PRZYKŁAD WYKORZYSTANIA PROTOKOŁU DATA CONCENTRATOR SIMPLE ACQUISITION PROTOCOL (DCSAP)

Spis treści

Spis treści 2
1 Słownik pojęć 7
2 Wstęp 9
3 Założenia protokołu 11
3.1 Utrzymywanie sesji sieciowej z DCU 11
3.2 Idempotentne operacje i separacja sesji 11
3.3 Asynchroniczne przekazywanie odpowiedzi 12
3.4 Wykorzystanie protokołu DLMS/COSEM 12
3.5 Rozszerzona adresacja DLMS 12
3.6 Opcjonalne zabezpieczenie za pomocą SSL 12
4 Komunikacja 13
4.1 Polecenia 15
4.1.1 Zapytanie o A+ licznika o identyfikatorze 1 16
4.1.2 Żądanie ustawienia rozmiaru bufora na pomiary zatrzaskiwane w profilu z drugim okresem zatrzaskiwania do licznika o identyfikatorze 11 17
4.1.3 Żądanie rozłączenia stycznika licznika o identyfikatorze 15 18
4.2 Odpowiedzi 19
4.2.1 Odpowiedź na zapytanie o A+ licznika o identyfikatorze 1 20
4.2.2 Odpowiedź na żądanie ustawienia rozmiaru bufora na pomiary zatrzaskiwane w profilu z drugim okresem zatrzaskiwania od licznika o identyfikatorze 11 20
4.2.3 Odpowiedź na żądanie rozłączenia stycznika licznika o identyfikatorze 15 21
4.3 Notyfikacje 22
4.3.1 Powiadomienie o zdarzeniu zarejestrowanym przez licznik o identyfikatorze 127 w pierwszym dzienniku zdarzeń 22
5 Obiekty COSEM 23
5.1 Reguły numerowania nowych klas i obiektów 24
5.1.1 Dobór nowych identyfikatorów klas interfejsów (*class_id*) 24
5.1.2 Dobór nowych kodów OBIS 24
5.2 Klasy Interfejsów 25
5.2.1 Klasa *String list* 26
5.2.2 Klasa *Meter list* 26

5.2.3 Klasa *Device firmware* ... 27
5.2.4 Klasa *Device basic information* ... 30
5.2.5 Klasa *Device run information* ... 31
5.2.6 Klasa *Event list* ... 32
5.2.7 Klasa DCSAP *network statistics* ... 34
5.3 Globalne obiekty koncentratora ... 35
5.3.1 Obiekt *Data concentrator meter list* ... 35
5.3.2 Obiekt *Data concentrator firmware* ... 36
5.3.3 Obiekt *Data concentrator run information* ... 36
5.3.4 Obiekt *Data concentrator event list* ... 37
5.3.5 Obiekt *Data concentrator NTP server list* ... 37
5.3.6 Obiekt *Data concentrator network statistics* ... 37
5.4 Sesyjne obiekty koncentratora ... 38
5.4.1 Obiekt *Data concentrator meter data caching enable* ... 38
5.4.2 Obiekt *Data concentrator asynchronous notification enable* ... 39
5.5 Globalne obiekty liczników realizowane przez koncentrator ... 39
5.5.1 Obiekt *Meter information* ... 39
5.5.2 Obiekt *Meter firmware* ... 40
6 Wykorzystanie protokołu DCSAP ... 41
6.1 Rozpoczęcie sesji DCSAP z koncentratorem ... 41
6.2 Zamknięcie sesji DCSAP ... 45
6.3 Synchronizacja topologii ... 45
6.4 Odczyt rejestru układu ... 45
6.5 Zapis rejestru układu ... 47
6.6 Konfiguracja układu ... 47
6.7 Pobieranie parametrów konfiguracyjnych układu ... 47
6.8 Bezpośrednia komunikacja z układem ... 48
6.9 Restart koncentratora ... 48
6.10 Aktualizacja oprogramowania koncentratora ... 50
6.11 Aktualizacja oprogramowania licznika ... 57
6.12 Wyłączenie stycznika ... 65
6.13 Odbieranie zdarzeń układu ... 65
7 Rekomendacje dla implementacji serwera protokołu DCSAP ... 67
7.1 Port TCP ... 67

7.2 Ilość obsługiwanych, równoległych sesji DCSAP 67
7.3 Rozmiar listy liczników 67
7.4 Rozmiar dziennika zdarzeń 67
7.5 Bezpieczeństwo sesji DCSAP 67
7.6 Synchronizacja czasu 68
7.7 Przepływność pojedynczej sesji DCSAP 68
7.8 Model serwera DCSAP 68
7.9 Sterowanie przepływem w połączeniu TCP 68
8 Opis implementacji referencyjnej 69
8.1 Kodowanie A-XDR 69
8.2 Obsługa gniazd TCP 78
8.3 Serwer DCSAP 81
8.4 Podsystem połączeń DCSAP 82
8.5 Podsystem obiektów DCSAP 85
8.6 Podsystem liczników 87
9 Literatura: 89

Spis tabel

Tab. 1 Kody błędów dostępne w komunikatach protokołu DCSAP 14
Tab. 2 Reguły doboru kodów OBIS dla nowych obiektów 25
Tab. 3 Klasy interfejsów wprowadzone na potrzeby protokołu DCSAP 25
Tab. 4 Klasa *String list* (*class_id* = 40100) 26
Tab. 5 Opis atrybutów klasy *String list* (*class_id* = 40100) 26
Tab. 6 Klasa Meter list (*class_id* = 3300) 26
Tab. 7 Opis atrybutów klasy *Meter list* (*class_id* = 40000) 26
Tab. 8 Klasa *Device firmware* (*class_id* = 40101) 27
Tab. 9 Kody statusowe aktualizacji oprogramowania jakie może przyjmować atrybut *last_update_status* klasy *Device firmware* 28
Tab. 10 Opis atrybutów i metod klasy Device firmware (class_id = 40101) 29
Tab. 11 Klasa *Device basic information* (*class_id* = 40102) 30
Tab. 12 Opis atrybutów klasy *Device basic information* (*class_id* = 40102) 30
Tab. 13 Klasa *Device run information* (*class_id* = 40103) 31
Tab. 14 Opis atrybutów i metod klasy Device run information (class_id = 40103) 31
Tab. 15 Klasa Event list (class id = 40001) 32
Tab. 16 Opis atrybutów i metod klasy Event list (class_id = 40001) 32
Tab. 17 Kody powodów wystąpienia lub zapisania zdarzenia jakie może przyjmować pole reason struktury event_list_entry i opisy pól stowarzyszonych 33
Tab. 18 Klasa DCSAP network statistics (class id = 40002) 34
Tab. 19 Opis atrybutów klasy *DCSAP network statistics* (*class id* = 4002) 34
Tab. 20 Globalne obiekty koncentratora 35
Tab. 21 Sesyjne obiekty koncentratora 38
Tab. 22 Globalne obiekty liczników realizowane przez koncentrator 39

Spis rysunków

Rys. 1 Akwizycja danych oparta o protokół DCSAP 9
Rys. 2 Sesja DCSAP 10
Rys. 3 Diagram sekwencji operacji wykonywanych przez system akwizycji po rozpoczęciu pierwszej sesji DCSAP z koncentratorem 42
Rys. 4 Diagram sekwencji operacji wykonywanych przez system akwizycji po rozpoczęciu sesji DCSAP z koncentratorem 43
Rys. 5 Diagram sekwencji podtrzymywania połączenia z koncentratorem 44
Rys. 6 Diagram sekwencji próby podtrzymywania połączenia z koncentratorem i bezczynności sesji w przypadku problemów z łączem 44
Rys. 7 Diagram sekwencji odczytu z licznika – wariant pomyślny 45
Rys. 8 Diagram sekwencji odczytu z licznika – wariant nieprawidłowego atrybutu 46
Rys. 9 Diagram sekwencji odczytu z licznika – wariant nieistniejącego obiektu 46
Rys. 10 Diagram sekwencji odczytu z licznika – wariant nieznanego identyfikatora licznika 46
Rys. 11 Diagram sekwencji zapisu do licznika – wariant pomyślny 47
Rys. 12 Diagram sekwencji zapisu do licznika – wariant niepomyślny 47
Rys. 13 Diagram sekwencji zapisu do koncentratora i odczytu bezpośredniego z układu 48
Rys. 14 Diagram sekwencji restartowania koncentratora – wariant niepomyślny 49
Rys. 15 Diagram sekwencji restartowania koncentratora – wariant pomyślny 49
Rys. 16 Diagram sekwencji aktualizacji oprogramowania koncentratora – wariant pomyślny 51
Rys. 17 Diagram sekwencji aktualizacji oprogramowania koncentratora – 1. wariant niepomyślny 52
Rys. 18 Diagram sekwencji aktualizacji oprogramowania koncentratora – 2. wariant niepomyślny 52
Rys. 19 Diagram sekwencji aktualizacji oprogramowania koncentratora – 3. wariant niepomyślny 53
Rys. 20 Diagram sekwencji aktualizacji oprogramowania koncentratora – 4. wariant niepomyślny 54
Rys. 21 Diagram sekwencji aktualizacji oprogramowania koncentratora – 5. wariant niepomyślny 55
Rys. 22 Diagram sekwencji aktualizacji oprogramowania koncentratora – 6. wariant niepomyślny 56
Rys. 23 Diagram sekwencji aktualizacji oprogramowania licznika – wariant pomyślny 58
Rys. 24 Diagram sekwencji aktualizacji oprogramowania licznika – 1. wariant niepomyślny 59
Rys. 25 Diagram sekwencji aktualizacji oprogramowania licznika – 2. wariant niepomyślny 59
Rys. 26 Diagram sekwencji aktualizacji oprogramowania licznika – 3. wariant niepomyślny 60
Rys. 27 Diagram sekwencji aktualizacji oprogramowania licznika – 4. wariant niepomyślny 61
Rys. 28 Diagram sekwencji aktualizacji oprogramowania licznika – 5. wariant niepomyślny 62
Rys. 29 Diagram sekwencji aktualizacji oprogramowania licznika – 6. wariant niepomyślny 63
Rys. 30 Diagram sekwencji aktualizacji oprogramowania licznika – 7. wariant niepomyślny 64
Rys. 31 Diagram sekwencji rozłączenia stycznika w liczniku 65
Rys. 32 Diagram sekwencji odbierania nowych zdarzeń z licznika 66

1 Słownik pojęć

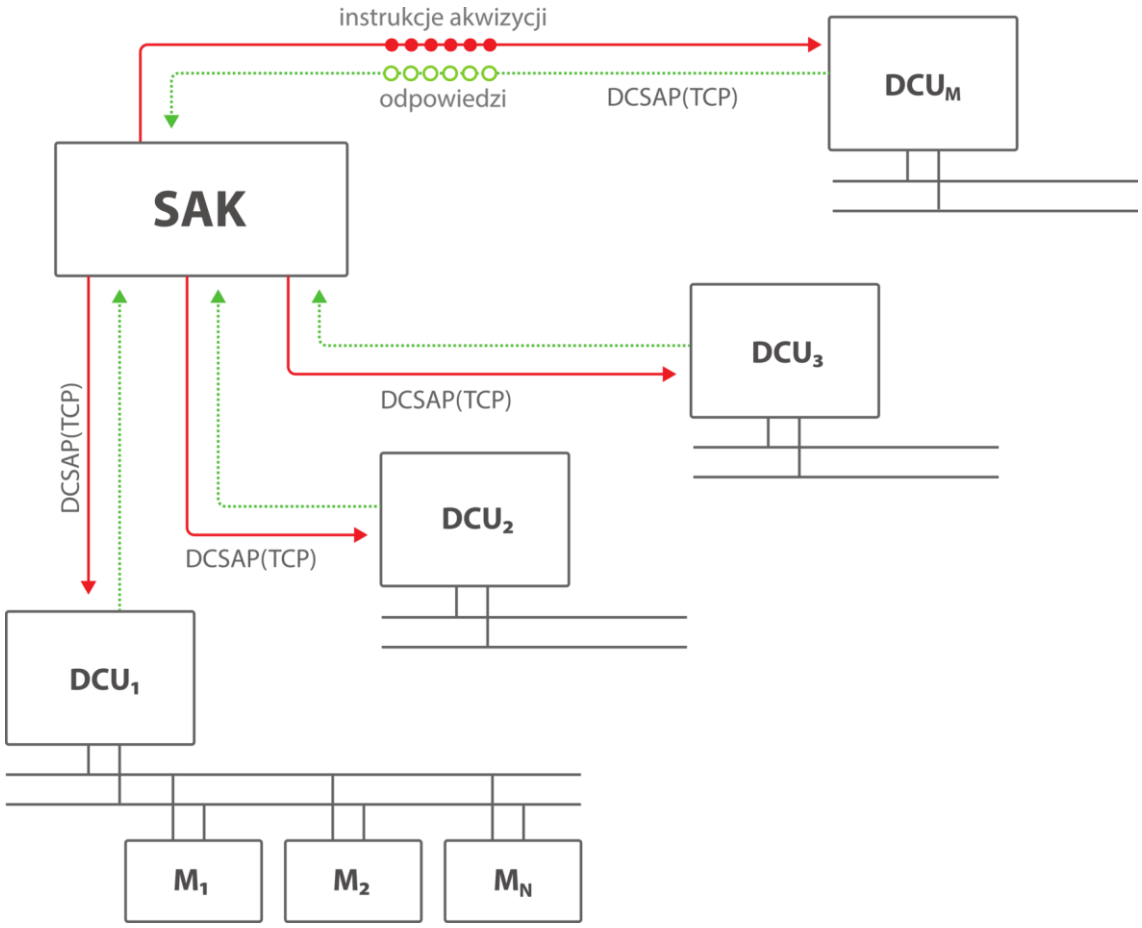
Termin/skrót	Objaśnienie
AES	*Advanced Encryption Standard* – symetryczny szyfr blokowy przyjęty przez National Institute of Standard and Technology.
AMI	*Advanced Metering Infrastructure* (Zaawansowana Infrastruktura Pomiarowa) – kompleksowy system liczników, systemów komunikacyjnych i aplikacji do 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 pewnych operacji, która pozwala na ich wielokrotne stosowanie bez zmiany wyniku.
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.
SAK	*System Akwizycji* – część Aplikacji AMI odpowiedzialna za pozyskiwanie odczytów z liczników energii elektrycznej poprzez koncentratory danych.
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.
URL	*Uniform Resource Locator* – format adresowania zasobów w sieci.

2 Wstęp

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

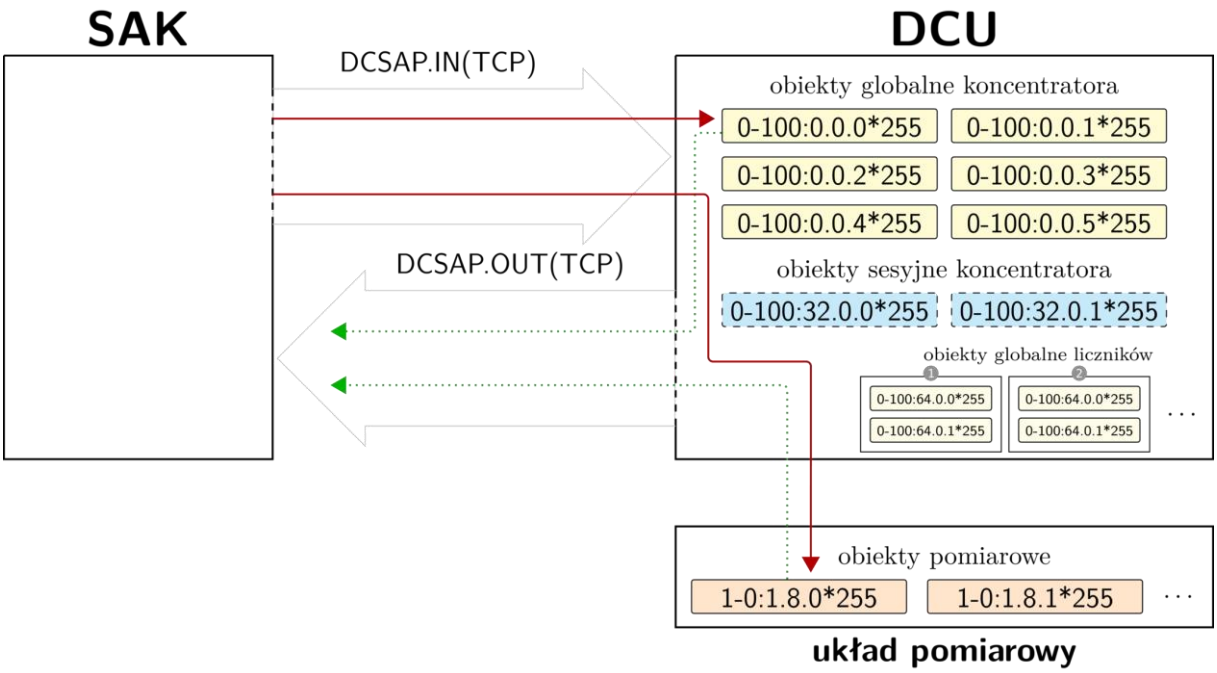
U podstaw akwizycji za pomocą protokołu DCSAP leży założenie o komunikacji z koncentratorami za pomocą strumienia instrukcji akwizycji przekazywanego poprzez dwukierunkowe, bezstratne połączenie utrzymywane z każdym koncentratorem. Instrukcja akwizycji (operacja) to pojedyncze polecenie zapisu/odczytu 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 (SAK) nawiązuje połączenia ze wszystkimi koncentratorami, a następnie zleca im wykonywanie różnych operacji. Są wśród nich operacje zmieniające stan koncentratorów i liczników. Większość operacji dotyczy jednak odczytu danych pomiarowych. Za pomocą tego samego połączenia 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 oraz na 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 pobieranie 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, dostępne wyłącznie w ramach danego połączenia. Obiekty sesyjne wprowadzono w celu sterowania 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

Protokół został opracowany zgodnie z następującymi założeniami.

