty:

• *port* – port serwera.

Wynik:

• >= 0 deskryptor gniazda.

• < 0 niepowodzenie.

```
int tcpsocket_accept(int sockfd);
```

Akceptuje połączenie od klienta. Zwracany jest deskryptor gniazda obsługującego połączenie z klientem.

Argumenty:

• *sockfd* – deskryptor gniazda serwera.

Wynik:

• >= 0 deskryptor gniazda obsługującego połączenie z klientem.

• < 0 niepowodzenie.

```
int tcpsocket_send(int sockfd, uint8 *buff, uint32 len, int32 timeout);
```

Wysyła dane przez otwarte gniazdo. Jeżeli operacja nie powiedzie się w zadanym czasie zwracany jest błąd.

Argumenty:

- *sockfd* – deskryptor gniazda.
- *buff* – bufor z danymi do wysłania.
- *len* – ilość danych do wysłania.
- *timeout* – czas na wykonanie operacji w milisekundach.

Wynik:

- 0 dane zostały wysłane.
- < 0 niepowodzenie.

```
int tcpsocket_receive(int sockfd, uint8 *buff, uint32 len, int32 timeout);
```

Odbiera dane z otwartego gniazda. Jeżeli operacja nie powiedzie się w zadanym czasie zwracany jest błąd.

Argumenty:

- *sockfd* – deskryptor gniazda.
- *buff* – bufor na dane.
- *len* – ilość danych do odebrania.
- *timeout* – czas na wykonanie operacji w milisekundach..

Wynik:

- 0 dane zostały odebrane.
- < 0 niepowodzenie.

8.3 Serwer DCSAP

Prosta implementacja serwera nasłuchującego i akceptującego połączenia od klientów. Serwer uruchamia oddzielny wątek czekający na nowe połączenia. Dla każdego nowego klienta tworzone jest dedykowane połączenie obsługiwane w podsystemie połączeń DCSAP.

```
int dcsapserver_start(uint16 port, int32 timeout, int32 idle, int connmax);
```

Uruchamia serwer nasłuchujący na wszystkich interfejsach na wskazanym porcie.

Argumenty:

• *port* – port serwera.

• *timeout* – czas w milisekundach na wykonanie operacji wysyłania i odbierania danych w ustanowionym połączeniu z klientem.

• *idle* – czas w milisekundach maksymalnej bezczynności klienta, po którym nastąpi jego rozłączenie.

• *connmax* – maksymalna liczba jednocześnie obsługiwanych klientów.

Wynik:

• 0 serwer uruchomiony.

• < 0 niepowodzenie.

```
static void *dcsapserver_thread(void *data);
```

Wewnętrzna funkcja wykonująca kod wątku serwera DCSAP.

8.4 Podsystem połączeń DCSAP

Podsystem ten obsługuje niezależne połączenia od klientów. Na potrzeby każdego z nich tworzona jest struktura opisująca połączenie oraz uruchamiany jest dedykowany wątek. Jest on odpowiedzialny za odbieranie komunikatów oraz wstępne ich parsowanie. Puste komunikaty są natychmiast odsyłane do klienta zgodnie z wymaganiami protokołu. W ramach połączenia realizowane są obiekty sesyjne sterujące notyfikacjami oraz ewentualnym buforowaniem odpowiedzi liczników. Polecenia przekazywane są do kolejnego podsystemu obiektów DCSAP. Przekazanie komunikatu do kolejnego podsystemu wiąże się ze zwiększeniem licznika referencji danego połączenia co powstrzymuje usunięcie struktury z jego opisem. Wynika to z faktu, że wskaźnik na strukturę połączenia nie może być unieważniony do momentu odesłania odpowiedzi na wszystkie przekazana polecenia.

Podsystem połączeń odpowiedzialny jest także za rozsyłanie do odpowiednich klientów komunikatów notyfikacji. Trafiają one tylko do tych klientów, którzy włączyli mechanizm asynchronicznych notyfikacji za pomocą odpowiedniego obiektu sesyjnego.

```
int dcsapconn_create(int sockfd, int32 timeout, int32 idle, int connmax);
```

Uruchamia obsługę nowego połączenia nawiązanego przez klienta. Jeżeli obsługiwana jest już maksymalna liczba klientów, funkcja nie powodzi się. W trakcie poprawnego wykonania tworzona jest struktura opisująca połączenia, następnie jest ona dodawana do listy obsługiwanych połączeń. Tworzone są obiekty sesyjne. Uruchamiany jest wątek obsługujący odbieranie komunikatów. Funkcja ta wywoływana jest przez wątek serwera DCSAP.

Argumenty:

• *sockfd* – deskryptor gniazda połączenia.

• *timeout* – czas w milisekundach na wykonanie operacji wysyłania i odbierania danych w ustanowionym połączeniu z klientem.

• *idle* – czas w milisekundach maksymalnej bezczynności klienta, po którym nastąpi jego rozłączenie.

