

# WebAPI SAOL MUW

SAOL RSWS



## **System Alarmowania i Ostrzegania Ludności**

**Mazowiecki Urząd Wojewódzki w Warszawie**

Plac Bankowy 3/5, 05-077 Warszawa

+48 22 692 64 71

[kjanicki@mazowieckie.pl](mailto:kjanicki@mazowieckie.pl)

[www.mazowieckie.pl](http://www.mazowieckie.pl)

Opracował: inż. Tomasz Marcinkowski

Wersja WebAPI: 3.01

## Spis Treści

<b>1</b>	<b>API SAOL MUW – w pigułce</b> .....	3
<b>2</b>	<b>Komunikacja – podstawowe informacje</b> .....	3
<b>3</b>	<b>Uwierzytelnianie nowego klienta</b> .....	4
3.1	Przekazanie danych do MUW .....	4
3.2	Pierwszy pakiet danych - wysyłka .....	4
3.3	Pierwszy pakiet danych - odbiór .....	5
3.4	Algorytm odbierania danych z serwera .....	6
<b>4</b>	<b>Dodanie nowego obiektu do MUW</b> .....	6
4.1	Uwierzytelnienie połączenia .....	6
4.2	Jednorazowy kod autoryzacyjny połączenia .....	6
4.2.1	Żądanie klienta .....	6
4.2.2	Odpowiedź serwera .....	7
4.3	Przesłanie danych nowego punktu alarmowania .....	7
<b>5</b>	<b>Użytkownicy</b> .....	9
5.1	Wysyłanie żądania przez serwer .....	9
5.2	Odpowiedź klienta .....	9
<b>6</b>	<b>Integracja urządzeń</b> .....	10
6.1	Syrena wirnikowa .....	10
6.1.1	Żądanie danych przez serwer .....	10
6.1.2	Odpowiedź i wysyłanie danych przez klienta .....	10
6.2	Syrena elektroniczna .....	13
6.2.1	Żądanie danych przez serwer .....	13
6.2.2	Odpowiedź i wysyłanie danych przez klienta .....	13
6.3	Stacja pogodowa .....	16
6.4	Czujnik skażeń .....	16
6.5	Zegar DCF .....	16
6.6	Limnimetry .....	16
6.7	Centrale .....	16
6.8	Wszystkie urządzenia w jednej paczce danych .....	16
6.9	Pozostałe urządzenia .....	17
<b>7</b>	<b>Zdarzenia systemowe</b> .....	17
7.1	Syrena wirnikowa .....	17
7.2	Syrena elektroniczna .....	17
7.3	Stacja pogodowa .....	18

7.4	Czujnik skażeń.....	18
7.5	Limnimetr.....	19
7.6	Alarmy .....	19
7.6.1	Alarm na pojedynczej syrenie.....	19
7.6.2	Fizyczne uruchomienie alarmu z przycisku w syrenie.....	20
7.6.3	Potwierdzenie włączenia alarmu w syrenie .....	20
7.7	Komunikaty .....	20
7.7.1	Komunikat na pojedynczej syrenie.....	21
7.7.2	Fizyczne uruchomienie komunikatu z przycisku w syrenie.....	21
7.7.3	Potwierdzenie włączenia komunikatu w syrenie .....	21
7.8	Inne zdarzenia gdzie indziej nie sklasyfikowane .....	21
<b>8</b>	<b>Zarządzanie systemami podrzędnymi .....</b>	<b>22</b>
8.1	Uruchomienie alarmów z MUW na wszystkich syrenach w powiecie/mieście .....	22
8.2	Uruchomienie alarmów z MUW na wybranych syrenach w powiecie/mieście .....	22
8.3	Uruchomienie komunikatów głosowych z MUW na wszystkich syrenach w powiecie/mieście .....	23
8.4	Uruchomienie komunikatów głosowych z MUW na wybranych syrenach w powiecie/mieście.....	23
8.5	Ogłaszanie komunikatów głosowych nadawanych przez mikrofon z centrali MUW na wszystkich syrenach w powiecie/mieście .....	24
8.6	Ogłaszanie komunikatów głosowych nadawanych przez mikrofon z centrali MUW na wybranych syrenach w powiecie/mieście .....	24
8.7	Zatrzymanie ogłaszania alarmów i komunikatów alarmowych .....	25
8.7.1	Zatrzymanie alarmów z MUW na wszystkich syrenach w powiecie/mieście .....	25
8.7.2	Zatrzymanie alarmów z MUW na wybranych syrenach w powiecie/mieście.....	25
8.7.3	Zatrzymanie komunikatów głosowych z MUW na wszystkich syrenach w powiecie/mieście .....	26
8.7.4	Zatrzymanie komunikatów głosowych z MUW na wybranych syrenach w powiecie/mieście .....	26
8.8	Kasowanie pamięci alarmów.....	27
8.8.1	Serwer -> klient.....	27
8.8.2	Klient -> serwer.....	28
<b>9</b>	<b>Bezpieczeństwo odbierania powiadomień URLC .....</b>	<b>29</b>

## 1 API SAOL MUW – w pigułce

WebAPI SAOL MUW jest narzędziem pozwalającym na komunikację centrali wojewódzkiej z systemami obcymi. API systemowe pozwala na wymianę asynchroniczną i synchroniczną pomiędzy SAOL MUW a systemami obcymi. W dokumencie znajduje się zbiór komend oraz sposób komunikacji w jaki należy komunikować się z SAOL MUW. Zawarta w dokumencie specyfikacja pozwala na pełną integrację systemów obcych z SAOL MUW.

## 2 Komunikacja – podstawowe informacje

WebAPI SAOL MUW widnieje pod publicznym adresem IP. W dalszej części dokumentu będziemy posługiwać się skrótem „IP” , które będzie równe zapisowi „[http://adres\\_ip\\_muw](http://adres_ip_muw)” lub „[https://adres\\_ip\\_muw](https://adres_ip_muw)”. Wszystkie funkcje API zostały stworzone w kontrolerze o nazwie `eth_api_inne.php` , do którego należy odwoływać się w następujący sposób:

```
IP/eth_api_inne/nazwa_wykonywanej_funkcji
```

WebAPI SAOL MUW to aplikacja serwerowa, w dalszej części dokumentu SAOL MUW będzie określone jako „*serwer*”, a system obcy jako „*klient*”.

Wyróżniamy dwa sposoby uwierzytelnienia klienta:

- bez autoryzacji
- z autoryzacją

Dane do serwera przekazywane są za pomocą metody POST(ulrc) na wskazany adres IP, kontroler i funkcję.