3.1 Utrzymywanie sesji sieciowej z DCU

Komunikacja pomiędzy serwerem systemu akwizycji a koncentratorem odbywa się poprzez pojedyncze połączenie TCP/IP tworzące sesje. Sesja nie jest ograniczona czasowo i może być podtrzymywana tak długo jak tylko pozwalają na to warunki telekomunikacyjne. Przy tym nie jest zakładana obecność opcji *keepalive* połączenia TCP – system akwizycji, w przypadku braku żądań, okresowo wysyła pusty komunikat, który koncentrator musi odesłać w niezmienionej postaci. Pozwala to sprawdzić czy połączenie z koncentratorem nie zostało utracone. Połączenie nawiązywane jest od strony systemu akwizycji. Z tego względu wszystkie koncentratory musza być osiągalne adresowo w warstwie IP dla systemu akwizycji. System akwizycji może nawiązać wiele sesji z pojedynczym DCU.

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. Zakłada się, ze system akwizycji 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 system akwizycji może zlecać kolejne operacje zanim otrzyma potwierdzenie wykonania poprzednich. Tą samą drogą co odpowiedzi przekazywane są asynchroniczne notyfikacje pochodzące z koncentratora i liczników.

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.

3.5 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.

3.6 Opcjonalne zabezpieczenie za pomocą SSL

Szyfrowanie komunikacji może zostać zrealizowane za pomocą protokołu SSL na poziomie połączenia TCP/IP. Jest ono jednak opcjonalne i zakłada się, ze zarzadzanie kluczami i inne zagadnienia związane z szyfrowaniem zostaną rozwiązane poza protokołem DCSAP. Istnieje też możliwość sterowania opcjonalnymi mechanizmami koncentratora dotyczącymi szyfrowania poprzez protokół DCSAP, o ile tylko producent zdefiniuje odpowiednie obiekty COSEM do tego przeznaczone.

4 Komunikacja

W komunikacji pomiędzy systemem akwizycji a koncentratorami używane są komunikaty kodowane zgodnie ze standardem A-XDR [3]. Pierwotnie planowane było użycie kodowania BER, ale ponieważ to A-XDR jest używany w komunikacji DLMS jak i w warstwie aplikacji COSEM, kodując przy tym te same dane w mniejszej liczbie bajtów niż pozwalają na to inne protokoły komunikacji dedykowane do rozwiązań *Smart Metering* [10], postanowiono ostatecznie zastosować efektywniejsze kodowanie.

Podstawę stanowi komunikat DLMS, do którego dodane zostały identyfikator urządzenia, identyfikator komunikatu oraz rozmiar komunikatu DLMS stający się kodem błędu w przypadku wartości ujemnych. Na listingu 1 przedstawiono opis w formacie ASN.1. Typy *Unsigned32*, *Unsigned64* i *Integer32* zdefiniowano w dokumencie [2], [5] i są to typy *INTEGER* z odpowiednimi zakresami przyjmowanych wartości, przez co w postaci zakodowanej zajmują odpowiednio 4, 8 i 4 bajty. Tym samym nagłówek komunikatu ma zawsze stały rozmiar wynoszący 16 bajtów. Typ xDLMS-APDU zdefiniowano w tym samym dokumencie.

```
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 do przesyłania poleceń z systemu akwizycji do koncentratora, a także do przesyłania odpowiedzi i notyfikacji o zdarzeniach z koncentratora do systemu akwizycji. Komunikaty przesyłane są asynchronicznie w obu kierunkach. Znaczenie kolejnych pól komunikatu wyjaśniono w komentarzach na Listing 1.

Identyfikator urządzenia wskazuje na koncentrator lub licznik w nim zarejestrowany, który jest adresatem polecenia. W przypadku odpowiedzi pochodzącej z koncentratora, identyfikator urządzenia wskazuje na nadawcę komunikatu. W przypadku odpowiedzi lub notyfikacji określa, z którego licznika lub koncentratora pochodzi. Wartość 0 oznacza koncentrator, natomiast kolejne wartości są dynamicznie przydzielane przez koncentrator wykrytym i zarejestrowanym licznikom. Polityka przydzielania identyfikatorów zależy od producenta koncentratora.

Identyfikator komunikatu polecenia musi być użyty przez koncentrator w komunikacie będącym odpowiedzią na niego. Polityka używania identyfikatorów komunikatów zależy od systemu akwizycji, ale powinna być wykorzystana do powiązania odpowiedzi z poleceniem, zwłaszcza jeżeli system akwizycji wysyła kolejne polecenia nie czekając na odpowiedzi do poprzednich. W komunikacie notyfikacji to pole 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 na żądania źle zaadresowane, źle wypełnione, nieotrzymujące odpowiedzi z docelowego urządzenia w maksymalnym czasie przewidzianym przez koncentrator lub zaadresowane do urządzeń, w których zostały wcześniej zainicjowane czynności utrzymaniowe (jak aktualizacja oprogramowania), wymagające zaprzestania na pewien czas pełnienia przez te urządzenia ich standardowych funkcji. Zwracane wówczas kody błędów przedstawiono w Tab. 1.

Typy danych używane w komunikatach DLMS zostały szczegółowo opisane w dokumencie [2], [7].

KB	Symbol	Opis
-1	*EUNKNOWN*	Nieznany identyfikator urządzenia.
-2	*EWRONGSIZE*	Nieprawidłowy rozmiar danych (ujemny).
-3	*EPARTIAL*	Niekompletne dane.
-4	*EINVALID*	Nieprawidłowe dane.
-5	*ETIMEOUT*	Czas oczekiwania na odpowiedź z urządzenia uległ przekroczeniu.
-6	*EINACCESSIBLE*	Urządzenie jest świadomie niedostępne (np. z powodu aktualizacji).

Tab. 1 Kody błędów dostępne w komunikatach protokołu DCSAP

Kody błędów może zgłaszać jedynie koncentrator. Powinien ich używać tylko wtedy, gdy komunikat jest źle zaadresowany, niepoprawnie zbudowany, próba uzyskania odpowiedzi na wyrażone w nim żądanie do docelowego urządzenia nie powodzi się w czasie, który koncentrator uznaje za maksymalny lub też koncentrator zainicjował uprzednio zlecenie, którego realizacja wymaga tymczasowego zaprzestania świadczenia przez to urządzenie dedykowanych usług. Nie należy tych błędów mylić z błędami przewidzianymi w typach *Data-Access-Result* i *Action-Result* (używanymi w odpowiedziach), dotyczącymi komunikacji w warstwie DLMS/COSEM.

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.

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. Kolejność obsługi żądań o zwiększonym priorytecie w koncentratorze powinna być zgodna z kolejnością ich otrzymywania.

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.

Dalej przedstawiono przykłady poleceń oraz sposób ich kodowania:

4.1.1 Zapytanie o 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)
```

4.1.2 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
```

4.1.3 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 duża odpowiedź z licznika zostanie rozbita na wiele ramek przy transmisji do koncentratora, powinna zostać „przepakowana” w jedną odpowiedź przesyłaną do systemu akwizycji. Na jeden komunikat polecenia wysyłany przez system akwizycji do koncentratora powinien w odpowiedzi zawsze przyjść jeden komunikat z odpowiedzią operacji.

Przykłady odpowiedzi dla prezentowanych wcześniej przykładów poleceń oraz sposób ich kodowania przedstawiono na kolejnej stronie tego podrozdziału.

4.2.1 Odpowiedź na zapytanie o 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)
```

4.2.2 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
```

4.2.3 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. 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 odbywa się poprzez medium bez algorytmów unikania i wykrywania kolizji. W takim przypadku system akwizycji musi w pełni kontrolować transmisję w obu kierunkach. Można to zagwarantować wykonując pojedyncze polecenia i czekając na odpowiedzi do nich oraz nie uaktywniając asynchronicznego przesyłania notyfikacji.

Przykład notyfikacji oraz sposób jej kodowania podano w dalszej części tego podrozdziału.

4.3.1 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
```

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.

Specyfikacja protokołu DCSAP zawiera minimalny zestaw takich obiektów i opisuje ich semantykę. Producenci urządzeń mogą przedstawiać swoje specyficzne mechanizmy i związane z nimi obiekty realizujące dodatkową funkcjonalność. W ten sposób protokół DCSAP jest łatwo rozszerzalny.

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 (zamiast koncentratora, a więc różnym od 0).

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 i do tego synchroniczna, tzn. są obsługiwane natychmiastowo (jedyne zwłoki wynikają z opóźnień dostępu do pożądanych lub zmienianych obiektów) i bez sięgania do zewnętrznych zasobów (tj. znajdujących się poza koncentratorem lub wymagających komunikacji z urządzeniami zewnętrznymi), co dodatkowo ułatwia implementację.
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 numerowania nowych klas i obiektów

Aby uniknąć chaosu identyfikacyjnego przy okazji definiowania nowych klas jak i konkretnych obiektów z nich korzystających, zdecydowano się na wprowadzenie kilku zasad regulujących ten proces.

5.1.1 Dobór nowych identyfikatorów klas interfejsów (*class_id*)

Na potrzeby klas interfejsów obiektów realizowanych przez koncentrator i wymaganych przy komunikacji za pomocą protokołu DCSAP, wykorzystywany będzie jedynie przedział 40000–40199. Przedział ten jest częścią większego przedziału 32768–65535 przewidzianego dla grup użytkowników (*user group specific ICs*, patrz rozdz. 4 w [1], [7]).

W ramach wybranego przedziału zdefiniowane są 2 podprzedziały:

• 40000-40099 – dla klas interfejsów obiektów dedykowanych wyłącznie koncentratorom,

• 40100-40199 – dla klas interfejsów obiektów ogólnego przeznaczenia.

Numery przyznawane interfejsom klas w ramach tego dokumentu to kolejne numery z odpowiednich podprzedziałów, które jeszcze nie zostały wykorzystane.

W przypadku dostawców zewnętrznych chcących wykorzystać przedstawiony tu schemat, alokacja numerów powinna odbywać się od końca rzeczonych podprzedziałów, co pozwoli uniknąć kolizji w przypadku ewentualnych rewizji dokumentu rozszerzających standardowo wymaganą listę obiektów.

5.1.2 Dobór nowych kodów 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.

Kody OBIS przyznawane obiektom w ramach tego dokumentu to kolejne możliwie najniższe numery z pul zdefiniowanych w Tab. 2.

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	255
Sesyjne obiekty koncentratora	0	100	32-63	d	e	255
Globalne obiekty licznika realizowane przez koncentrator	0	100	64-95	d	e	255
Obiekty definiowane przez dostawców	0	100	128-255	d	e	255

Tab. 2 Reguły doboru kodów OBIS dla nowych obiektów

d, e - 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. 3 i opisane w dalszej części tego podrozdziału.

Nazwa klasy interfejsu	Numer klasy (class_id)	Kardynalność w koncentratorze	Kardynalność w liczniku
String list	40100	1…n	0…n
Meter list	40000	1	0
Device firmware	40101	1	1
Device basic information	40102	0…1	1
Device run information	40103	1	0…1
Event list	40001	1	0
DCSAP network statistics	40002	1	0

Tab. 3 Klasy interfejsów wprowadzone na potrzeby protokołu DCSAP

5.2.1 Klasa *String list*

String list		**1**	***class_id* = 40100, *version* = 0**		
Atrybuty		**Typ danych**	**Min.**	**Max.**	**Def**
1. *logical_name*	(stat.)	*octet-string*			
2. *string_list*	(stat.)	*array of octet-string*			
3. *entires_in_use*	(dyn.)	*double-long-unsigned*			
4. *max_entries*	(stat.)	*double-long-unsigned*			
Specyficzne metody:		obowiązkowe (m) / nie (o)			

Tab. 4 Klasa *String list* (*class_id* = 40100)

Atrybut	**Opis**
string_list	Lista ciągów znaków.
entries_in_use	Liczba elementów na liście.
max_entries	Maksymalna liczba elementów na liście.

Tab. 5 Opis atrybutów klasy *String list* (*class_id* = 40100)

5.2.2 Klasa *Meter list*

Meter list		**1**	***class_id* = 33000, *version* = 0**		
Atrybuty		**Typ danych**	**Min.**	**Max.**	**Def**
1. *logical_name*	(stat.)	*octet-string*			
2. *meter_table*	(dyn.)	*array of meter_list_entry*			
3. *entires_in_use*	(dyn.)	*double-long-unsigned*			
4. *max_entries*	(stat.)	*double-long-unsigned*			
Specyficzne metody:		obowiązkowe (m) / nie (o)			

Tab. 6 Klasa Meter list (*class_id* = 3300)

Atrybut	**Opis**
meter_table	Tablica struktur (patrz Listing 2) reprezentujących informacje o licznikach. Możliwy jest wybór selektywny (patrz Listing 3).
entires_in_use	Liczba wpisów w tablicy. Gdy osiąga maksimum, nowe wpisy nadpisują wpisy najdawniej aktualizowanych niewidocznych liczników.
max_entries	Maksymalna liczba wpisów w tablicy.

Tab. 7 Opis atrybutów klasy *Meter list* (*class_id* = 40000)

```
meter_list_entry ::= structure
{
       last_change_seq_id    long64-unsigned
-- numer sekwencyjny w ramach całej tabeli ostatniej zmiany rekordu
       last_change_time      date-time
-- czas ostatniej zmiany rekordu
       id                    double-long-unsigned
-- liczbowy identyfikator licznika przydzielony przez koncentrator
       manufacturer          octet-string
-- identyfikator producenta licznika (3 znaki drukowane)
       name                  octet-string
-- nazwa licznika nadana przez producenta (maks. 16 znaków drukowanych)
       present               boolean
-- licznik jest widoczny przez koncentrator
}
-- identyfikator sieciowy licznika to złożenie pól manufacturer i name
```

Listing 2 Struktura pojedynczego wpisu w atrybucie *meter_table* klasy *Meter list*

```
-- wymagany access-selector = 1, access-parameters przyjmuje wówczas postać:
recent_entries_descriptor ::= long64-unsigned
-- wybór pozycji, których last_change_seq_id jest większy od podanej wartości
```

Listing 3 Deskryptor selektywnego wyboru wpisów z atrybutu *meter_table* klasy *Meter list*

5.2.3 Klasa *Device firmware*

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

Tab. 8 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. 9 Kody statusowe aktualizacji oprogramowania jakie może przyjmować atrybut *last_update_status* klasy *Device firmware*

Atrybut	Opis
version	Wersja oprogramowania obecna na urządzeniu. W przypadku wielu niezależnie wersjonowanych modułów, powinna podawać ich poszczególne wersje.
checksum	Suma kontrolna całego oprogramowania.
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. 9.
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*.

Tab. 10 Opis atrybutów i metod klasy Device firmware (class_id = 40101)

5.2.4 Klasa *Device basic information*

Device basic information		**0...1**	***class_id* = 40102, *version* = 0**		
Atrybuty		**Typ danych**	**Min.**	**Max.**	**Def**
1. *logical_name*	(stat.)	*octet-string*			
2. *config_id*	(dyn.)	*double-long-unsigned*			
3. *passport*	(dyn.)	*octet-string*			
Specyficzne metody		obowiązkowe (m) / nie (o)			

Tab. 11 Klasa *Device basic information* (*class_id* = 40102)

Atrybut	**Opis**
config_id	Identyfikator (numer sekwencyjny) konfiguracji urządzenia.
passport	Specyficzny dla producenta ciąg danych identyfikujących parametry urządzenia. Zawiera obligatoryjnie model i wersję oprogramowania.

Tab. 12 Opis atrybutów klasy *Device basic information* (*class_id* = 40102)

5.2.5 Klasa *Device run information*

Device run information		**0...1**	***class_id* = 40103, *version* = 0**		
Atrybuty		**Typ danych**	**Min.**	**Max.**	**Def**
1. *logical_name*	(stat.)	*octet-string*			
2. *start_count*	(dyn.)	*double-long-unsigned*			
3. *last_start_time*	(dyn.)	*date-time*			
4. *last_start_status*	(dyn.)	*integer*			
5. *curr_uptime_secs*	(dyn.)	*double-long-unsigned*			
6. *prev_start_time*	(dyn.)	*date-time*			
7. *prev_start_status*	(dyn.)	*integer*			
8. *prev_uptime_secs*	(dyn.)	*double-long-unsigned*			
Specyficzne metody:		obowiązkowe (m) / nie (o)			
1. *restart()*		m			

Tab. 13 Klasa *Device run information* (*class_id* = 40103)

Atrybut	**Opis**
start_count	Liczba dotychczasowych uruchomień urządzenia.
last_start_time	Czas, kiedy rozpoczęło się ostatnie uruchamianie urządzenia.
last_start_status	Status ostatniego uruchomienia. 0 oznacza brak problemów.
curr_uptime_secs	Ile sekund działa urządzenie od ostatniego uruchomienia.
prev_start_time	Czas, kiedy rozpoczęło się poprzednie uruchamianie urządzenia.
prev_start_status	Status poprzedniego uruchomienia. 0 oznacza brak problemów.
prev_uptime_secs	Ile sekund działało urządzenie w poprzednim uruchomieniu.
restart()	Żądanie restartu urządzenia. Jeżeli rozpoczęto wcześniej procedurę aktualizacji oprogramowania, która jeszcze się nie zakończyła, żądanie restartu jest odrzucane i w odpowiedzi przekazany zostaje błąd *temporary-failure*. W przeciwnym przypadku zwracany jest kod *success* i rozpoczyna się procedura restartowania urządzenia.