• *connmax* – maksymalna liczba jednocześnie obsługiwanych klientów.

Wynik:

• 0 uruchomiono obsługę połączenia.

• < 0 niepowodzenie lub przekroczona maksymalna liczba jednocześnie obsługiwanych klientów.

```
int dcsapconn_response(dcsapconn_data_t *conn, uint32 deviceid, uint64 messageid, int32
datasize, uint8 *dlmsdata);
```

Odsyła odpowiedź na przyjęte polecenie jeżeli nie nastąpiło rozłączenie połączenia. Zmniejszany jest licznik referencji sygnalizując zwolnienie zasobu jakim jest struktura opisująca połączenie. W przypadku zerwania połączenia, struktura ta musi pozostać w pamięci dopóki kolejne podsystemy nie zwrócą odpowiedzi na wszystkie przekazane im polecenia otrzymane w ramach tego połączenia. Synchronizowanie wysyłania wielu odpowiedzi jednocześnie realizowane jest dedykowaną sekcją krytyczną niezależną dla każdego połączenia.

Argumenty:

- *conn* – wskaźnik struktury połączenia.
- *deviceid* – identyfikator urządzenia.
- *messageid* – identyfikator komunikatu.
- *datasize* – rozmiar danych DLMS. Ujemna wartość oznacza błąd i brak danych.
- *dlmsdata* – dane DLMS, tylko w przypadku dodatniej wartości argumentu *datasize*.

Wynik:

- 0 odpowiedź została wysłana.
- < 0 niepowodzenie. Połączenie zostanie zamknięte, nie należy ponawiać.

```
int dcsapconn_notify(uint32 deviceid, int32 datasize, uint8 *dlmsdata);
```

Rozsyła notyfikację. Wyszukiwane są połączenia z włączonym mechanizmem notyfikacji. Dla każdego z nich zwiększany jest licznik referencji, po czym, już poza sekcją krytyczną zabezpieczającą listę połączeń, następuje wywołanie funkcji *dcsapconn_response* i wysłanie notyfikacji.

Argumenty:

- *deviceid* – identyfikator urządzenia.
- *datasize* – rozmiar danych DLMS. Musi być dodatni.
- *dlmsdata* – dane DLMS.

Wynik:

- 0 notyfikacja została rozesłana.
- < 0 niepowodzenie lub niepoprawny datasize.

```
static void *dcsapconn_thread(void *data);
```

Wewnętrzna funkcja wykonująca kod wątku połączenia DCSAP.

8.5 Podsystem obiektów DCSAP

Podsystem obiektów pozwala na dynamiczne rejestrowanie i usuwanie obiektów COSEM o charakterze globalnym oraz sesyjnym. Obiekty mogą być wiązane z adresem koncentratora lub dowolnego licznika. Podsystem przechowuje listę wszystkich zarejestrowanych obiektów, a dla każdego z nich klasę, identyfikator instancji, wskaźnik na strukturę danych oraz wskaźniki na funkcje przetwarzające proste polecenia dla obiektu (*get*, *set* oraz *action*). W tym podsystemie realizowane jest parsowanie komunikatów DLMS i przekazanie do realizacji w kontekście wskazanej instancji obiektu. Polecenia, które nie zostaną obsłużone przez podsystem obiektów DCSAP trafiają do podsystemu liczników.

```
int dcsapobjects_register(dcsapconn_data_t *conn, uint32 deviceid, uint16 classid, uint64
instanceid, void *data, get_function get, set_function set, action_function action);
```

Rejestruje obiekt COSEM o charakterze globalnym lub sesyjnym w przypadku podania wskaźnika struktury połączenia. Obiekty mogą być wiązane z adresem koncentratora lub dowolnego licznika. Przekazywana jest klasa, identyfikator instancji, wskaźnik na strukturę danych oraz wskaźniki na funkcje przetwarzające proste polecenia dla obiektu (*get*, *set* oraz *action*).

Argumenty:

- *conn* – wskaźnik struktury połączenia lub NULL dla obiektów globalnych.
- *deviceid* – identyfikator urządzenia.
- *classid* – klasa obiektu.
- *instanceid* – instancja obiektu.
- *data* – wskaźnik na dane obiektu.
- *get* – wskaźnik na funkcję przetwarzającą polecenia *get*.
- *set* – wskaźnik na funkcję przetwarzającą polecenia *set*.
- *action* – wskaźnik na funkcję przetwarzającą polecenia *action*.

Wynik:

- 0 zarejestrowano obiekt.
- < 0 niepowodzenie lub obiekt istnieje.

```
int dcsapobjects_delete(dcsapconn_data_t *conn, uint32 deviceid, uint16 classid, uint64
instanceid);
```

Usuwa obiekt COSEM z listy podsystemu. Sekcja krytyczna chroni przed usuwaniem obiektu, który bierze udział w realizacji polecenia.