TCP jest protokołem działającym w trybie klient-serwer. Serwer oczekuje na nawiązanie połączenia na określonym porcie. Klient inicjuje połączenie do serwera. W protokole TCP do nawiązania połączenia pomiędzy dwoma hostami wykorzystywana jest procedura nazwana *three-way handshake*. W sytuacji normalnej jest ona rozpoczynana, gdy host A chce nawiązać połączenie z hostem B, procedura wygląda następująco:

- host A wysyła do hosta B segment SYN wraz z informacją o dolnej wartości numerów sekwencyjnych używanych do numerowania segmentów wysyłanych przez host A (np. 100) a następnie przechodzi w stan SYN-SENT,
- host B, po otrzymaniu segmentu SYN, przechodzi w stan SYN-RECEIVED i, jeżeli również chce nawiązać połączenie, wysyła hostowi A segment SYN z informacją o dolnej wartości numerów sekwencyjnych używanych do numerowania segmentów wysyłanych przez host B (np. 300) oraz segment ACK z polem numeru sekwencji ustawionym na wartość o jeden większą niż wartość pola sekwencji pierwszego segmentu SYN hosta A, czyli 101.
- host A, po odebraniu segmentów SYN i ACK od hosta B przechodzi w stan ESTABLISHED i wysyła do niego segment ACK potwierdzający odebranie segmentu SYN (numer sekwencji ustawiony na 301)
- host B odbiera segment ACK i przechodzi w stan ESTABLISHED
- host A może teraz rozpocząć przesyłanie danych

Używany format daty to czas unixowy, czas POSIX – system reprezentacji czasu mierzący liczbę sekund od 1970 roku UTC, czyli od chwili zwanej początkiem epoki Uniksa (ang. *Unix Epoch*). Nie uwzględnia sekund przestępnych, zatem rzeczywista liczba sekund jakie upłynęły od początku epoki Uniksa jest większa o liczbę sekund przestępnych. W systemie

operacyjnym Unix i pochodnych czas jest przedstawiany jako 32-bitowa liczba sekund, które upłynęły od 1 stycznia 1970. Daną tę interpretuje się jako liczbę ze znakiem (ang. *signed integer*), w której wartości ujemne nie są wykorzystywane, dlatego dostępny przedział czasu wynosi  $2^{31}-1$  sekund, co daje wartość równą 2 147 483 647. Pierwsze 10<sup>9</sup> sekund od początku epoki Uniksa upłynęło 9 września 2001, godz. 01:46:40 GMT. Chwilę tę nazwano "Unix billennium". Systemy uniksowe były odporne na tzw. problem roku 2000: 32-bitowy Unix time wyczerpie się 19 stycznia 2038 o godz. 03:14:07 UTC – wtedy pojawi się problem roku 2038. Obecnie trwają prace, które mają wyeliminować problem roku 2038. Niektóre strony internetowe umożliwiają śledzenie czasu uniksowego na bieżąco, zaś w aplikacji mobilnej Google Play można ustawić czas uniksowy na stronę główną. W dalszej części dokumentu czas w formacie uniksowym będzie określany jako „date”.

**Wszystkie dane pół wysyłki(ang. Post fields) w całym WebAPI SAOL MUW są jednostkowo(każde z osobna) szyfrowane metodą base64 przed wysyłką. Wyjątkami są:**

- pole/klucz „session” nazywany w dokumencie „session\_id”
- pole/klucz PIN
- pole/klucz „check” – suma kontrolna

### 3 Uwierzytelnianie nowego klienta

#### 3.1 Przekazanie danych do MUW

Pierwszym etap dodawania nowego klienta do serwera jest przekazanie adresu IP klienta do obsługi systemu SAOL MUW. Obsługa SAOL MUW wprowadza adres IP klienta do firewall'a, aby klient uzyskał możliwość jakiegokolwiek komunikacji z serwerem.

Obsługa SAOL MUW wygeneruje dla dedykowanego adresu IP klienta kod uwierzytelniający, który będzie niezbędny do prawidłowej komunikacji z SAOL MUW i przekaże go obsłudze klienta.

Do komunikacji między centralami niezbędny jest również klucz licencyjny, który klient otrzymuje od MUW lub wykupuje klucz licencyjny u producenta lub dystrybutora systemu SAOL MUW i przekazuje obsłudze MUW.

#### 3.2 Pierwszy pakiet danych - wysyłka

Po otrzymaniu kodu uwierzytelniającego należy potwierdzić tączność wysyłając odpowiednią paczkę danych na adres:

```
IP/eth_api_inne/potwierdzam_uwierzytelnienie_klient
```

Należy przestać dane zakodowane poprzez algorytm szyfrujący.

**\*\***Nazwa pola wysyłki – „session” -> dane: wygenerowany losowo unikalny identyfikator komendy składający się z alfa numerycznych znaków(wyłączając polskie znaki dialektyczne) o stałej długości 13 znaków.

Nazwa pola wysyłki - „date\_send” -> dane: date (zakodowane logarytmem szyfrującym base64)

Nazwa pola wysyłki – „PIN” -> dane -> string utworzony z danych:

**\***date\_send(kodowane base64), **\*\*\*** kod uwierzytelniający(string odwrócony – pisany od tyłu oraz kodowany base64),session

Podany string należy zakodować przed wysyłką metodą szyfrującą MD5 i następnie metodą szyfrującą SHA1.

Nazwa pola wysyłki – „check” -> dane: string składający się z pól:

*\*session,date\_send,PIN*

Podany string należy zakodować przed wysyłką metodą szyfrującą SHA1 i następnie metodą szyfrującą MD5.

\*String pisany jest w sposób ciągły, nie występują znaki interpunkcyjne pomiędzy seriami danymi, należy zachować kolejność przekazywanych danych oraz wielkości znaków

\*\* W dalszej części dokumentu string składający się z alfa numerycznych znaków (wyłączając polskie znaki dialektyczne) będzie nazywany „session\_id”

\*\*\* kod uwierzytelniający (string odwrócony – pisany od tyłu oraz kodowany base64) w dalszej części dokumentu będzie nazywany „OB\_pin”

### 3.3 Pierwszy pakiet danych - odbiór

Po poprawnym odebraniu danych przez serwer zostaną zwrócone dane na adres IP klienta:  
Adres odpowiedzi:

IP\_klient/eth\_api/uwierzytelnienie

Nazwa pola odbioru – „session\_id” -> dane: wygenerowany losowo unikalny identyfikator komendy składający się z alfa numerycznych znaków (wyłączając polskie znaki dialektyczne) o stałej długości 13 znaków.