Tab. 14 Opis atrybutów i metod klasy Device run information (class_id = 40103)

5.2.6 Klasa *Event list*

Event list		0…1	*class_id* = 40001, *version* = 0		
Atrybuty		**Typ danych**	**Min.**	**Max.**	**Def**
1. *logical_name*	(stat.)	*octet-string*			
2. *event_log*	(dyn.)	*array of event_list_entry*			
3. *entires_in_use*	(dyn.)	*double-long-unsigned*			
4. *max_entries*	(stat.)	*double-long-unsigned*			
Specyficzne metody		obowiązkowe (m) / nie (o)			
1. *push(event)*		m			

Tab. 15 Klasa Event list (class id = 40001)

Atrybut	**Opis**
event_log	Tablica struktur (patrz Listing 4) reprezentujących informacje o zdarzeniach. Możliwy jest wybór selektywny (patrz Listing 5).
entires_in_use	Liczba wpisów w tablicy.
max_entries	Maksymalna liczba wpisów w tablicy zdarzeń.
push(event)	Powoduje dodanie opisu zdarzenia przekazanego w argumencie do tablicy. Pole reason zostanie nadpisane wartością 255 (*EV_PUSH*). *event ::= event_list_entry*

Tab. 16 Opis atrybutów i metod klasy Event list (class_id = 40001)

```
event_list_entry ::= structure
{
       seq_id         long64-unsigned
-- numer sekwencyjny zdarzenia
       time           date-time
-- czas wystąpienia zdarzenia
       device_id      double-long-unsigned
-- liczbowy identyfikator urządzenia, którego dotyczy zdarzenie
       reason         unsigned
-- powód wystąpienia lub zapisania zdarzenia
       status         integer
-- kod statusowy stowarzyszony z powodem, 0 jeżeli nie dotyczy
       recorded_data  data
-- dodatkowe dane stowarzyszone ze zdarzeniem
       comment        octet-string
-- dodatkowy komentarz
}
```

Listing 4 Struktura pojedynczego wpisu w atrybucie *event_log* klasy *Event list*

```
-- wymagany access-selector = 1, access-parameters przyjmuje wówczas postać:
recent_entries_descriptor ::= long64-unsigned
-- wybór pozycji, których last_change_seq_id jest większy od podanej wartości
```

Listing 5 Deskryptor selektywnego wyboru wpisów z atrybutu *event_log* klasy *Event list*

KP	Symbol	Opis
0	*EV_START*	Uruchomienie urządzenia. Pola *status* i *recorded_data* przyjmują wówczas odpowiednio wartości atrybutów *last_start_status* i *start_count* obiektu klasy *Device run information*.
1	*EV_RESTART*	Zlecenie restartowania urządzenia. Pole status przyjmuje wówczas wartość pola *result* przekazywanego w odpowiedzi do klienta wywołującego metodę *restart()* obiektu klasy *Device run information* (kod *success* lub błąd *temporary-failure*). Pole *recorded_data* jest wypełniane *null-data*.
2	*EV_UPDATEINIT*	Zlecenie aktualizacji oprogramowania urządzenia. Pole status przyjmuje wówczas wartość pola *result* przekazywanego w odpowiedzi do klienta wywołującego metodę *start_update*() obiektu klasy Device firmware (kod *success* lub błąd *temporary-failure*). Pole *recorded_data* przyjmuje wówczas wartość atrybutu *last_update_id* obiektu klasy *Device firmware* (nową, tj. już zinkrementowaną, w przypadku sukcesu lub obecną w przypadku błędu).
3	*EV_UPDATEFINI*	Zakończenie aktualizacji oprogramowania urządzenia. Pola *status* i *recorded_data* przyjmują wówczas odpowiednio wartości atrybutów *last_update_status* i *last_update_id* obiektu klasy *Device firmware*.
4	*EV_METERSTAT*	Zmiana obecności licznika. Pole status przyjmuje wówczas nową wartość pola *present* z odpowiedniego rekordu *meter_list_entry* listy liczników. Pole *recorded_data* jest wypełniane strukturą (*structure*) zawierającą pola *manufacturer* i *name* z tego samego rekordu.
255	*EV_PUSH*	Zdarzenie będące następstwem wykonania metody *push()*.

Tab. 17 Kody powodów wystąpienia lub zapisania zdarzenia jakie może przyjmować pole reason struktury event_list_entry i opisy pól stowarzyszonych

5.2.7 Klasa DCSAP *network statistics*

DCSAP *network statistics*		**0…1**	***class_id* = 40002, *version* = 0**		
Atrybuty		**Typ danych**	**Min.**	**Max.**	**Def**
1. *logical_name*	(stat.)	*octet-string*			
2. *sessions_opened*	(dyn.)	*long64-unsigned*			
3. *sessions_active*	(dyn.)	*long64-unsigned*			
4. *bytes_received*	(dyn.)	*long64-unsigned*			
5. *bytes_sent*	(dyn.)	*long64-unsigned*			
6. *messages_received*	(dyn.)	*long64-unsigned*			
7. *messages_sent*	(dyn.)	*long64-unsigned*			
8. *dcsap_reqs_completed*	(dyn.)	*long64-unsigned*			
9. *meter_reqs_completed*	(dyn.)	*long64-unsigned*			
Specyficzne metody		obowiązkowe (m) / nie (o)			

Tab. 18 Klasa DCSAP network statistics (class id = 40002)

Atrybut	**Opis**
sessions_opened	Liczba wszystkich dotychczas otwartych sesji DCSAP w ramach bieżącego uruchomienia koncentratora.
sessions_active	Liczba obecnie utrzymywanych sesji DCSAP w ramach bieżącego uruchomienia koncentratora.
bytes_received	Liczba bajtów odebranych w warstwie aplikacji ze wszystkich sesji DCSAP w ramach bieżącego uruchomienia koncentratora.
bytes_sent	Liczba bajtów wysłanych w warstwie aplikacji ze wszystkich sesji DCSAP w ramach bieżącego uruchomienia koncentratora.
messages_received	Liczba komunikatów odebranych we wszystkich sesjach DCSAP w ramach bieżącego uruchomienia koncentratora.
messages_sent	Liczba komunikatów wysłanych we wszystkich sesjach DCSAP w ramach bieżącego uruchomienia koncentratora.
dcsap_reqs_completed	Liczba zrealizowanych zleceń skierowanych do obiektów realizowanych przez koncentrator we wszystkich sesjach DCSAP w ramach bieżącego uruchomienia koncentratora.
meter_reqs_completed	Liczba zrealizowanych zleceń skierowanych do obiektów realizowanych przez liczniki we wszystkich sesjach DCSAP w ramach bieżącego uruchomienia koncentratora.

Tab. 19 Opis atrybutów klasy *DCSAP network statistics* (*class id* = 4002)

5.3 Globalne obiekty koncentratora

Wszystkie globalne obiekty koncentratora (przedstawione w Tab. 20), poza obiektem *Data concentrator network statistics*, są persystentne, tzn. restart koncentratora nie powoduje ich resetowania.

Obiekty globalne koncentratora wymagane przez protokół DCSAP	Klasa COSEM	Kod OBIS					
		A	B	C	D	E	F
Data concentrator meter list	40000	0	100	0	0	0	255
Data concentrator firmware	40101	0	100	0	0	1	255
Data concentrator run information	40103	0	100	0	0	2	255
Data concentrator event list	40001	0	100	0	0	3	255
Data concentrator NTP server list	40100	0	100	0	0	4	255
Data concentrator network statistics	40002	0	100	0	0	5	255

Tab. 20 Globalne obiekty koncentratora

5.3.1 Obiekt *Data concentrator meter list*

Najważniejszą funkcją koncentratora jest wykrywanie i rejestrowanie nowych liczników oraz utrzymywanie listy liczników aktualnie działających w jego zasięgu. Protokół DCSAP definiuje sposób w jaki koncentrator dzieli się tą listą z systemem akwizycji.

Zakładamy, że koncentrator posiada wystarczająco duży bufor na listę aktualnie podłączonych liczników oraz dodatkowo jest w stanie przechować na tej liście pewną, wystarczającą do poprawnego działania systemu akwizycji, liczbę rekordów liczników usuniętych (nieaktywnych). Lista ta posiada co najwyżej jeden wpis dotyczący danego licznika. Jeżeli licznik zostaje wyłączony z systemu, to aktualny rekord licznika zostaje odpowiednio oznaczony. Z każdym rekordem związany jest czas ostatniej aktualizacji oraz globalny numer sekwencyjny zmiany, dzięki czemu system akwizycji może pozyskiwać informacje o zmianie tej listy poprzez zapytanie o rekordy nowsze, niż ostatnio otrzymane. Taki addytywny algorytm odczytywania listy liczników jest poprawny pod warunkiem utrzymania przez koncentrator monotoniczności numeru sekwencyjnego zmiany.

Jeżeli wszystkie dostępne rekordy listy będą już zajęte przez stare wpisy dotyczące usuniętych liczników, wszystkie nowo zarejestrowane powinny nadpisywać najstarsze z nich (o najniższych numerach sekwencyjnych). Przy założeniu, że liczba dostępnych rekordów będzie wystarczająco duża a częstotliwość uzupełniania informacji przez system akwizycji będzie duża w stosunku do częstotliwości pojawiania się całkowicie nowych liczników, mechanizm jest w stanie zapewnić całkowitą synchronizację listy po stronie systemu akwizycji.

Oprócz globalnego numeru sekwencyjnego zmiany, czasu ostatniej aktualizacji oraz liczbowego identyfikatora licznika przydzielonego przez koncentrator, w każdym rekordzie listy znajduje się kod producenta licznika, nazwa licznika nadana przez producenta oraz oznaczenie obecności licznika w systemie.

Należy podkreślić, że zmiany w obiekcie *Data concentrator meter list* muszą być realizowane atomowo. Nie jest dopuszczalne by było możliwe wyczytanie rekordu dotyczącego któregokolwiek licznika, który nie został jeszcze w pełni wypełniony (w przypadku nowo wykrytego licznika) lub w pełni zmieniony (w przypadku licznika, który przestał być widoczny lub jest widoczny ponownie). Czas ostatniej zmiany rekordu (lub jego dodania) jest aktualizowany wraz z pozostałymi polami struktury *meter_list_entry*.

5.3.2 Obiekt *Data concentrator 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. 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.

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.

5.3.3 Obiekt *Data concentrator run information*

Koncentrator musi udostępniać podstawowe informacje na temat bieżącego i poprzedniego uruchomienia oraz liczby wszystkich uruchomień. Są to informacje diagnostyczne, które, przy okresowym sprawdzaniu, pozwalają wykryć problemy ze zbyt częstym restartowaniem się urządzeń lub ich nieprawidłowym działaniem. Ponadto obiekt ten pozwala wykonać restart koncentratora.

5.3.4 Obiekt *Data concentrator event list*

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.

Koncentrator zatrzaskuje następujące zdarzenia: uruchomienie koncentratora, zlecenie jego restartu, zlecenie aktualizacji oprogramowania, zakończenie aktualizacji oprogramowania i zmianę obecności licznika (np. pojawienie się nowego, odłączenie się starego, a także ponowne pojawienie się wcześniej odłączonego). Obsługiwane jest także zatrzaskiwanie zdarzeń generowanych przez klienta wywołującego metodę *push()* tego obiektu, do której przekazywany jest opis zdarzenia zgodny ze strukturą *event_list_entry* (patrz Listing 4). By uniemożliwić tworzenie nieprawdziwych zdarzeń, pole *reason* tej struktury jest podmieniane na wartość *EV_PUSH* w przypadku tak generowanych zdarzeń.

Jeżeli wszystkie dostępne rekordy listy będą już zajęte, nowe zdarzenia powinny nadpisywać najstarsze obecne na liście (o najniższych numerach sekwencyjnych). Każde nowe zdarzenie dostaje nowy (zinkrementowany) numer sekwencyjny, a więc dziennik zdarzeń jest de facto buforem cyklicznym.

5.3.5 Obiekt *Data concentrator NTP server list*

Koncentratory muszą umożliwiać ustawienie listy adresów serwerów NTP, aby możliwe było używanie tych samych serwerów czasu w całej infrastrukturze. Obiekt ten pozwala zdefiniować owe serwery w postaci tablicy ciągów znaków reprezentujących nazwy domenowe bądź adresy IP.

5.3.6 Obiekt *Data concentrator network statistics*

Koncentrator musi zbierać podstawowe statystyki sieciowe z komunikacji prowadzonej za pomocą protokołu DCSAP. Statystyki te nie są persystentne i są zbierane od nowa po każdym uruchomieniu koncentratora.

Śledzeniu podlegają następujące wielkości: liczba wszystkich dotychczas otwartych sesji, liczba obecnie utrzymywanych sesji, liczba bajtów odebranych w warstwie aplikacji (ze wszystkich sesji łącznie), liczba bajtów wysłanych (ze wszystkich sesji łącznie), liczba odebranych komunikatów (we wszystkich sesjach łącznie), liczba wysłanych komunikatów (we wszystkich sesjach łącznie), liczba zrealizowanych zleceń skierowanych do obiektów realizowanych przez koncentrator (we wszystkich sesjach łącznie) oraz liczba zrealizowanych zleceń skierowanych do obiektów realizowanych przez liczniki (we wszystkich sesjach łącznie).

5.4 Sesyjne obiekty koncentratora

Wszystkie sesyjne obiekty koncentratora (przedstawione w Tab. 21) są tymczasowe, powoływane do życia wraz z utworzeniem każdej nowej sesji i usuwane po jej zakończeniu. Obiekty sesyjne w różnych sesjach są od siebie niezależne.

Obiekty sesyjne koncentratora wymagane przez protokół DCSAP	Klasa COSEM	Kod OBIS					
		A	B	C	D	E	F
Data concentrator meter data caching enable	1	0	100	32	0	0	255
Data concentrator asynchronous notification enable	1	0	100	32	0	1	255

Tab. 21 Sesyjne obiekty koncentratora

5.4.1 Obiekt *Data concentrator meter data caching 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.

Atrybut *value* tego obiektu jest typu *boolean* i domyślnie przyjmuje wartość *true*.

5.4.2 Obiekt *Data concentrator asynchronous 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.

Z założenia system akwizycji używa pojedynczej sesji komunikacyjnej z koncentratorem, ale nic nie stoi na przeszkodzie, żeby na przykład notyfikacje asynchroniczne obsługiwać w dedykowanej sesji.

Atrybut *value* tego obiektu jest typu *boolean* i domyślnie przyjmuje wartość *false*.

5.5 Globalne obiekty liczników realizowane przez koncentrator

Wszystkie globalne obiekty liczników realizowane przez koncentrator (przedstawione w Tab. 22) są persystentne, tzn. restart koncentratora nie powoduje ich resetowania.

Obiekty globalne licznika wymagane przez protokół DCSAP	Klasa COSEM	Kod OBIS					
		A	B	C	D	E	F
Meter information	40102	0	100	64	0	0	255
Meter firmware	40101	0	100	64	0	1	255

Tab. 22 Globalne obiekty liczników realizowane przez koncentrator

5.5.1 Obiekt *Meter information*

Obiekt przechowujący identyfikator konfiguracji licznika (inkrementowany z każdą jego zmianą) oraz pole zawierające informacje dotyczące typu licznika i jego właściwości, tzw. paszport. Strukturę paszportu każdy producent definiuje samodzielnie. Obowiązkowymi elementami tej struktury muszą być następujące informacje:

- model licznika,
- wersja oprogramowania licznika.

5.5.2 Obiekt *Meter firmware*

Wszystkie liczniki muszą umożliwić 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.