Argumenty:

- *conn* – wskaźnik struktury połączenia lub NULL dla obiektów globalnych.
- *deviceid* – identyfikator urządzenia.
- *classid* – klasa obiektu.
- *instanceid* – instancja obiektu.

Wynik:

- 0 usunięto obiekt z listy.
- < 0 obiekt nie istnieje.

```
int dcsapobjects_request(dcsapconn_data_t *conn, uint32 deviceid, uint64 messageid, int32
datasize, uint8 *dlmsdata);
```

Wykonuje polecenie skierowane do zarejestrowanych obiektów. Z listy zarejestrowanych obiektów wybierany jest adres danych obiektu oraz obsługująca go funkcja. Jeżeli zaadresowany jest obiekt licznika nie odwzorowany w tym podsystemie nastąpi przekazanie polecenia do podsystemu liczników.

Argumenty:

- *conn* – wskaźnik struktury połączenia.
- *deviceid* – identyfikator urządzenia.
- *messageid* – identyfikator komunikatu.
- *datasize* – rozmiar danych DLMS. Musi być dodatni.
- *dlmsdata* – dane DLMS.

Wynik:

- 0 polecenie zostało zrealizowane lub przekazane do podsystemu liczników.

• < 0 niepowodzenie lub niepoprawny *datasize*.

8.6 Podsystem liczników

Tutaj trafiają polecenia, które koncentrator powinien przekazać do realizacji licznikom rzeczywistym. Są one wkładane do kolejki z której mogą być wybierane w dowolnej kolejności. Oddzielna kolejka obsługuje polecenia priorytetowe. W tym podsystemie może być realizowane buforowanie niektórych odpowiedzi liczników.

```
int dcsapmeters_request(dcsapconn_data_t *conn, uint32 deviceid, uint64 messageid, int32
datasize, uint8 *dlmsdata);
```

Przyjmuje do wykonania polecenie skierowane do liczników rzeczywistych. Komunikat jest kopiowany do lokalnej kolejki. Oddzielnie obsługiwana jest kolejka poleceń priorytetowych. Sterowanie zwracane jest bez oczekiwania na wykonanie polecenia. Podsystem odpowiedzialny jest za przekazanie odpowiedzi do podsystemu połączeń DCSAP.

Argumenty:

• *conn* – wskaźnik struktury połączenia.

• *deviceid* – identyfikator urządzenia.

• *messageid* – identyfikator komunikatu.

• *datasize* – rozmiar danych DLMS. Musi być dodatni.

• *dlmsdata* – dane DLMS.

Wynik:

• 0 polecenie zostało przyjęte do realizacji.

• < 0 niepowodzenie lub niepoprawny *datasize.*

```
int dcsapmeters_cancel(dcsapconn_data_t *conn);
```

Funkcja anuluje nie wykonane jeszcze polecenia przyjęte w ramach wskazanego połączenia. Wywoływana jest z podsystemu połączeń DCSAP. Zwracana jest ilość anulowanych operacji, które nie zwrócą już żadnego wyniku.

Argumenty:

• *conn* – wskaźnik struktury połączenia.

Wynik:

• Ilość anulowanych poleceń.

```
int dcsapmeters_start();
```

Funkcja uruchamia wątek podsystemu liczników.

Wynik:

• 0 sukces.

```
static void *dcsapmeters_thread(void *data)
```

Wewnętrzna funkcja wykonująca kod wątku podsystemu liczników.

9 Literatura:

1. DLMS UA 1000-1:2010, „Blue Book”, COSEM Identification System and Interface Classes, 10th ed., 2010-08-26.
2. DLMS UA 1000-2:2009, „Green Book”, DLMS/COSEM Architecture and Protocols, 7th ed., 2009-12-22.
3. IEC 61334-6:2000, Distribution automation using distribution line carrier systems – Part 6: A-XDR encoding rule, 1st ed., 2000-06.
4. IEC 62056-21:2002, Electricity metering – Data exchange for meter reading, tariff and load control – Part 21: Direct local data exchange, 1st ed., 2002-05.
5. IEC 62056-53:2006, Electricity metering – Data exchange for meter reading, tariff and load control – Part 53: COSEM application layer, 2nd ed., 2006-12.
6. IEC 62056-61:2006, Electricity metering – Data exchange for meter reading, tariff and load control – Part 61: Object identification system (OBIS), 2nd ed., 2006-11.
7. IEC 62056-62:2006, Electricity metering – Data exchange for meter reading, tariff and load control – Part 62: Interface classes, 2nd ed., 2006-11.
8. Draft Standard for PowerLine Intelligent Metering Evolution, version R1.3.6, PRIME Alliance TWG.
9. D. Mills et al., RFC5905, Network Time Protocol Version 4: Protocol and Algorithms Specification, 2010-06.
10. S. Feuerhahn et al., „Comparison of the communication protocols DLMS/COSEM, SML and IEC 61850 for smart metering applications”, in: IEEE International Conference on Smart Grid Communications, 2011, pp. 410–415.