Nazwa pola odbioru – „date\_get” -> dane: date – data do synchronizacji czasu - serwer -> klient, klient -> serwer

Nazwa pola odbioru – „PIN” -> dane: string utworzony z danych:

\* klucz\_licencyjny,session\_id ,date(kodowane base64), kod uwierzytelniający(string odwrócony – pisany od tyłu oraz kodowany base64)

Podany string jest zakodowany przed wysyłką metodą szyfrującą MD5 i następnie metodą szyfrującą SHA1.

Nazwa pola odbioru – „check” -> dane -> string składający się z pól:

*\* session\_id,date\_send,PIN*

Podany string jest zakodowany przed wysyłką metodą szyfrującą SHA1 i następnie metodą szyfrującą MD5.

\*String pisany jest w sposób ciągły, nie występują znaki interpunkcyjne pomiędzy seriami danymi

**Nazwa pola wysyłki lub nazwa pola odbioru w dalszej części dokumenty będzie nazywany „kluczem”.**

### 3.4 Algorytm odbierania danych z serwera

Algorytm klienta odbierający dane z serwera musi sprawdzać poprawność otrzymywanych danych. Po stronie klienta należy sprawdzić:

1. Adres IP nadawcy - jeśli inny od adresu IP serwera dane powinny zostać natychmiast odrzucone, a adres IP, który próbował się komunikować zapisany do rejestru i wyświetlony obsłudze oprogramowaniu klienta.
2. Przyrównać wartość PINu odebranego, z PINem wygenerowanym z danych odebranych w pozostałych paczkach danych oraz z danymi zapisanymi tj. kod uwierzytelniający.
3. Sprawdzenie sumy kontrolnej, czy paczka danych nie uległa deformacji podczas przesyłania.

## 4 Dodanie nowego obiektu do MUW

Dodawanie nowego obiektu oraz nowych użytkowników do MUW wykorzystując metodę podwójnego uwierzytelnienia. Są to opcje wymagające dodatkowego bezpieczeństwa przesyłanych danych.

### 4.1 Uwierzytelnienie połączenia

W pierwszym etapie dodawania nowego obiektu do MUW, klient zwraca się z żądaniem do serwera o kod autoryzacyjny połączenia. Serwer generuje jednorazowy kod autoryzacyjny składający się z 13 znaków alfa numerycznych (z wyłączeniem polskich znaków diakrytycznych) i otwiera bramę połączenia na 5 sekund. Po upływie 5 sekund brama autoryzacyjna zostaje zamknięta, a kod autoryzacyjny traci ważność.

### 4.2 Jednorazowy kod autoryzacyjny połączenia

#### 4.2.1 Żądanie klienta

Klient wysyła żądanie na adres:

```
IP/eth_api_inne/autoryzacja_polaczenia
```

Należy przestać dane zakodowane poprzez algorytm szyfrujący.

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: string „AUTORYZACJA” (z zachowaniem wielkości znaków)

\*Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,zadanie,OB\_pin,date*

\*Podany string należy zakodować przed wysyłką metodą szyfrującą MD5 i następnie metodą szyfrującą SHA1, w dalszej części dokumentu PIN będzie zawsze kodowany tą metodą. String pisany jest w sposób ciągły, nie występują znaki interpunkcyjne pomiędzy seriami danymi, należy zachować kolejność przekazywanych danych oraz wielkości znaków.

\*\*Klucz – „check” -> dane -> string utworzony z danych:

*session\_id,date,zadanie,PIN*



**\*\*** Dane w polu „check” należy zakodować przed wysyłką metodą szyfrującą SHA1 i następnie metodą szyfrującą MD5, w dalszej części dokumentu klucz „check” będzie zawsze kodowany tą metodą. Na dane pola „check” składają się wszystkie klucze przesyłane w danym pakiecie danych.

String pisany jest w sposób ciągły, nie występują znaki interpunkcyjne pomiędzy seriami danymi, należy zachować kolejność przekazywanych danych oraz wielkości znaków.

#### 4.2.2 Odpowiedź serwera

Serwer po odebraniu prawidłowej paczki danych odpowiada na żądanie klienta danymi:

Adres odpowiedzi:

```
IP_klient/eth_api/autoryzacja_polaczenia_good
```

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „pass” -> dane: string odwrócony (pisany od tyłu np. qwerty1234567 -> 7654321ytrewq) oraz kodowany base64, składający się z 13 znaków alfa numerycznych (z wyłączeniem polskich znaków diakrytycznych)

Klucz – „PIN” -> dane -> string utworzony z danych: *klucz\_licencyjny, session\_id, pass, OB\_pin, date*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id, date, pass, PIN*

Przesłane przez serwer dane zawarte w kluczu „pass” należy zapisać i wysłać w kolejnym żądaniu do serwera.

#### 4.3 Przesłanie danych nowego punktu alarmowania

Przesłany jednorazowy kod autoryzacyjny połączenie z klucza „pass”, jest ważny 5 sekund, należy w tym czasie wysłać dane przyłączenia obiektu do SAOL MUW.

Do przesłania danych należy utworzyć tablicę asocjacyjną:

Klucz – „session\_id” -> dane: session\_id

Klucz – „centrala\_identity” -> dane z systemu obcego: identyfikator przyłączanej centrali składający się z 13 znaków alfa numerycznych (z wyłączeniem polskich znaków diakrytycznych)