6 Wykorzystanie protokołu DCSAP

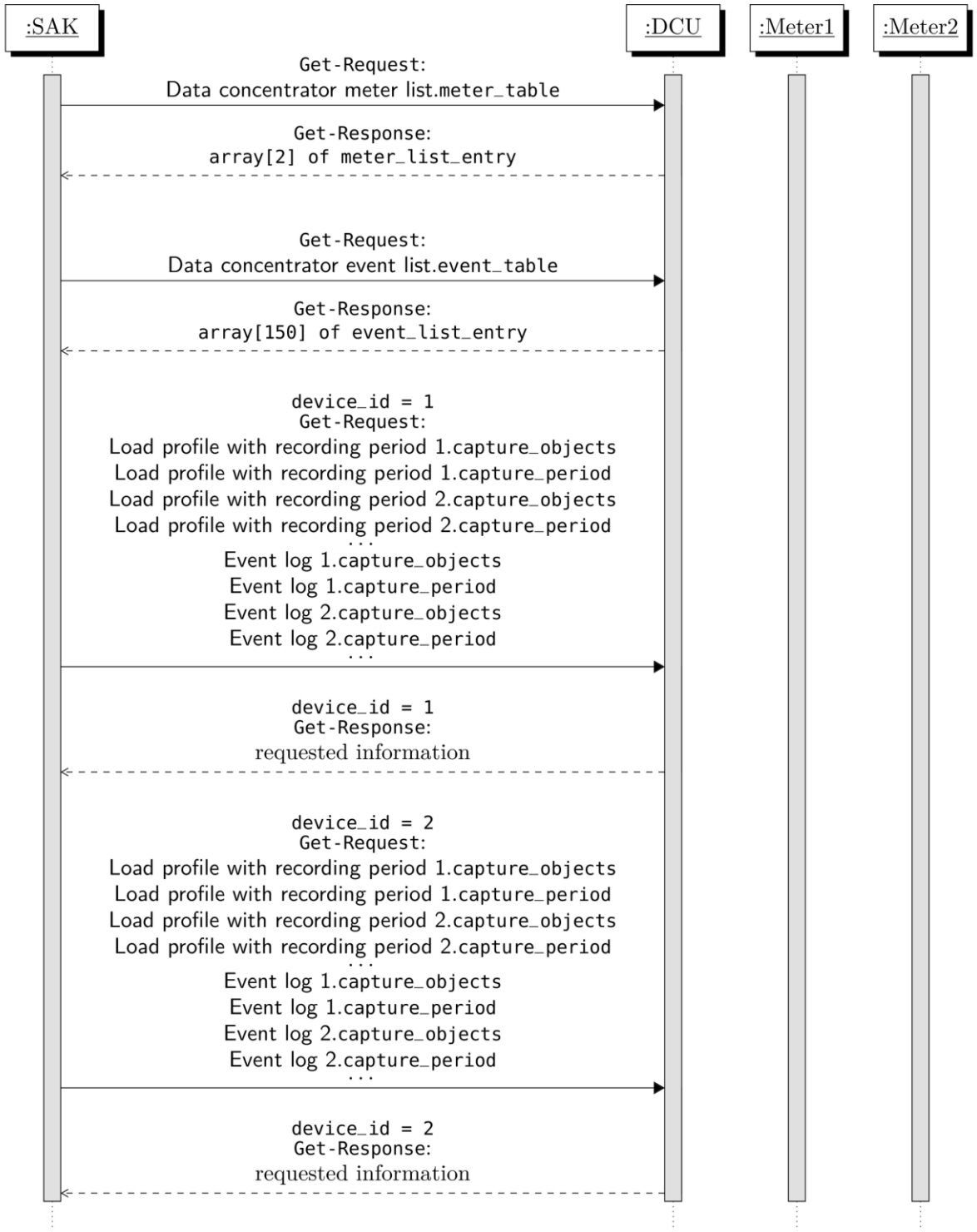
W tym rozdziale przedstawiono najbardziej typowe przykłady wykorzystania protokołu DCSAP w procesie prowadzenia akwizycji.

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.

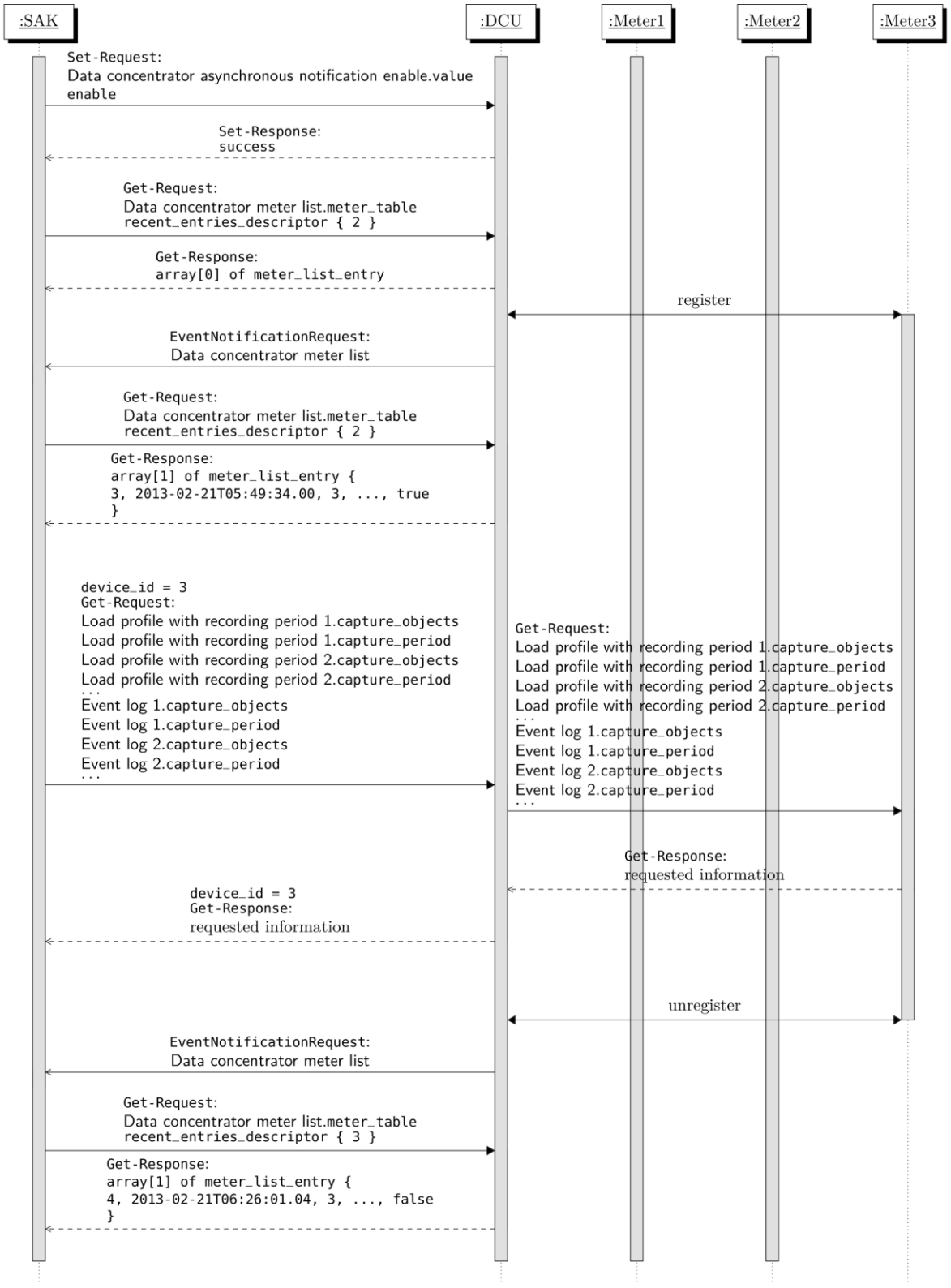
Jeżeli przez 5 minut nie są wysyłane polecenia do koncentratora, system akwizycji powinien wysłać pusty komunikat i oczekiwać na odpowiedź (patrz Rys. 5, Rys. 6). W ten sposób wykryte zostanie ewentualne zerwanie połączenia spowodowane nieplanowanym sprzętowym restartem koncentratora lub, w przypadku 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).



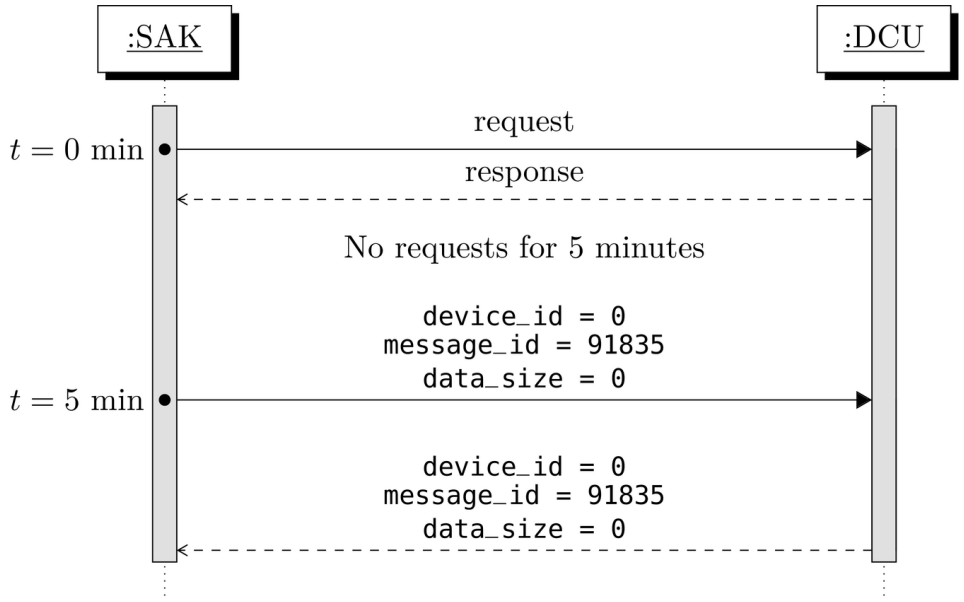
Rys. 3 Diagram sekwencji operacji wykonywanych przez system akwizycji po rozpoczęciu pierwszej sesji DCSAP z koncentratorem

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

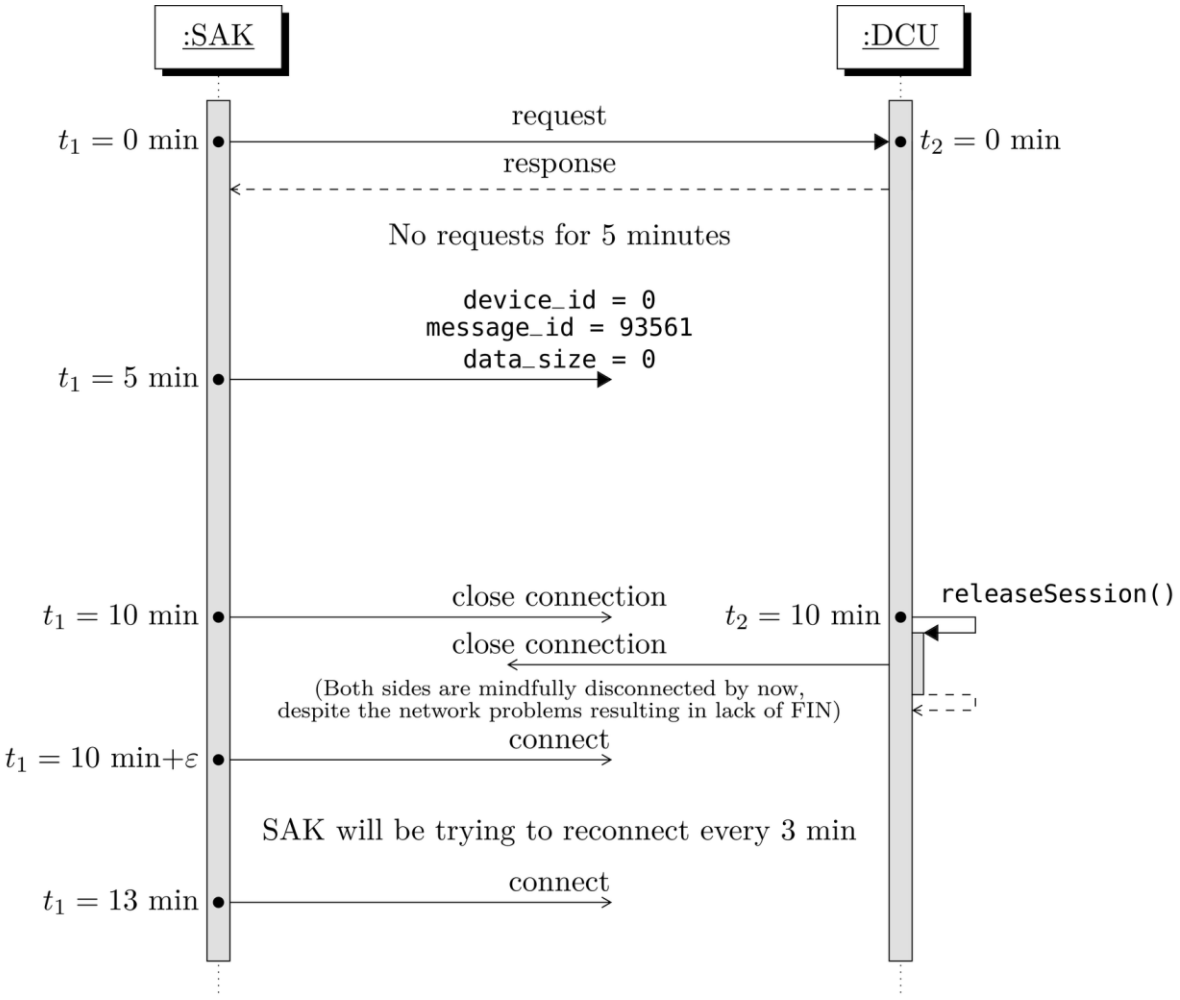


Rys. 4 Diagram sekwencji operacji wykonywanych przez system akwizycji po rozpoczęciu sesji DCSAP z koncentratorem

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.

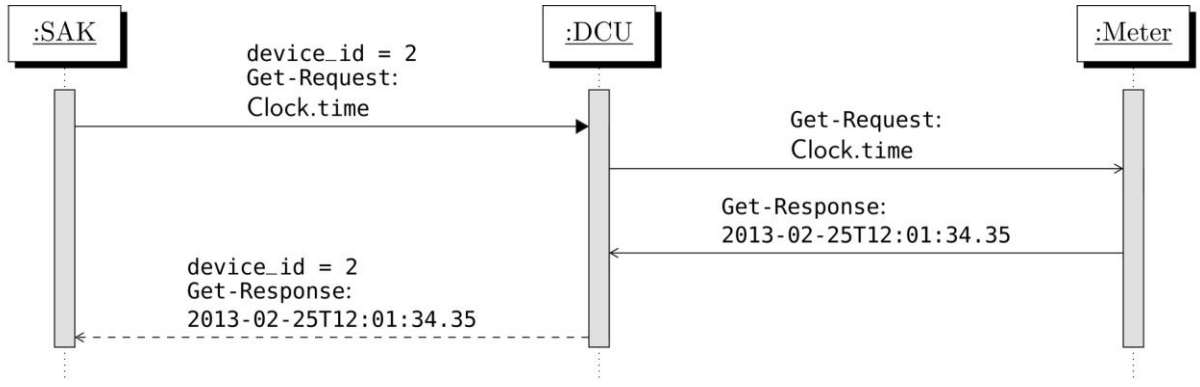
6.3 Synchronizacja topologii

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).

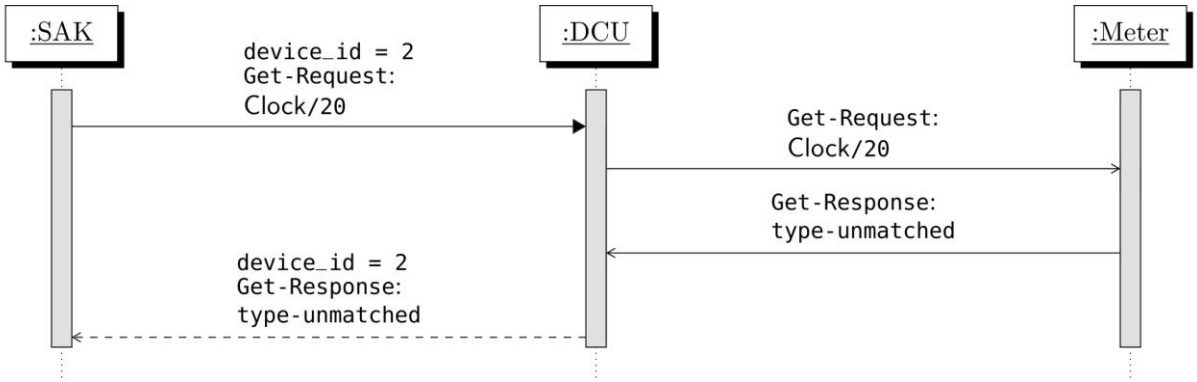
6.4 Odczyt rejestru układu

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).