Klucz – „obiekt\_oddzial” -> dane z systemu obcego: string do 100 znaków

Klucz – „obiekt\_obiekt” -> dane z systemu obcego: string do 500 znaków

Klucz – „obiekt\_ulica” -> dane z systemu obcego: string do 45 znaków

Klucz – „obiekt\_miasto” -> dane z systemu obcego: string do 35 znaków

Klucz – „obiekt\_kod” -> dane z systemu obcego: string do 10 znaków

Klucz – „obiekt\_woj” -> dane z systemu obcego: string do 50 znaków

Klucz – „obiekt\_kraj” -> dane z systemu obcego: string do 30 znaków

Klucz – „obiekt\_faks” -> dane z systemu obcego: string do 32 znaków

Klucz – „obiekt\_mail” -> dane z systemu obcego: string do 50 znaków

Klucz – „obiekt\_www” -> dane z systemu obcego: string do 50 znaków

Klucz – „obiekt\_nazwa” -> dane z systemu obcego: string do 100 znaków

Klucz – „obiekt\_tel” -> dane z systemu obcego: string do 50 znaków

Klucz – „obiekt\_typ” -> dane z systemu obcego: string do 150 znaków

Klucz – „obiekt\_ip” -> dane z systemu obcego: string do 20 znaków



Klucz – „obiekt\_adres\_radiowy” -> dane z systemu obcego: string do 4 znaków  
Klucz – „podstrona” -> dane z systemu obcego: według istniejących unikalnych identyfikatorów powiatów/miast na prawach powiatu(załącznik 4)  
Klucz – „miasto\_obszar” -> dane z systemu obcego: string do 50 znaków  
Klucz – „miasto\_obszar\_identity” -> dane z systemu obcego: identyfikator nowo powstałego obszaru/miasta składający się z 13 znaków alfa numerycznych(z wyłączeniem polskich znaków dialektycznych)  
Klucz – „województwo” -> ID województwa : załącznik 4 do WebAPI MUW SAOL  
Klucz – „kod\_wersji” -> dane z systemu obcego: kod wersji to numer porządkowy wysyłanych danych, rozpoczynający się od 1 i każdorazowo rosnący o 1 int(255).

Dane które należy wysłać na adres:

IP/eth\_api\_inne/obiekt\_eth\_insert

Klucz – „session\_id” -> dane: session\_id  
Klucz – „date” -> dane: date  
Klucz – „data\_send” -> dane: Dane z tablicy asocjacyjnej należy zamienić na stringa (funkcja PHP serialize())  
Klucz – „pass” -> dane: jednorazowy kod uwierzytelniający przesyłany na żądanie klienta od serwera, pisany od przodu(odebrany: 7654321ytrewq, wysłać: qwerty1234567)  
Klucz – „PIN” -> dane -> string utworzony z danych:  
*klucz\_licencyjny,session\_id,data\_send,pass,OB\_pin,date*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,data\_send,pass,PIN*

Serwer odpowie danymi w przypadku prawidłowego odbioru paczki danych:

Adres odpowiedzi:

IP\_klient/eth\_api/potwierdzenie

Klucz – „session\_id” -> dane: session\_id  
Klucz – „date” -> dane: date  
Klucz – „old\_session” -> dane: kod identyfikacyjny żądanie klienta, na które serwer odpowiada  
Klucz – „kod” -> dane: kod\_wersji  
Klucz – „PIN” -> dane -> string utworzony z danych: *klucz\_licencyjny,session\_id, old\_session, kod,OB\_pin, date*  
Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,old\_session,kod,PIN*

Klient musi zapisać czy odpowiedź na jego żądanie została poprawnie odebrana, w przeciwnym wypadku po stronie klienta leży obowiązek kontroli wersji danych zapisanych w serwerze. Klient powinien w przypadku błędu w żądaniu, ponawiać co jakiś czas żądanie, lub monitorować połączenie z serwerem i w przypadku odzyskania łączności natychmiast wysłać paczkę danych ponownie.

## 5 Użytkownicy

Użytkownicy SAOL MUW mają możliwość logowania się zdalnie do central podrzędnych wykorzystując protokół komunikacyjny http lub https. Logowanie do central podrzędnych odbywa się za pomocą takich samych danych jak w systemie SAOL MUW. Związku z powyższym dodawanie użytkownika w SAOL MUW musi zostać zapisane również w systemach podrzędnych, w zakresie nie mniejszym niż obowiązkowe dane do logowania takie jak login i hasło. Hasło w systemie SAOL MUW jest kodowane metodą SHA1 i przechowywane w bazie danych. SAOL MUW pilnuje kodu wersji w przypadku dodawania i edycji użytkowników.

### 5.1 Wysłanie żądania przez serwer

Adres odbiorcy, na który serwer wysła żądanie:

```
IP_klient/eth_api/user_eth
```

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „identity” -> dane: unikalny identyfikator użytkownika

Klucz – „login” -> dane: login użytkownika string do 30 znaków

Klucz – „pass” -> dane: hasło użytkownika string do 42 znaków

\*Klucz – „kod\_wersji” -> dane: kod wersji

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,identity,login,pass,kod\_wersji,OB\_pin,date*

Klucz – „check” -> dane -> string utworzony z danych:

*session\_id,date,identity,login,pass,kod\_wersji,PIN*

\* kod wersji to numer porządkowy wysyłanych danych, rozpoczynający się od 1 i każdorazowo rosnący o 1, int(255) – nazywany będzie w dalszej części dokumentu „kod\_wersji”. Klient zapisuje dane użytkownika login i hasło, co pozwoli zalogować użytkownika do systemu podrzędnego.

### 5.2 Odpowiedź klienta

Klient musi przeanalizować dane i w przypadku poprawnej paczki danych odesłać odpowiedź.

Odpowiedź na adres:

```
IP/eth_api_inne/user_eth_good
```

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „old\_session” -> dane: kod identyfikacyjny żądanie klienta, na które serwer odpowiada

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,old\_session,OB\_pin,date*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,old\_session,PIN*

Serwer musi zapisać czy odpowiedź na jego żądanie została poprawnie odebrana, w przeciwnym wypadku po stronie serwera leży obowiązek kontroli wersji danych zapisanych

w serwerze. Serwer powinien w przypadku błędu w żądaniu, ponawiać co jakiś czas żądanie, lub monitorować połączenie z klientem i w przypadku odzyskania łączności natychmiast wysłać paczkę danych ponownie.

## 6 Integracja urządzeń

Administratorzy oraz operatorzy systemu SAOL MUW muszą mieć pełną wizualizację central oraz innych urządzeń zainstalowanych w MUW. Muszą znać położenie urządzeń (współrzędne na mapie) oraz obecny status urządzenia, czy jest sprawne, czy nie trwa na nim alarm lub inne. Dodatkowo SAOL MUW ma możliwość na żądanie ściągnięcia poniższych danych.

### 6.1 Syrena wirnikowa

#### 6.1.1 Żądanie danych przez serwer

Adres odbiorcy, na który serwer wysła żądanie:

```
IP_klient/eth_api/get_produkth_eth
```

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „GET\_PRODUKT”

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,zadanie,OB\_pin,date*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,PIN*

#### 6.1.2 Odpowiedź i wysyłanie danych przez klienta

##### 6.1.2.1 Odpowiedź

Klient zwraca w odpowiedzi na żądanie serwera dane:

„session\_id” -> dane: session\_id

„old\_session” -> dane: kod identyfikacyjny żądanie serwera, na które klient odpowiada

„date” -> dane: date

„produkt\_identity” -> dane: string 13 znaków

„produkt\_nazwa” -> dane: string do 70 znaków

„produkt\_symbol” -> dane: string do 10 znaków

„produkt\_adres\_radiowy” -> dane: string do 4 znaków

„produkt\_adres” -> dane: string do 150 znaków

„urządzenie\_typ\_id” -> dane: (według wytycznych – załącznik 2)

„produkt\_opis” -> dane: date

„produkt\_x” -> dane: string do 20 znaków

„produkt\_y” -> dane: string do 20 znaków

„produkt\_azymut” -> dane: integer do 4 znaków

„produkt\_filttr” -> dane: (według wytycznych – załącznik 3)

„produkt\_centrala\_identity” -> dane: identyfikator centrali

„produkt\_active” -> dane: integer 1 znak 0 lub 1-active

„produkt\_data\_dodania” -> dane: date

„produkt\_data\_modyfikacji” -> dane: date

„produkt\_modyfikacja\_przez” -> dane: login użytkownika

„produkt\_ilosc\_wyswietlen” -> dane: integer(255)

„produkt\_icon\_status” -> dane: 0 – nowa, 1 – sprawna, 2 – niesprawna, 3 – brak łączności, 4 – ogłasza komunikat alarm  
„produkt\_tryb\_pracy” -> dane: string N – normalny C – trening  
„produkt\_wykonywane\_zadanie” -> dane: string do 3 znaków (załącznik 6)  
„produkt\_sektor” -> dane: string (zakres sektorów A B C D E F G H),  
„produkt\_data\_testu” -> dane: date  
„produkt\_ipk” -> dane: string do 20 znaków  
„produkt\_data\_nadania” -> dane: date  
„produkt\_grupa” -> dane: pole puste  
„produkt\_typ\_ster” -> dane: string do 255 znaków  
„produkt\_pamiec\_alarmu” -> int 0,1 lub 2, 0 – brak, 1 – pamięć syr.sprawna, 2 – pamięć syr. Z usterką  
„produkt\_data\_instalacji” -> dane: date  
„produkt\_data\_uruchomienia” -> dane: date  
„produkt\_data\_bezp\_kondycji\_aku” -> dane: date  
„produkt\_typ\_akumulatora” -> dane: string do 150 znaków  
„produkt\_kod\_wersji” -> dane: intiger(255)

Powyższe dane muszą zostać rozdzielone separatorem „<par>” . Kolejne urządzenia/produkty rozdzielamy separatorem „<spacja>”. Należy zachować podaną powyżej kolejność przekazywanych danych i połączyć wszystkie dane o urządzeniach w jednego stringa i zwrócić w odpowiedzi na żądanie serwera.

### 6.1.2.2 Wysłanie

System SAOL MUW zbudowany jest z myślą o wielu podrzędnych systemach. Zaimplementowane algorytmy pilnują priorytetów w komunikacji między systemem głównym, a systemami podrzędnymi. Założone jest, iż w przypadku nadawania alarmu w którymkolwiek z podsystemów wszystkie akcje wymiany dużych paczek danych zostają wstrzymane. Z tego powodu korzystamy z opcji podwójnej autoryzacji. Jeśli algorytm serwera zezwoli na transfer danych, zostanie zwrócony kod uwierzytelniający połączenie, w przeciwnym wypadku serwer zwróci komunikat „WAIT.CZAS”.

CZAS to na przykład liczba 150 lub 250 lub inna z zakresu 001 do 999 oznaczająca ile czasu połączenie nie będzie możliwe. Jeśli paczka danych jest błędna serwer nie odpowie.

Klient wysyła żądanie na adres:

IP/eth\_api\_inne/autoryzacja\_polaczenia

Należy przestać dane zakodowane poprzez algorytm szyfrujący.

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: string „AUTO\_PROD” (z zachowaniem wielkości znaków)

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,zadanie,OB\_pin,date*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,PIN*

Po otrzymaniu identyfikatora sesji Klient wysyła żądanie na adres:

IP/eth\_api\_inne/send\_produk

Do przesłania danych należy utworzyć tablicę asocjacyjną:

Klucz – „produkt\_identity” -> dane: string 13 znaków  
Klucz – „produkt\_nazwa” -> dane: string do 70 znaków  
Klucz – „produkt\_symbol” -> dane: string do 10 znaków  
Klucz – „produkt\_adres\_radiowy” -> dane: string do 4 znaków  
Klucz – „produkt\_adres” -> dane: string do 150 znaków  
Klucz – „urządzenie\_typ\_id” -> dane: (według wytycznych - załącznik 2)  
Klucz – „produkt\_opis” -> dane: date  
Klucz – „produkt\_x” -> dane: string do 20 znaków  
Klucz – „produkt\_y” -> dane: string do 20 znaków  
Klucz – „produkt\_azymut” -> dane: intiger do 4 znaków  
Klucz – „produkt\_filtr” -> dane: (według wytycznych - załącznik 3)  
Klucz – „produkt\_centrala\_identity” -> dane: identyfikator centrali  
Klucz – „produkt\_active” -> dane: intiger 1 znak 0 lub 1-active  
Klucz – „produkt\_data\_dodania” -> dane: date  
Klucz – „produkt\_data\_modyfikacji” -> dane: date  
Klucz – „produkt\_modyfikacja\_przez” -> dane: login użytkownika  
Klucz – „produkt\_ilosc\_wyswietlen” -> dane: intiger(255)  
Klucz – „produkt\_icon\_status” -> dane: 0 – nowa, 1 – sprawna, 2 – niesprawna, 3 – brak łączności, 4 – ogłasza komunikat alarm  
Klucz – „produkt\_tryb\_pracy” -> dane: string N – normalny C - trening  
Klucz – „produkt\_wykonywane\_zadanie” -> dane: string do 3 znaków (załącznik 6)  
Klucz – „produkt\_sektor” -> dane: string (zakres sektorów A B C D E F G H),  
Klucz – „produkt\_data\_testu” -> dane: date  
Klucz – „produkt\_ipk” -> dane: string do 20 znaków  
Klucz – „produkt\_data\_nadania” -> dane: date  
Klucz – „produkt\_grupa” -> dane: pole puste  
Klucz – „produkt\_typ\_ster” -> dane: string do 255 znaków  
Klucz – „produkt\_pamiec\_alarmu” -> int 0,1 lub 2, 0 – brak, 1 – pamięć syr.sprawna, 2 – pamięć syr. z usterką  
Klucz – „produkt\_data\_instalacji” -> dane: date  
Klucz – „produkt\_data\_uruchomienia” -> dane: date  
Klucz – „produkt\_data\_bezp\_kondycji\_aku” -> dane: date  
Klucz – „produkt\_typ\_akumulatora” -> dane: string do 150 znaków  
Klucz – „produkt\_kod\_wersji” -> dane: intiger(255)