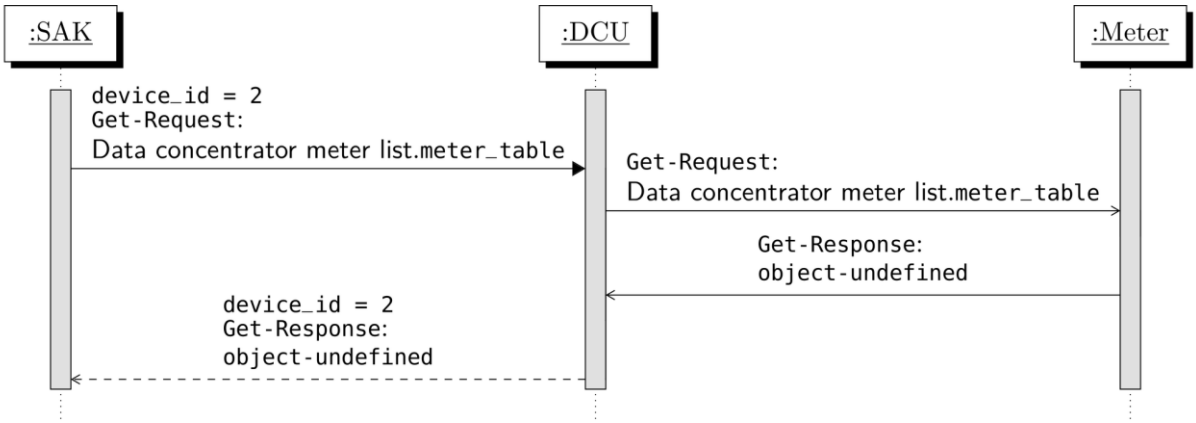
Rys. 7 Diagram sekwencji odczytu z licznika – wariant pomyślny

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



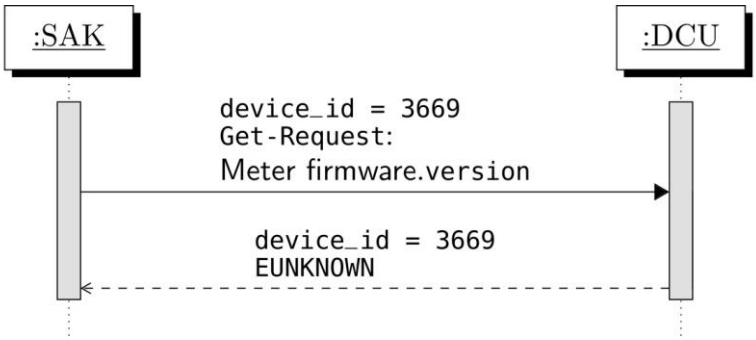
Rys. 8 Diagram sekwencji odczytu z licznika – wariant nieprawidłowego atrybutu

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



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

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

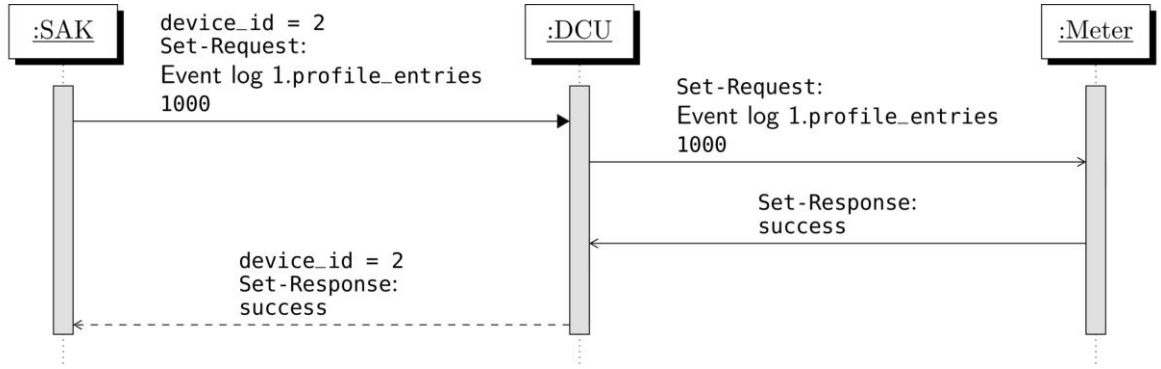


Rys. 10 Diagram sekwencji odczytu z licznika – wariant nieznanego identyfikatora licznika

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

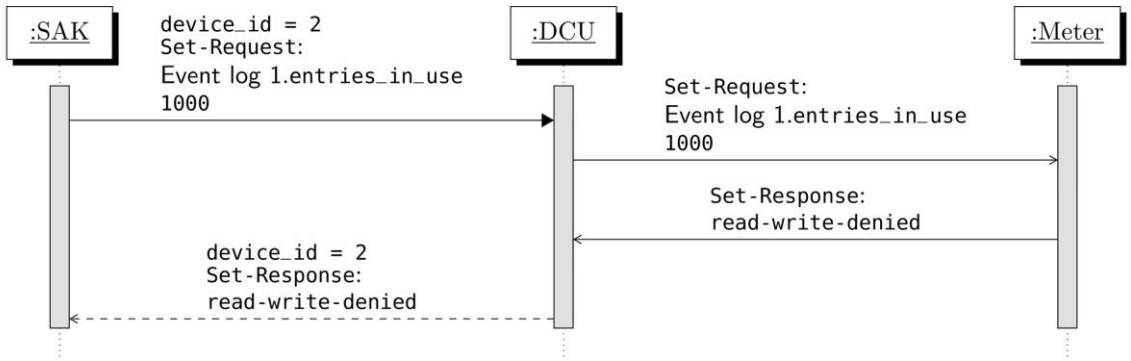
6.5 Zapis rejestru układu

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

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



Rys. 12 Diagram sekwencji zapisu do licznika – wariant niepomyślny

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

6.6 Konfiguracja układu

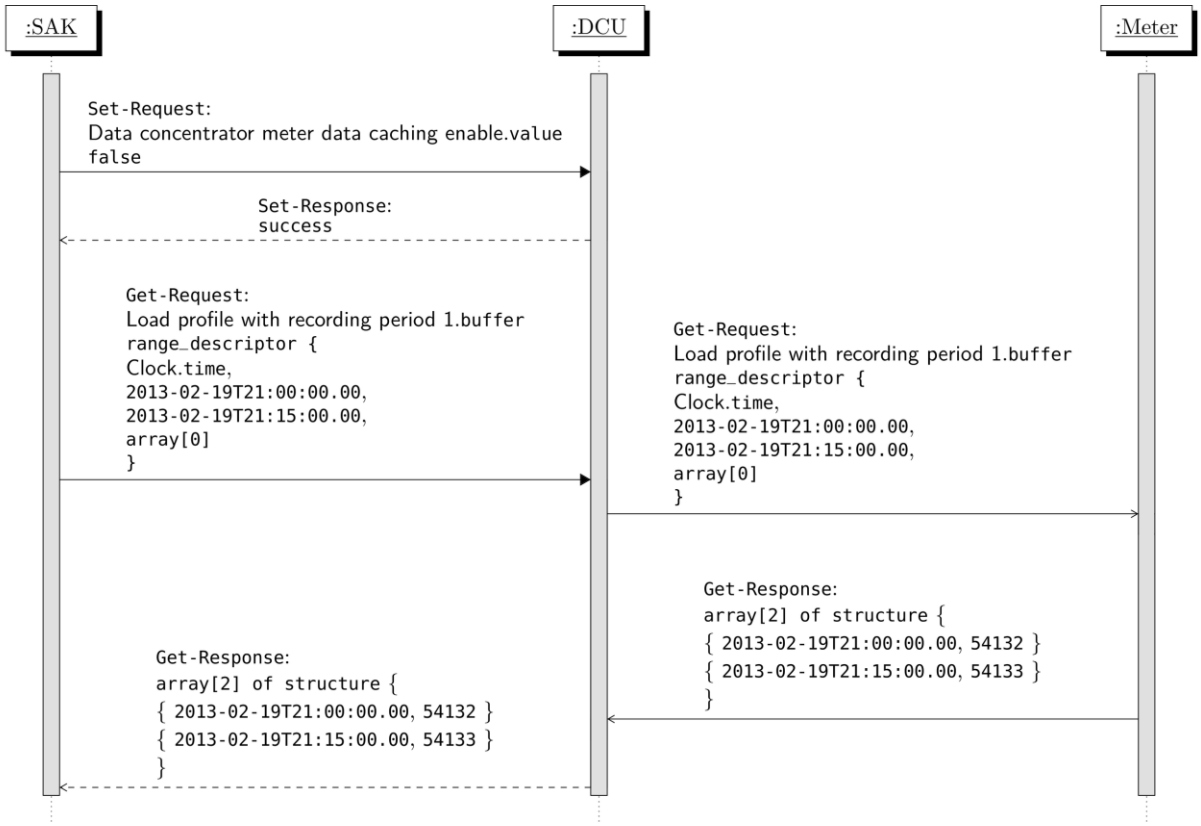
Konfiguracja układu polega na wysłaniu zestawu poleceń zapisu kompletu rejestrów danego układu. W przypadku niekompletnego potwierdzenia lub selektywnego niepowodzenia można ponowić cały zestaw operacji lub w celu optymalizacji wyłączyć z tego zestawu te operacje, na które przyszło potwierdzenie.

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 *Data concentrator 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.

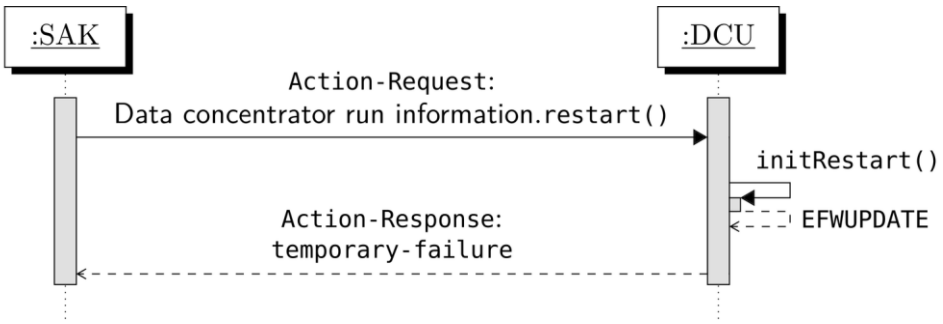


Rys. 13 Diagram sekwencji zapisu do koncentratora i odczytu bezpośredniego z układu

Założono, że koncentrator komunikuje się z licznikiem w warstwie aplikacji za pomocą protokołu COSEM, a atrybut *capture_objects* obiektu *Load profile with recording period 1* zawiera tylko czas i wartość A+.

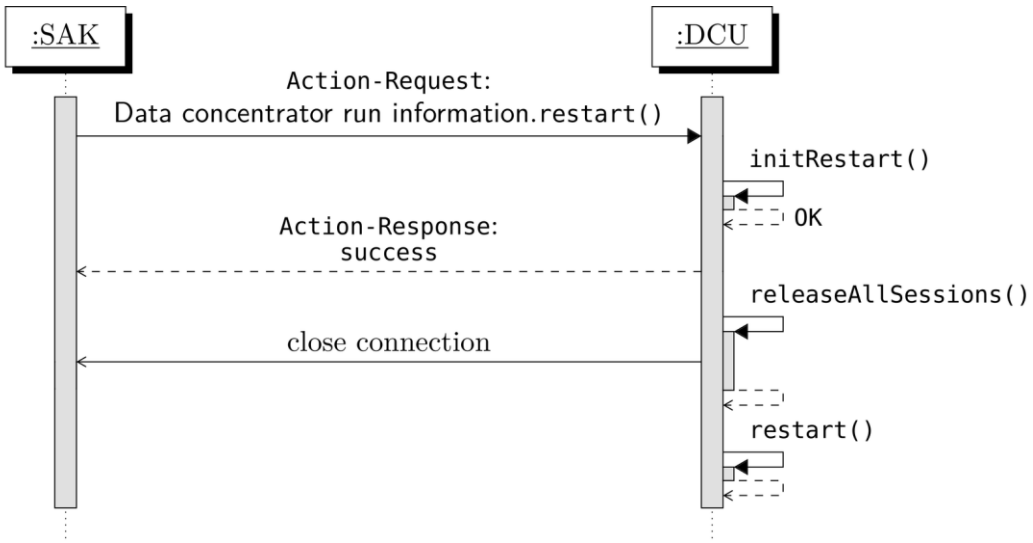
6.9 Restart koncentratora

Procedurę restartu koncentratora inicjuje się poprzez wywołanie metody *restart()* obiektu *Data concentrator run information*. 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

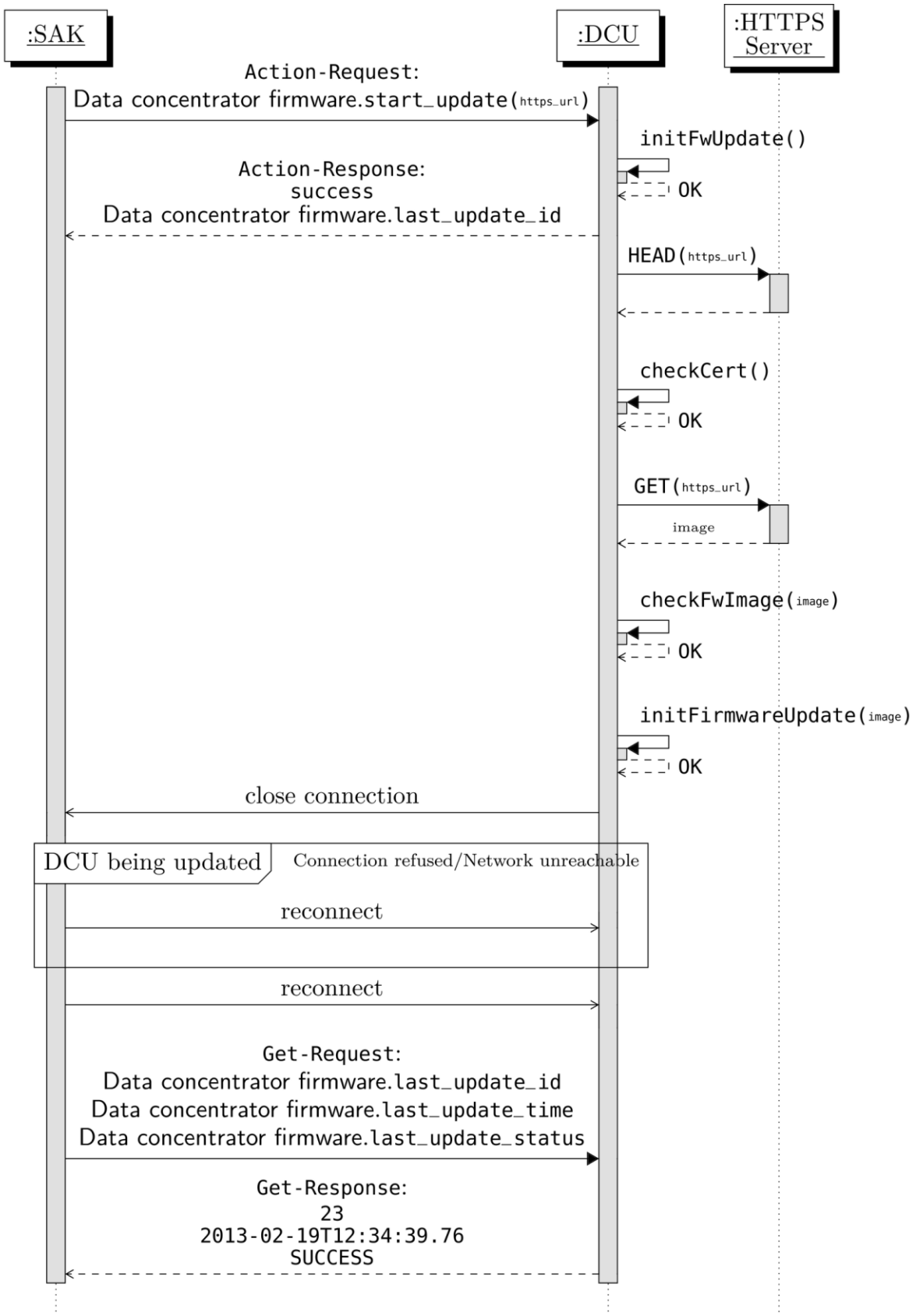
Założono, że przed próbą restartu koncentratora zlecono operację aktualizacji oprogramowania, która jeszcze się nie zakończyła.



Rys. 15 Diagram sekwencji restartowania koncentratora – wariant pomyślny

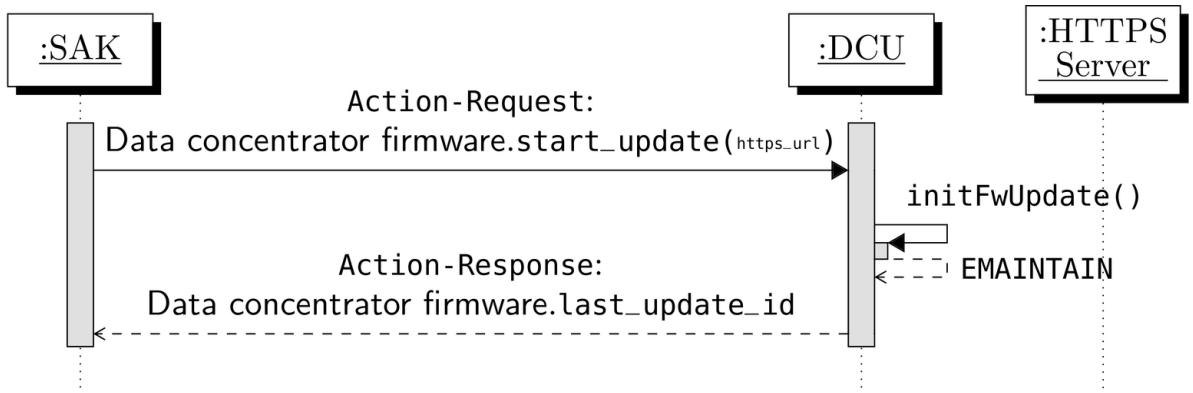
6.10 Aktualizacja oprogramowania koncentratora

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 i jeżeli może rozpocząć się aktualizacja, odpowiada sukcesem na otrzymane żądanie aktualizacji z systemu akwizycji. 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.