Dane urządzenia zapisujemy w drugą tablicę, dwuwymiarową asocjacyjną:

Klucz 1 (węzeł) -> identyfikator\_urządzenia -> Klucz 2 (węzeł) -> produkt\_kod\_wersji -> dane: serializowane dane o urządzeniu (php serialize())

Przykład:

```
$tablica['fhsjkadur4k2k'][3]= 'a:2:{s:3:"produkt_nazwa";s:5:"Syrena";s:4:"produkt_symbol";s:6:"SW-01"(...)};'
```

Wszystkie dane z drugiej tablicy ponownie serializujemy i przekazujemy do zmiennej o kluczu „data\_send”.

Klucz – „session\_id” -> dane: session\_id

Klucz – „old\_session” -> dane: kod identyfikacyjny żądanie serwera, na które klient odpowiada

Klucz – „date” -> dane: date

Klucz – „data\_send” -> dane: Dane z drugiej tablicy asocjacyjnej (serialize())

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,old\_session,date,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych:

*session\_id,old\_session,date,data\_send,PIN*

## 6.2 Syrena elektroniczna

### 6.2.1 Żądanie danych przez serwer

Adres odbiorcy, na który serwer wysyła żądanie:

IP\_klient/eth\_api/get\_produk\_t\_eth

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „GET\_PRODUKT”

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,OB\_pin,date*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,PIN*

### 6.2.2 Odpowiedź i wysyłanie danych przez klienta

#### 6.2.2.1 Odpowiedź

Klient zwraca w odpowiedzi na żądanie serwera dane:

„session\_id” -> dane: session\_id

„old\_session” -> dane: kod identyfikacyjny żądanie serwera, na które klient odpowiada

„date” -> dane: date

„produkt\_identity” -> dane: string 13 znaków

„produkt\_nazwa” -> dane: string do 70 znaków

„produkt\_symbol” -> dane: string do 10 znaków

„produkt\_adres\_radiowy” -> dane: string do 4 znaków

„produkt\_adres” -> dane: string do 150 znaków

„urządzenie\_typ\_id” -> dane: (według wytycznych - załącznik 2)

„produkt\_opis” -> dane: date

„produkt\_x” -> dane: string do 20 znaków

„produkt\_y” -> dane: string do 20 znaków

„produkt\_azymut” -> dane: intiger do 4 znaków

„produkt\_filttr” -> dane: (według wytycznych - załącznik 3)

„produkt\_centrala\_identity” -> dane: identyfikator centrali

„produkt\_active” -> dane: intiger 1 znak 0 lub 1-active

„produkt\_data\_dodania” -> dane: date

„produkt\_data\_modyfikacji” -> dane: date

„produkt\_modyfikacja\_przez” -> dane: login użytkownika

„produkt\_ilosc\_wyswietlen” -> dane: intiger(255)  
 „produkt\_icon\_status” -> dane: 0 – nowa, 1 – sprawna, 2 – niesprawna, 3 – brak łączności, 4 – ogłasza komunikat alarm  
 „produkt\_tryb\_pracy” -> dane: string N – normalny C - trening  
 „produkt\_wykonywane\_zadanie” -> dane: string do 3 znaków (załącznik 6)  
 „produkt\_sektor” -> dane: string (zakres sektorów A B C D E F G H),  
 „produkt\_data\_testu” -> dane: date  
 „produkt\_ipk” -> dane: string do 20 znaków  
 „produkt\_data\_nadania” -> dane: date  
 „produkt\_grupa” -> dane: pole puste  
 „produkt\_typ\_ster” -> dane: string do 255 znaków  
 Klucz – „produkt\_pamiec\_alarmu” -> int 0,1 lub 2, 0 – brak, 1 – pamięć syr.sprawna, 2 – pamięć syr. z usterką  
 „produkt\_data\_instalacji” -> dane: date  
 „produkt\_data\_uruchomienia” -> dane: date  
 „produkt\_data\_bezp\_kondycji\_aku” -> dane: date  
 „produkt\_typ\_akumulatora” -> dane: string do 150 znaków  
 „produkt\_kod\_wersji” -> dane: intiger(255)

Powyższe dane muszą zostać rozdzielone separatorem „<par>” . Kolejne urządzenia/produkty rozdzielamy separatorem „<spacja>”. Należy zachować podaną powyżej kolejność przekazywanych danych i połączyć wszystkie dane o urządzeniach w jednego stringa i zwrócić w odpowiedzi na żądanie serwera.

### 6.2.2.2 Wysłanie

System SAOL MUW zbudowany jest z myślą o wielu podrzędnych systemach.

Zaimplementowane algorytmy pilnują priorytetów w komunikacji między systemem głównym, a systemami podrzędnymi. Założone jest, iż w przypadku nadawania alarmu w którymkolwiek z podsystemów wszystkie akcje wymiany dużych paczek danych zostają wstrzymane. Z tego powodu korzystamy z opcji podwójnej autoryzacji. Jeśli algorytm serwera zezwoli na transfer danych, zostanie zwrócony kod uwierzytelniający połączenie, w przeciwnym wypadku serwer zwróci komunikat „WAIT.CZAS”.

CZAS to na przykład liczba 150 lub 250 lub inna z zakresu 001 do 999 oznaczająca ile czasu połączenie nie będzie możliwe. Jeśli paczka danych jest błędna serwer nie odpowie.

Klient wysyła żądanie na adres:

IP/eth\_api\_inne/autoryzacja\_polaczenia

Należy przestać dane zakodowane poprzez algorytm szyfrujący.