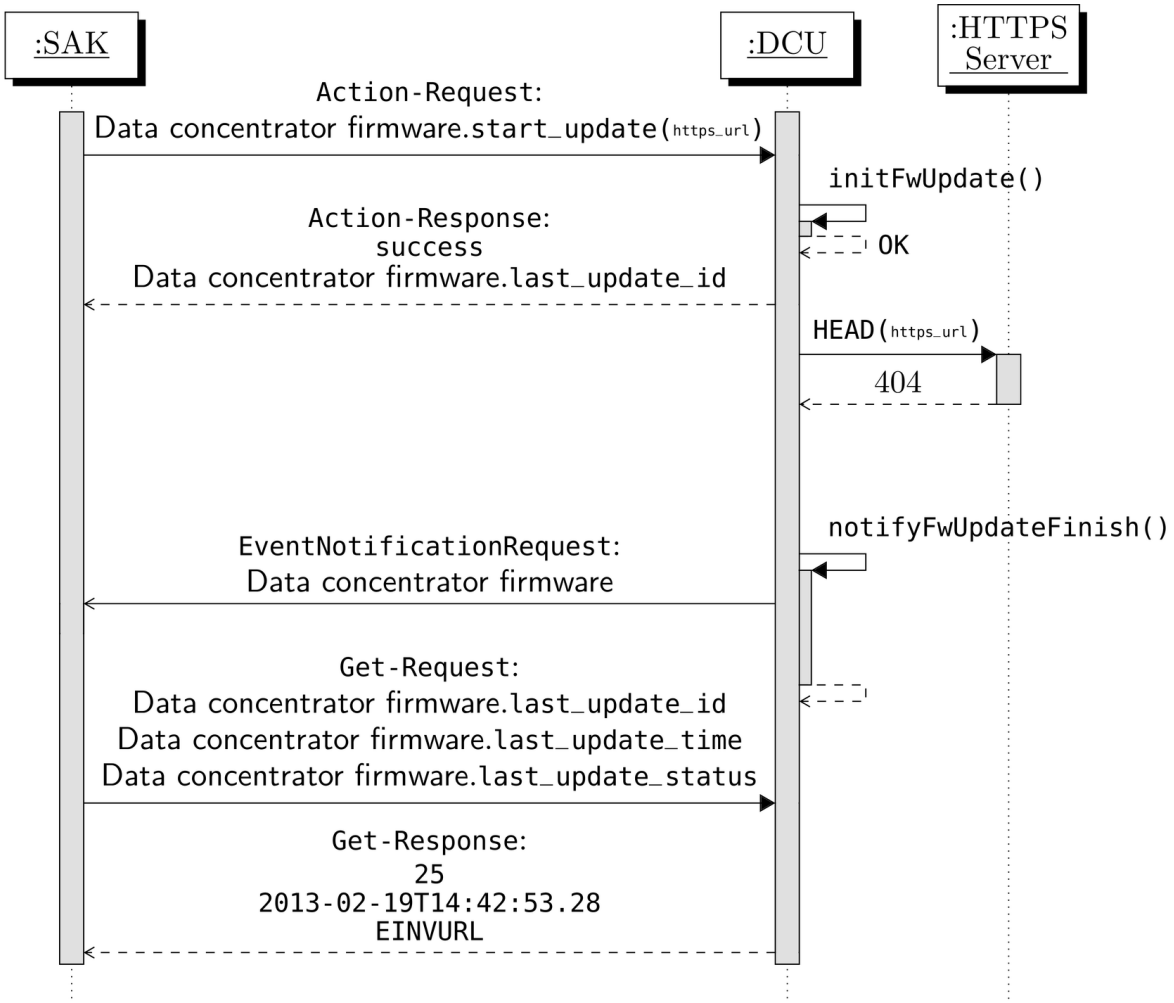
Rys. 16 Diagram sekwencji aktualizacji oprogramowania koncentratora – wariant pomyślny

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



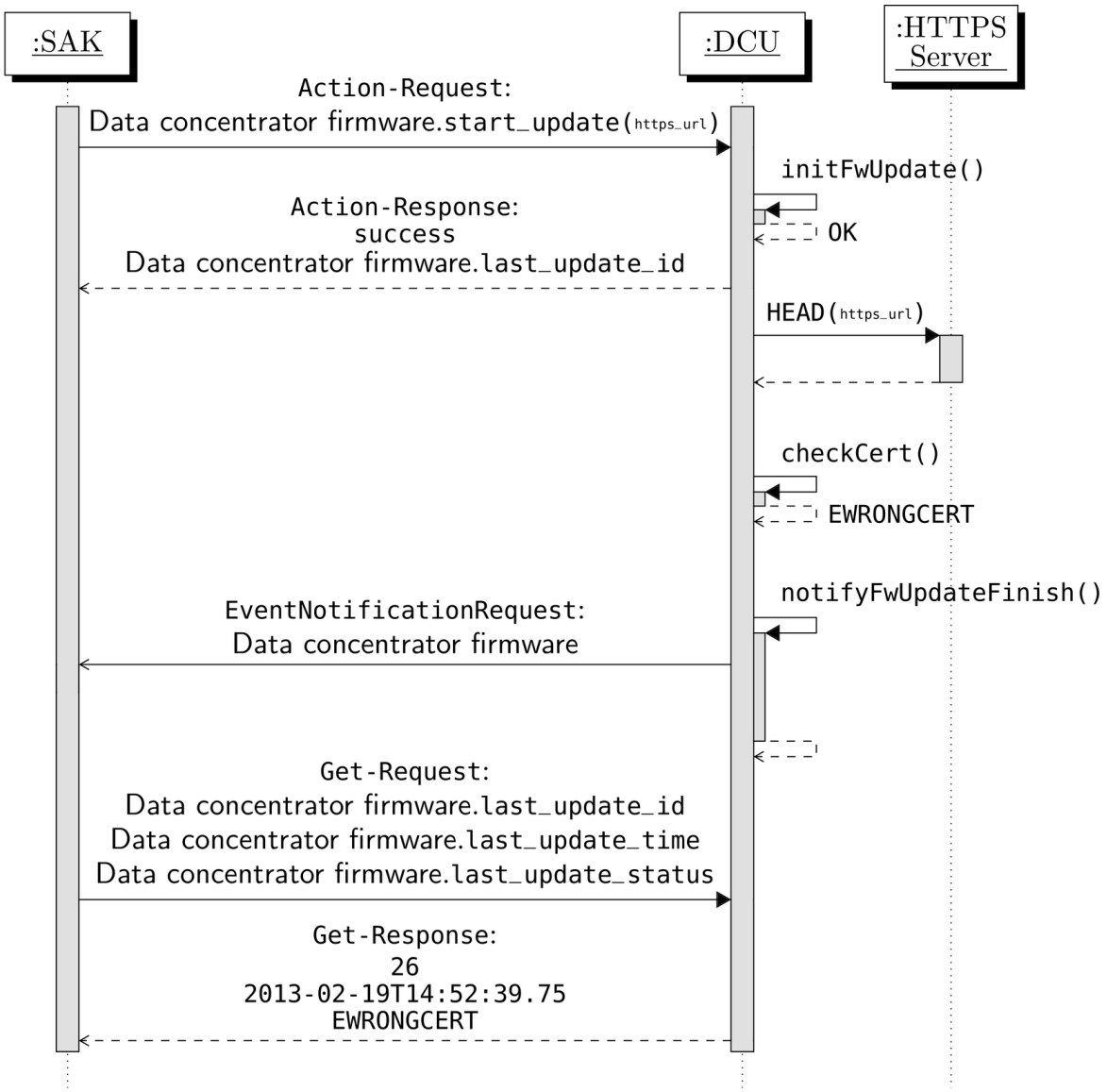
Rys. 17 Diagram sekwencji aktualizacji oprogramowania koncentratora – 1. wariant niepomyślny

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



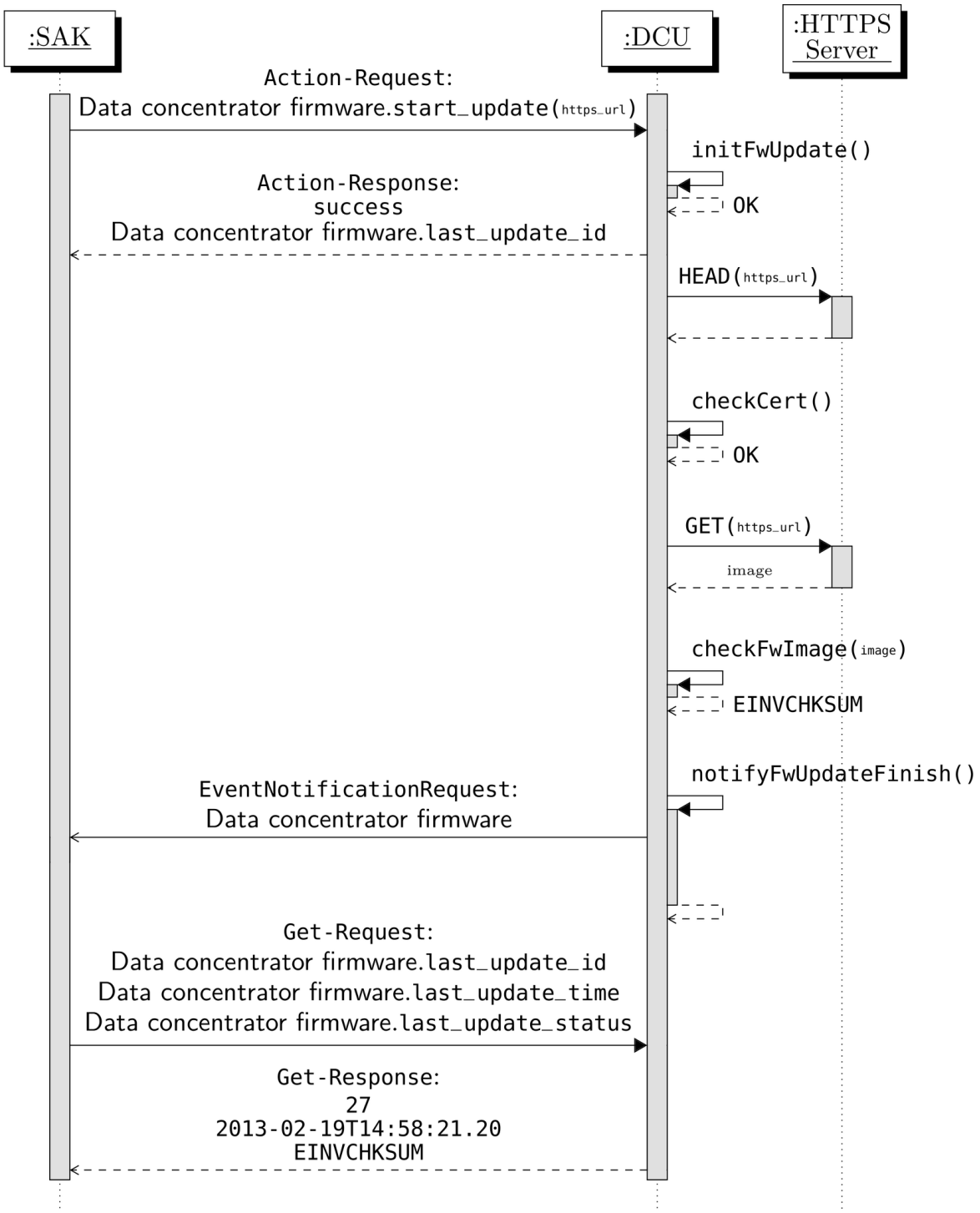
Rys. 18 Diagram sekwencji aktualizacji oprogramowania koncentratora – 2. wariant niepomyślny

Założono, że podany adres obrazu jest nieprawidłowy.



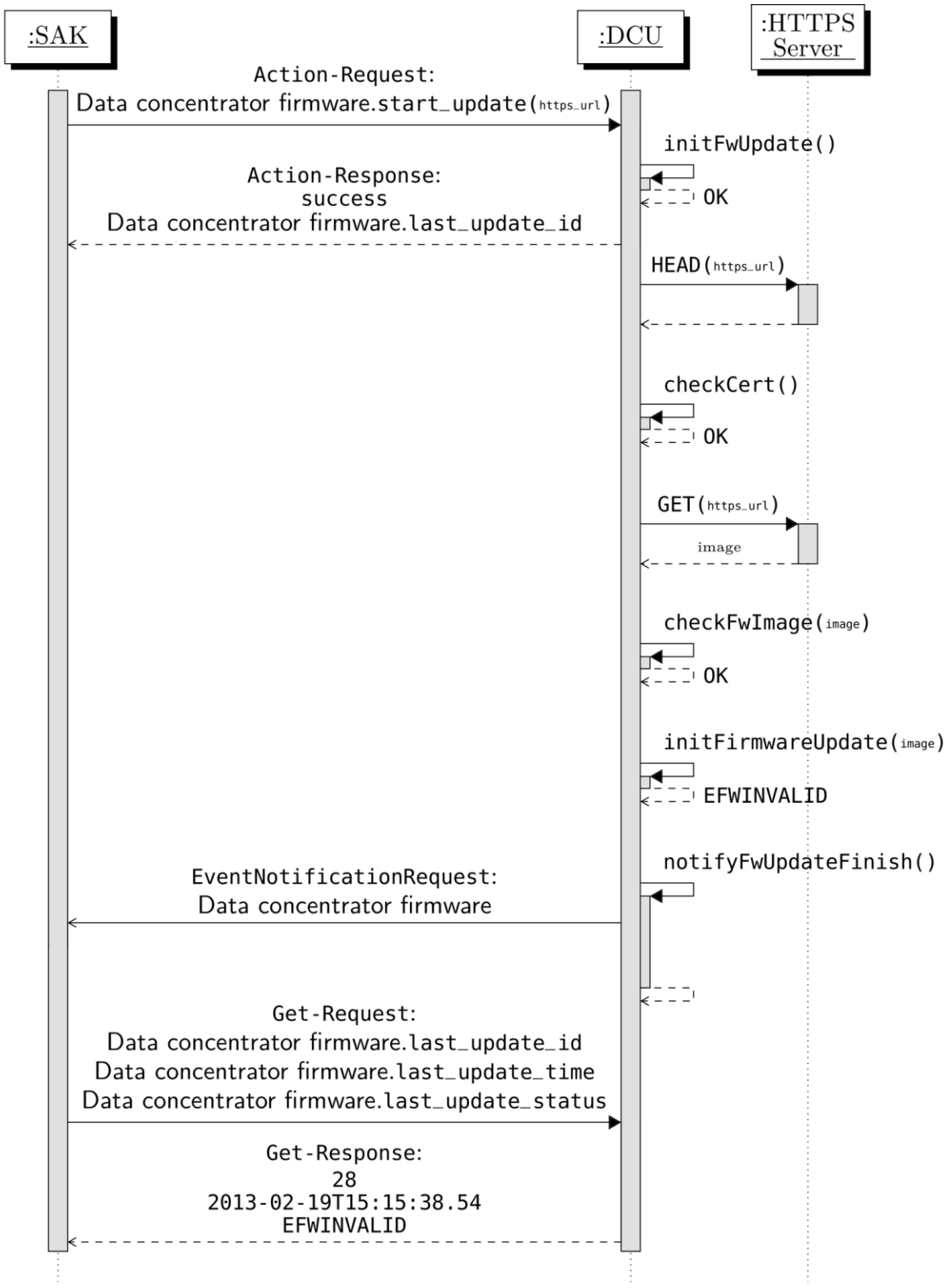
Rys. 19 Diagram sekwencji aktualizacji oprogramowania koncentratora – 3. wariant niepomyślny

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



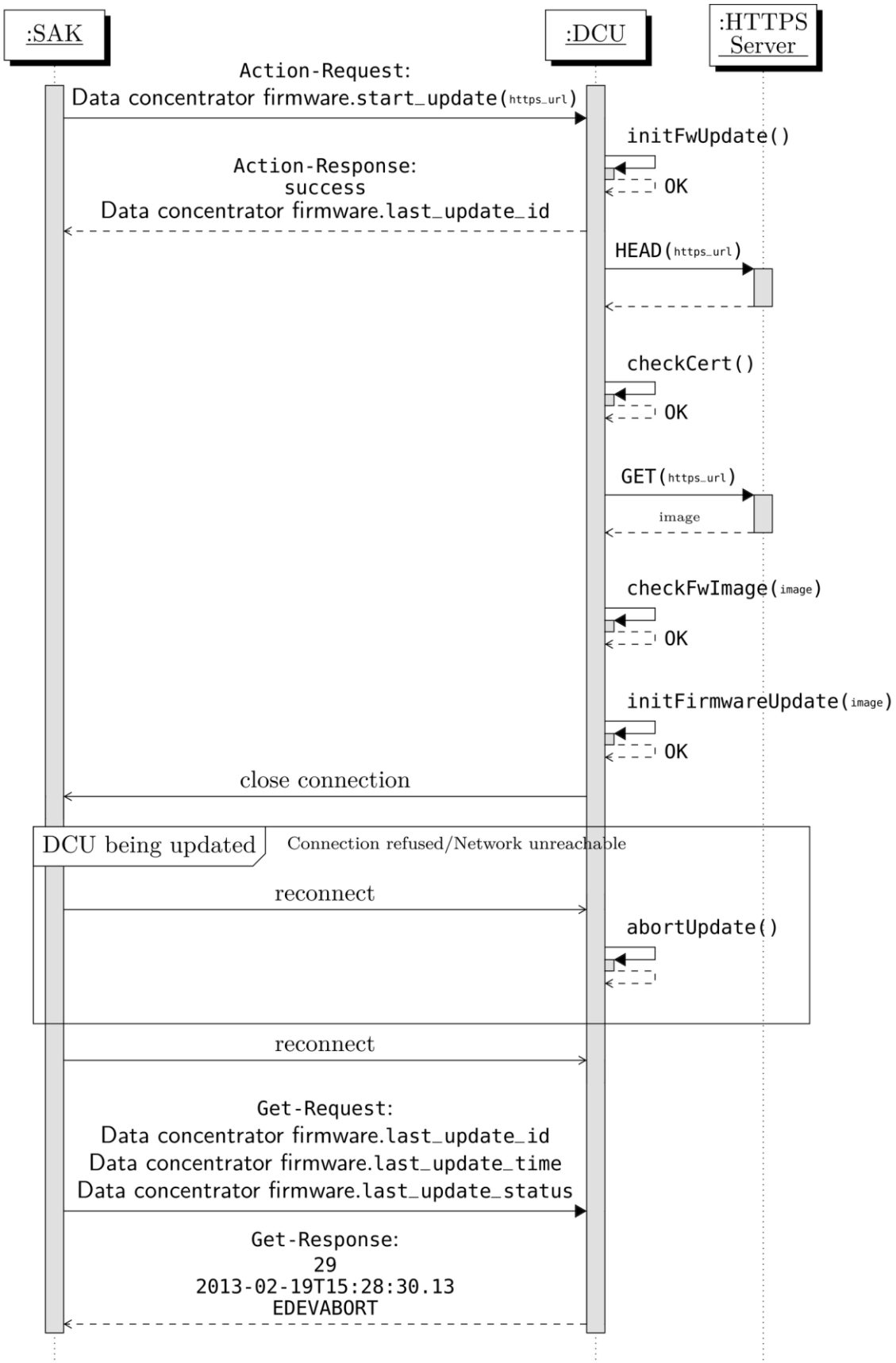
Rys. 20 Diagram sekwencji aktualizacji oprogramowania koncentratora – 4. wariant niepomyślny

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.

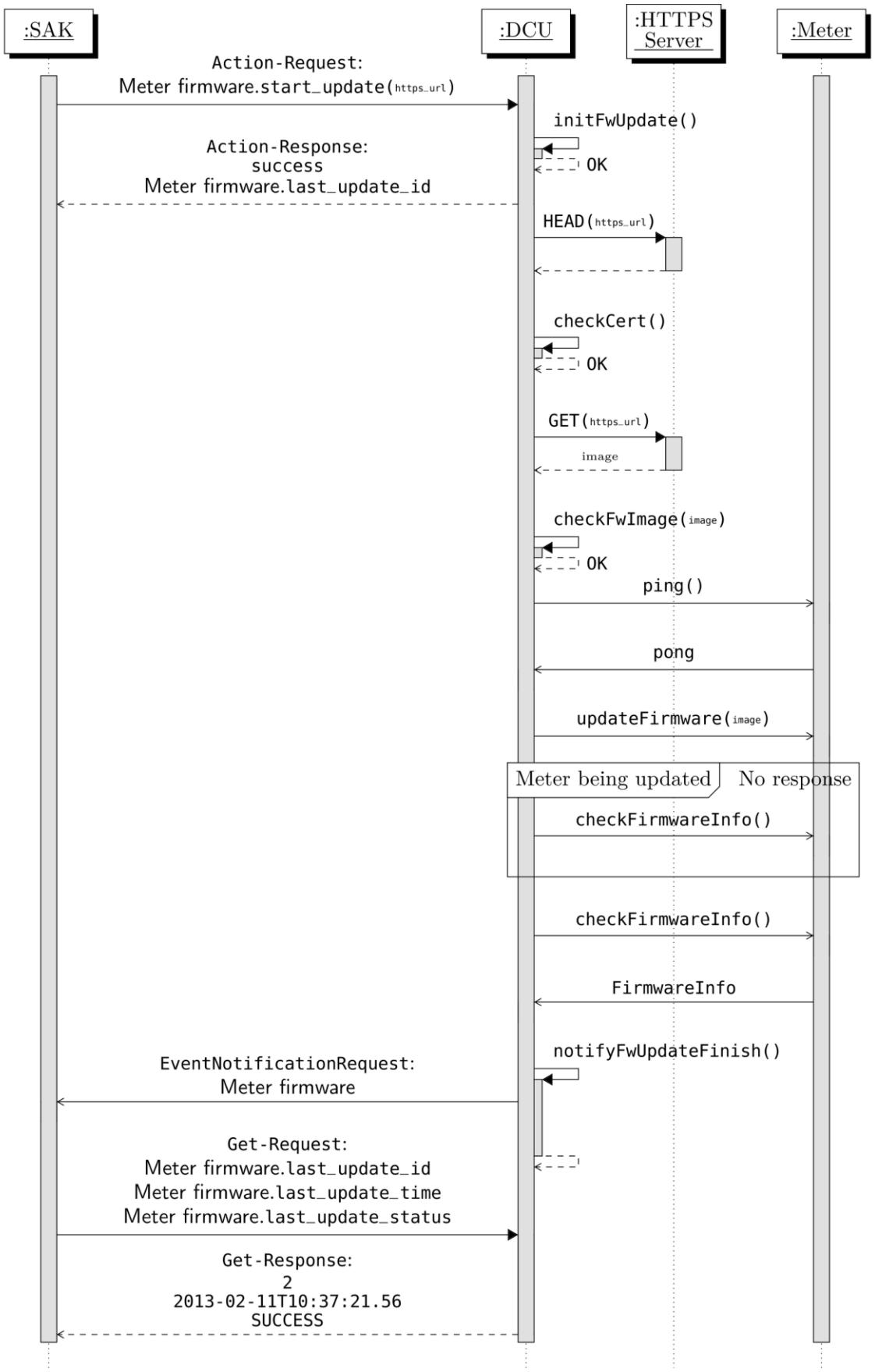


Rys. 22 Diagram sekwencji aktualizacji oprogramowania koncentratora – 6. wariant niepomyślny

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

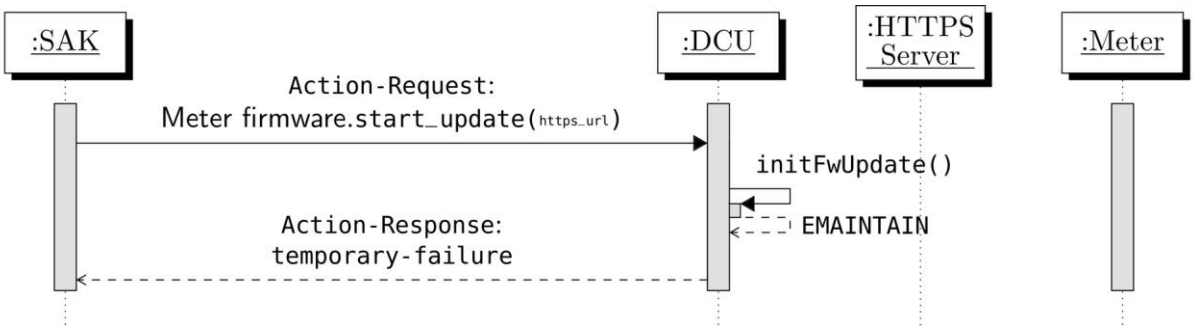
6.11 Aktualizacja oprogramowania licznika

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.



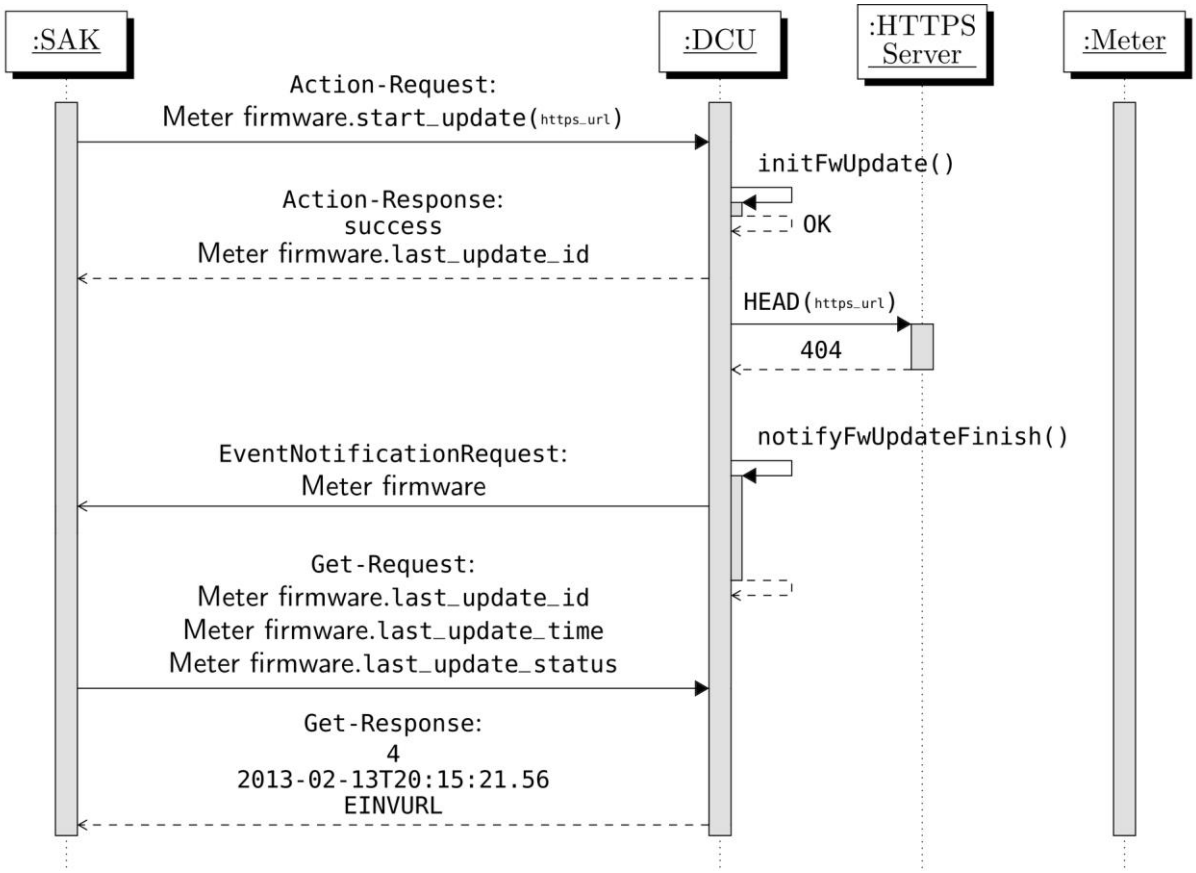
Rys. 23 Diagram sekwencji aktualizacji oprogramowania licznika – wariant pomyślny

Założono, że wszystkie kroki przebiegły pomyślnie.



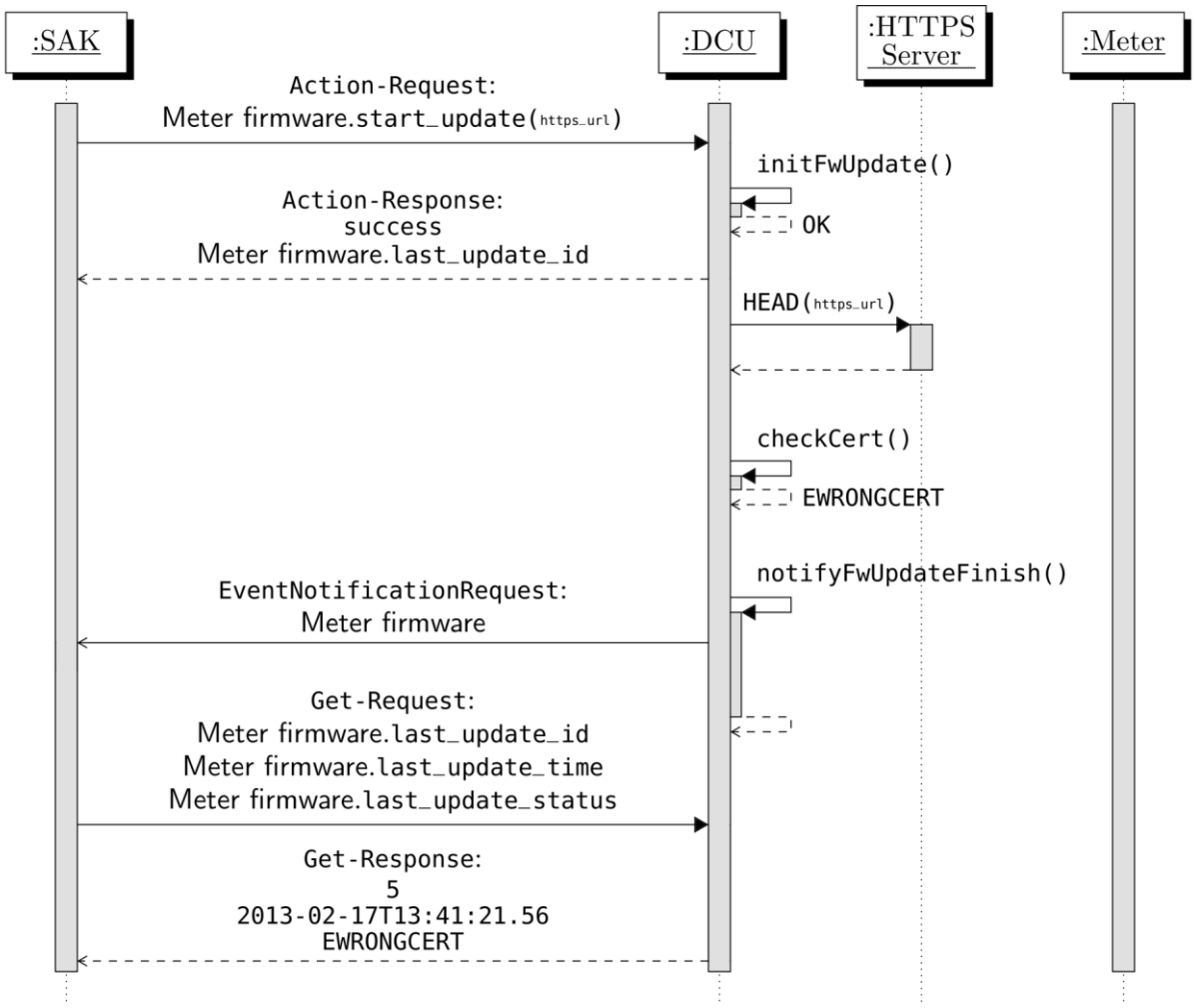
Rys. 24 Diagram sekwencji aktualizacji oprogramowania licznika – 1. wariant niepomyślny

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



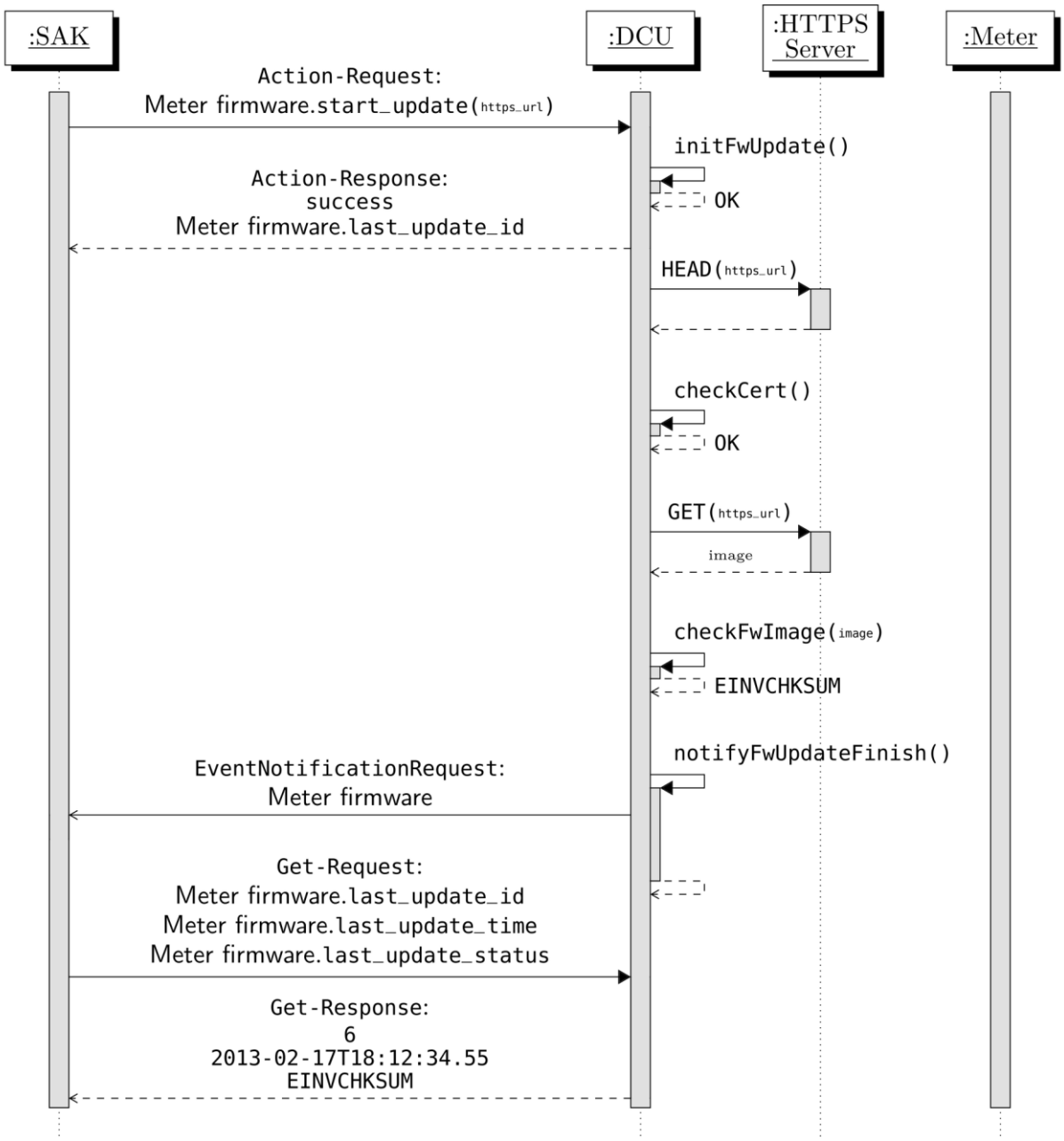
Rys. 25 Diagram sekwencji aktualizacji oprogramowania licznika – 2. wariant niepomyślny

Założono, że podany adres obrazu jest nieprawidłowy.