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: string „AUTORYZACJA” (z zachowaniem wielkości znaków)

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,zadanie,OB\_pin,date*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,PIN*



Po otrzymaniu identyfikatora sesji :  
Klient wysyła żądanie na adres:

IP/eth\_api\_inne/send\_produk

Do przesłania danych należy utworzyć tablicę asocjacyjną:

Klucz – „produkt\_identity” -> dane: string 13 znaków  
Klucz – „produkt\_nazwa” -> dane: string do 70 znaków  
Klucz – „produkt\_symbol” -> dane: string do 10 znaków  
Klucz – „produkt\_adres\_radiowy” -> dane: string do 4 znaków  
Klucz – „produkt\_adres” -> dane: string do 150 znaków  
Klucz – „urządzenie\_typ\_id” -> dane: (według wytycznych - załącznik 2)  
Klucz – „produkt\_opis” -> dane: date  
Klucz – „produkt\_x” -> dane: string do 20 znaków  
Klucz – „produkt\_y” -> dane: string do 20 znaków  
Klucz – „produkt\_azymut” -> dane: integer do 4 znaków  
Klucz – „produkt\_filtr” -> dane: (według wytycznych - załącznik 3)  
Klucz – „produkt\_centrala\_identity” -> dane: identyfikator centrali  
Klucz – „produkt\_active” -> dane: integer 1 znak 0 lub 1-active  
Klucz – „produkt\_data\_dodania” -> dane: date  
Klucz – „produkt\_data\_modyfikacji” -> dane: date  
Klucz – „produkt\_modyfikacja\_przez” -> dane: login użytkownika  
Klucz – „produkt\_ilosc\_wyswietlen” -> dane: integer(255)  
Klucz – „produkt\_icon\_status” -> dane: 0 – nowa, 1 – sprawna, 2 – niesprawna, 3 – brak łączności, 4 – ogłasza komunikat alarm  
Klucz – „produkt\_tryb\_pracy” -> dane: string N – normalny C - trening  
Klucz – „produkt\_wykonywane\_zadanie” -> dane: string do 3 znaków (załącznik 6)  
Klucz – „produkt\_sektor” -> dane: string (zakres sektorów A B C D E F G H),  
Klucz – „produkt\_data\_testu” -> dane: date  
Klucz – „produkt\_ipk” -> dane: string do 20 znaków  
Klucz – „produkt\_data\_nadania” -> dane: date  
Klucz – „produkt\_grupa” -> dane: pole puste  
Klucz – „produkt\_typ\_ster” -> dane: string do 255 znaków  
Klucz – „produkt\_pamiec\_alarmu” -> int 0,1 lub 2, 0 – brak, 1 – pamięć syr.sprawna, 2 – pamięć syr. z usterką  
Klucz – „produkt\_data\_instalacji” -> dane: date  
Klucz – „produkt\_data\_uruchomienia” -> dane: date  
Klucz – „produkt\_data\_bezp\_kondycji\_aku” -> dane: date  
Klucz – „produkt\_typ\_akumulatora” -> dane: string do 150 znaków  
Klucz – „produkt\_kod\_wersji” -> dane: integer(255)

Dane urządzenia zapisujemy w drugą tablicę, dwuwymiarową asocjacyjną:

Klucz 1 -> identyfikator\_urządzenia -> Klucz 2 -> produkt\_kod\_wersji -> dane: serializowane dane o urządzeniu (php serialize())

Przykład:

```
$tablica['fhsjkadur4k2k'][3]= 'a:2:{s:3:"produkt_nazwa";s:5:"Syrena";s:4:"produkt_symbol";s:6:"SW-01"(...)};'
```

Wszystkie dane z drugiej tablicy ponownie serializujemy i przekazujemy do zmiennej o kluczu „data\_send”.

Klucz – „session\_id” -> dane: session\_id

Klucz – „old\_session” -> dane: kod identyfikacyjny żądanie serwera, na które klient odpowiada

Klucz – „date” -> dane: date

Klucz – „data\_send” -> dane: Dane z drugiej tablicy asocjacyjnej (serialize())

Klucz – „kod\_wersji” -> dane: kod wersji

Klucz – „PIN” -> dane -> string utworzony z danych: *klucz\_licencyjny,session\_id,old\_session,date,data\_send,kod\_wersji,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,old\_session,date,data\_send,kod\_wersji,PIN*

### 6.3 Stacja pogodowa

Integracja urządzenia typu „stacja pogodowa” jest analogiczna do 6.1 Syrena wirnikowa lub 6.2 Syrena elektroniczna ,wyłączając klucze nie dotyczące tak jak między innymi „produkt\_azymut”, należy wysłać puste. Zadanie: „GET\_PRODUKT\_POG”.

### 6.4 Czujnik skażeń

Integracja urządzenia typu „czujnik skażeń” jest analogiczna do 6.1 Syrena wirnikowa lub 6.2 Syrena elektroniczna ,wyłączając klucze nie dotyczące tak jak między innymi „produkt\_azymut”, należy wysłać puste. Zadanie: „GET\_PRODUKT\_CZS”.

### 6.5 Zegar DCF

Integracja urządzenia typu „zegar DCF” jest analogiczna do 6.1 Syrena wirnikowa lub 6.2 Syrena elektroniczna ,wyłączając klucze nie dotyczące tak jak między innymi „produkt\_azymut”, należy wysłać puste. Zadanie: „GET\_PRODUKT\_DCF”.

### 6.6 Limnimetry

Integracja urządzenia typu „limnimetr” jest analogiczna do 6.1 Syrena wirnikowa lub 6.2 Syrena elektroniczna ,wyłączając klucze nie dotyczące tak jak między innymi „produkt\_azymut”, należy wysłać puste. Zadanie: „GET\_PRODUKT\_CZW”.

### 6.7 Centrale

Integracja urządzenia typu „centrala” jest analogiczna do 6.1 Syrena wirnikowa lub 6.2 Syrena elektroniczna ,wyłączając klucze nie dotyczące tak jak między innymi „produkt\_azymut”, należy wysłać puste. Zadanie: „GET\_PRODUKT\_CEN”.

### 6.8 Wszystkie urządzenia w jednej paczce danych

Integracja wszystkich urządzeń jest analogiczna do 6.1 Syrena wirnikowa lub 6.2 Syrena elektroniczna ,wyłączając klucze nie dotyczące tak jak między innymi „produkt\_azymut”, należy wysłać puste. Zadanie: „GET\_PRODUKT\_ALL”.

## 6.9 Pozostałe urządzenia

Integracja pozostałych urządzeń jest analogiczna do 6.1 Syrena wirnikowa lub 6.2 Syrena elektroniczna, wyłączając klucze nie dotyczące tak jak między innymi „produkt\_azymut”, należy wysłać puste. System przewiduje rozbudowę o każdy rodzaj czujnika pomiarowego lub inne.