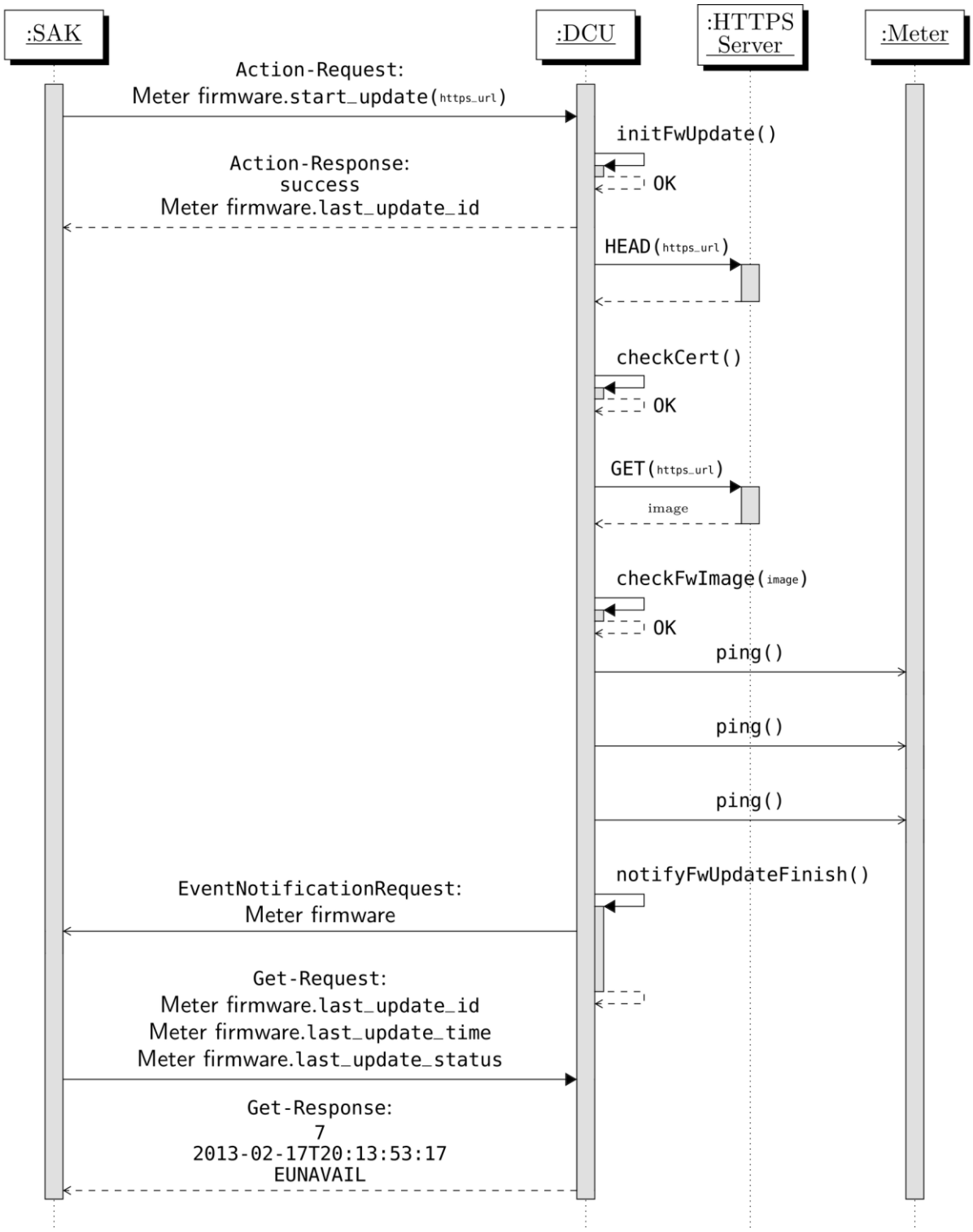
Rys. 26 Diagram sekwencji aktualizacji oprogramowania licznika – 3. wariant niepomyślny

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



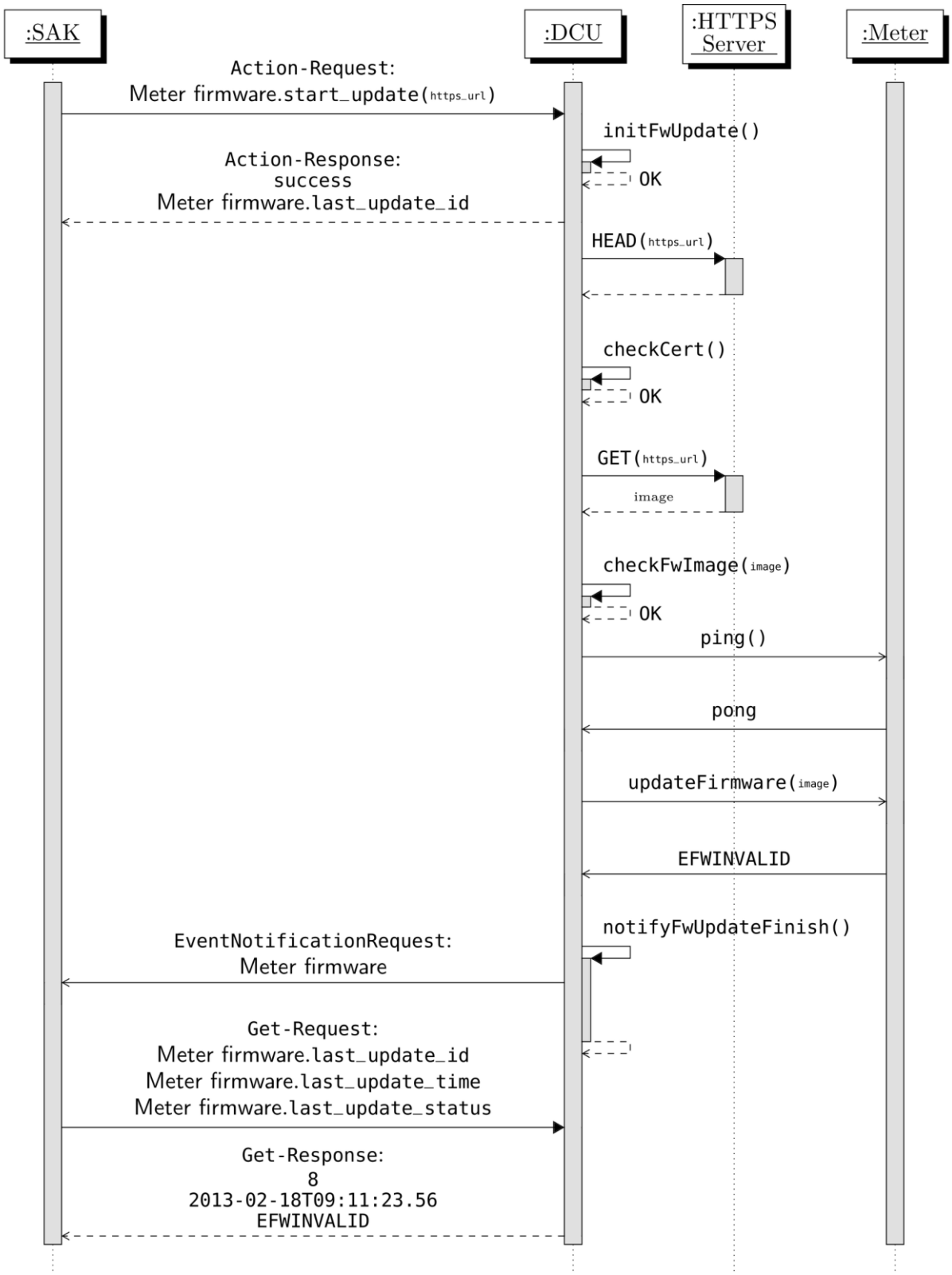
Rys. 27 Diagram sekwencji aktualizacji oprogramowania licznika – 4. wariant niepomyślny

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



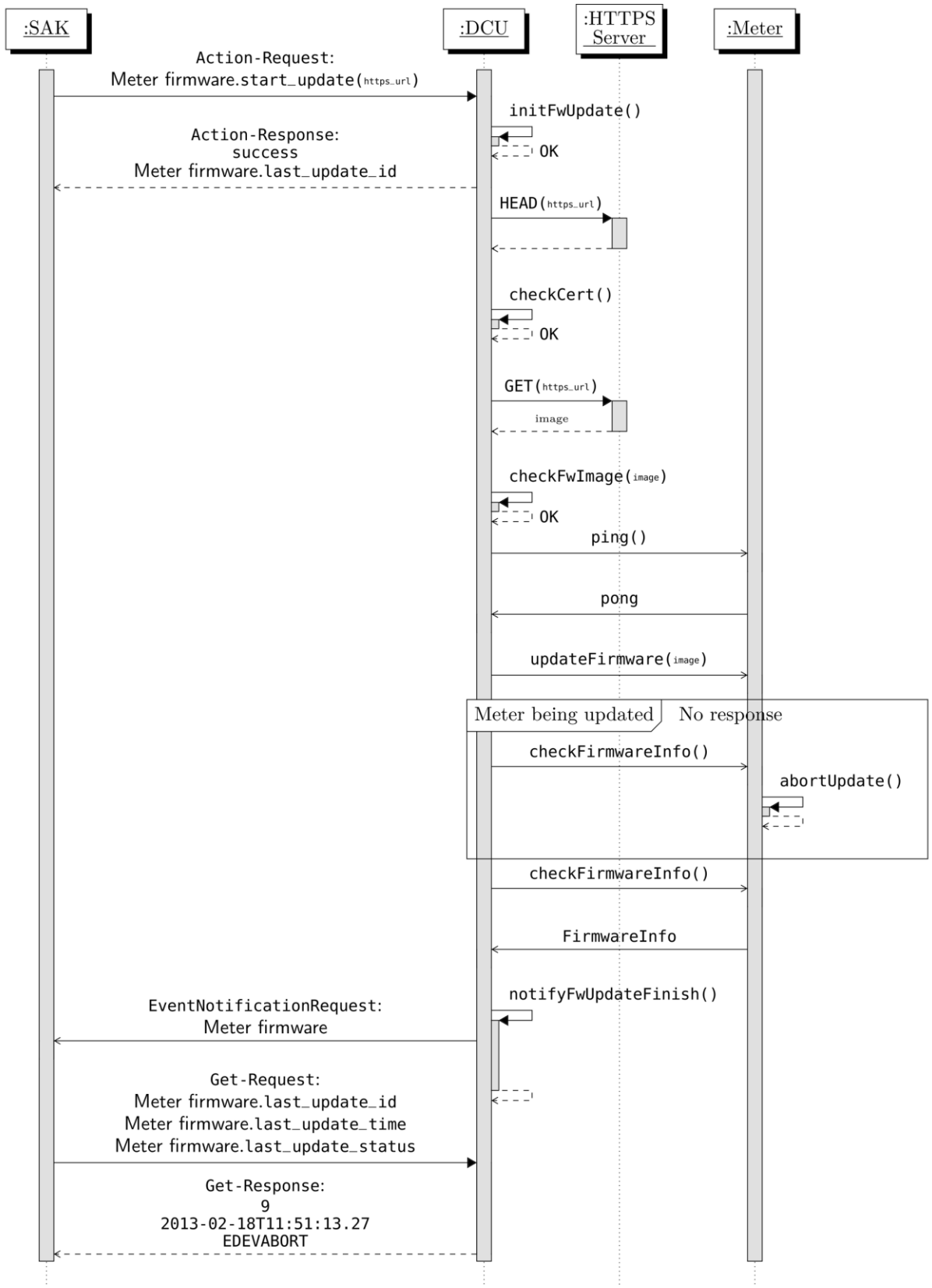
Rys. 28 Diagram sekwencji aktualizacji oprogramowania licznika – 5. wariant niepomyślny

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.

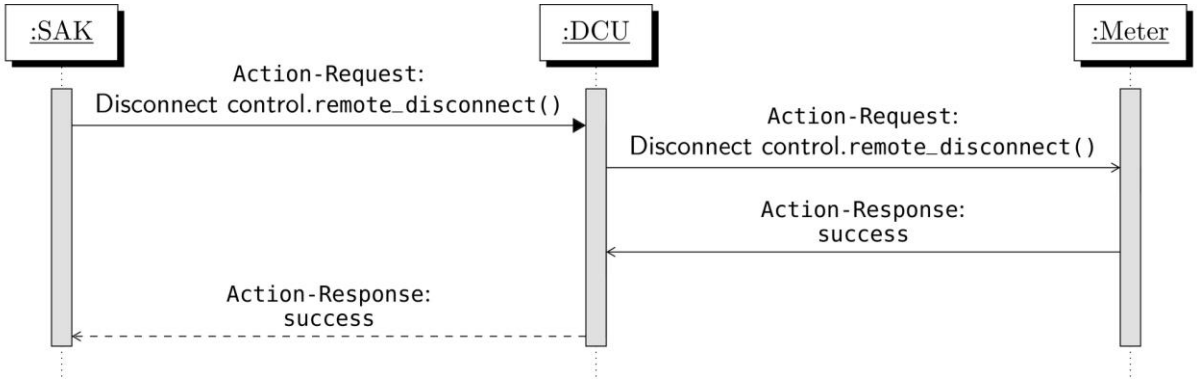


Rys. 30 Diagram sekwencji aktualizacji oprogramowania licznika – 7. wariant niepomyślny

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.



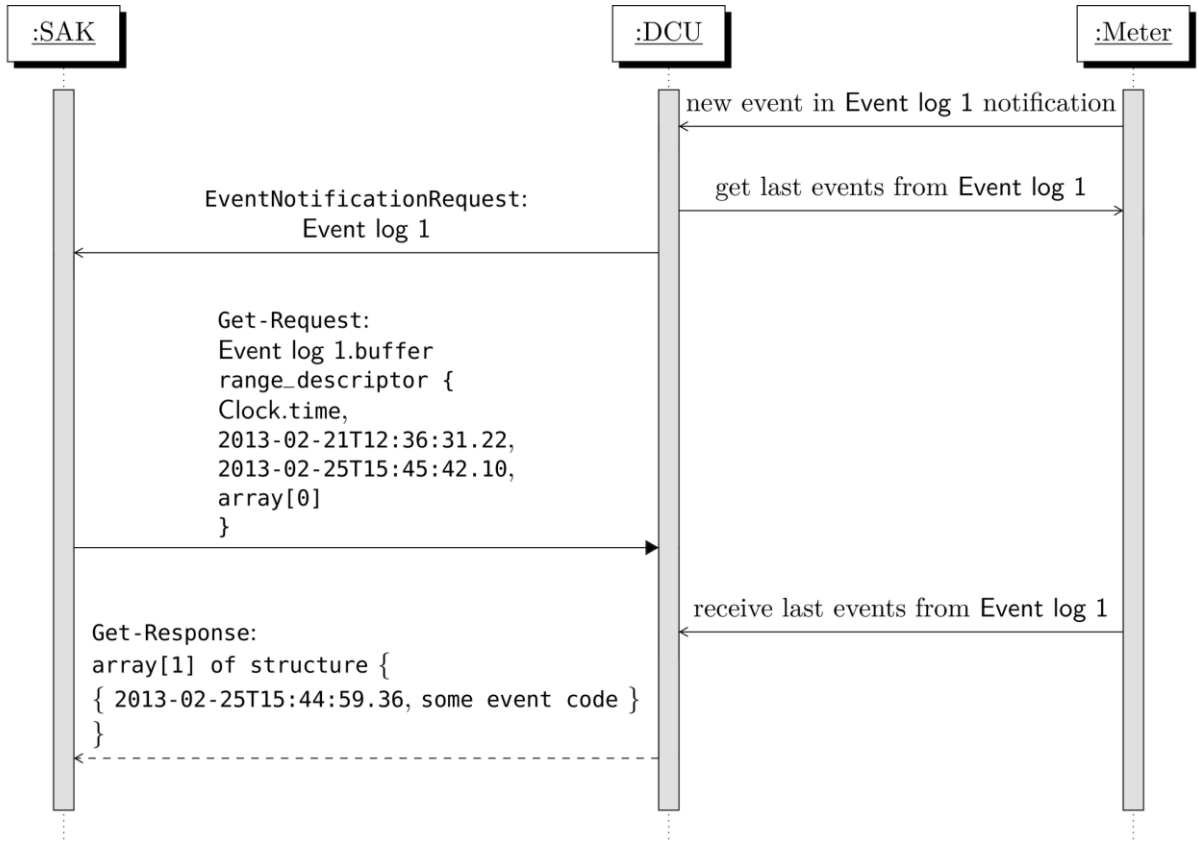
Rys. 31 Diagram sekwencji rozłączenia stycznika w liczniku

Założono, że koncentrator komunikuje się z licznikiem w warstwie aplikacji za pomocą protokołu COSEM.

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

Rekomendowany numerem portu dla protokołu DCSAP jest port numer 4069.

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 SSL z szyfrowaniem AES-CBS i uwierzytelnieniem za pomocą certyfikatu cyfrowego (i algorytmu DSA).

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 Model serwera DCSAP

Ze względu na łatwość testowania i implementacji preferowanym modelem implementacji serwera DCSAP jest serwer równoległy z wykorzystaniem oddzielnych wątków dla każdej sesji. Zaletą serwera równoległego jest możliwość implementacji sekwencyjnej obsługi instrukcji akwizycji.

7.9 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.