## 7 Zdarzenia systemowe

Serwer odbiera dane o zdarzeniach z systemów podrzędnych pod adresem:

IP/eth\_api\_inne/zdarzenie

### 7.1 Syrena wirnikowa

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „IN-SW”

Klucz – „data\_send” -> dane:

- identyfikator urządzenia
- status syreny: 1 – sprawna, 2 – niesprawna, 3 – brak łączności, 4 – ogłasza alarm (z uwzględnieniem ogłaszania alarmu np. 4A1; XX – spoczynek)
- pamięć alarmu : 0 – brak; 1 – sprawna syr. pamięć; 2- niesprawna syr. pamięć
- tryb pracy, (N – normalny, C - trening)
- zasilanie (brak, jest)
- informacja na temat przynależności do sektorów (zakres sektorów A B C D E F G H),  
Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „.” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### 7.2 Syrena elektroniczna

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „IN-SEL”

Klucz – „data\_send” -> dane:

- identyfikator urządzenia
- status syreny: 1 – sprawna, 2 – niesprawna, 3 – brak łączności, 4 – ogłasza komunikat/alarm (z uwzględnieniem jaki ogłasza alarm, lub jaki ogłasza komunikat np. 4A2 – ogłasza określony alarm, 4K1 – ogłasza określony komunikat, XX - spoczynek),
- pamięć alarmu : 0 – brak; 1 – sprawna syr. pamięć; 2- niesprawna syr. pamięć
- tryb pracy (N – normalny, C - trening),
- informacje na temat zasilania 24V (brak, w normie),
- informacje na temat zasilania 12V (brak, w normie),
- drzwi do syreny (otwarte, zamknięte),
- zasilanie 230V (brak, jest),
- napięcie 24 V z dokładnością do jednego miejsca po przecinku),

- napięcie 12 V z dokładnością do jednego miejsca po przecinku,
- sprawność głośników zestaw 1 (sprawne, niesprawne),
- sprawność głośników zestaw 2 (sprawne, niesprawne),
- sprawność wzmacniaczy zestaw 1 (sprawne, niesprawne),
- sprawność wzmacniaczy zestaw 2 (sprawne, niesprawne),
- napięcie akumulatora po teście z dokładnością do jednego miejsca po przecinku (V),
- prąd ładowania akumulatora (A),
- sprawność generatora (sprawny, niesprawny)

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### 7.3 Stacja pogodowa

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „IN-PG”

Klucz – „data\_send” -> dane:

- identyfikator urządzenia
- status stacji pogodowej: 1 – sprawna, 2 – niesprawna, 3 – brak łączności,
- data i godzina pomiaru (format *date*),
- temperatura w stopniach Celsjusza z dokładnością do jednego miejsca po przecinku,
- ciśnienie w hPa,
- wilgotność powietrza (%),
- siłę wiatru w m/s,
- kierunek wiatru,
- opady w mm/m<sup>2</sup> z ostatniej godziny,
- opady w mm/m<sup>2</sup> z ostatnich 24 godzin,

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### 7.4 Czujnik skażeń

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „IN-CZG”

Klucz – „data\_send” -> dane:

- identyfikator urządzenia
- status czujnika skażeń: 1 – sprawny, 2 – niesprawny, 3 – brak łączności,
- data i godzina pomiaru (format *date*),
- wartość średnia dawki promieniowania z ostatniej minuty w  $\mu\text{Sv/h}$
- wartość średnia dawki promieniowania z ostatniej godziny w  $\mu\text{Sv/h}$

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## 7.5 Limnimetr

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „IN-CZW”

Klucz – „data\_send” -> dane:

- identyfikator urządzenia
- status wodomierza: 1 – sprawny, 2 – niesprawny, 3 – brak łączności,
- data i godzina pomiaru(format *date*),
- pętla prądowa (mA)

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „, ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## 7.6 Alarmy

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „IN-ALARM”

Klucz – „data\_send” -> dane:

- jaki alarm (od A1 do A8)
- czy zostanie nadany komunikat po alarmie(T – tak, N – nie jeśli tak to jaki np. TK1)
- głośność(1 do 9),
- data (format *date*),
- na ilu syrenach podjęta próba uruchomienia
- identyfikatory syren na których została podjęta próba uruchomienia oddzielone separatorem  
„ <syr> ”

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „, ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin, date*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### 7.6.1 Alarm na pojedynczej syrenie

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „IN-SYRALARM”

Klucz – „data\_send” -> dane:

- jaki alarm (od A1 do A8)
- czy zostanie nadany komunikat po alarmie(T – tak, N – nie jeśli tak to jaki np. TK1)
- głośność(1 do 9),
- data (format *date*),
- identyfikator syreny

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### **7.6.2 Fizyczne uruchomienie alarmu z przycisku w syrenie**

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „MANUAL-SYRALARM”

Klucz – „data\_send” -> dane:

- jaki alarm (od A1 do A8)

- czy zostanie nadany komunikat po alarmie (T – tak, N – nie jeśli tak to jaki np. TK1)

- głośność (1 do 9),

- data (format *date*),

- identyfikator syreny

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### **7.6.3 Potwierdzenie włączenia alarmu w syrenie**

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „AUTO-SYRALARM”

Klucz – „data\_send” -> dane:

- potwierdzenie (1 – poprawnie, 0 – syrena uszkodzona)

- data (format *date*),

- identyfikator syreny

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## **7.7 Komunikaty**

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „IN-KOM”

Klucz – „data\_send” -> dane:

- jaki komunikat (od K1 do K8) dla mikrofonu (UU)

- data (format *date*),

- na ilu syrenach podjęta próba uruchomienia

- identyfikatory syren na których została podjęta próba uruchomienia oddzielone separatorem „ <syr> ”

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### **7.7.1 Komunikat na pojedynczej syrenie**

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „IN-SYRKOM”

Klucz – „data\_send” -> dane:

- jaki alarm (od K1 do K8) dla mikrofonu (UU)
- data (format *date*),
- identyfikator syreny

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „, ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### **7.7.2 Fizyczne uruchomienie komunikatu z przycisku w syrenie**

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „MANUAL-SYRKOM”

Klucz – „data\_send” -> dane:

- jaki alarm (od K1 do K8) dla mikrofonu (UU)
- data (format *date*),
- identyfikator syreny

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „, ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### **7.7.3 Potwierdzenie włączenia komunikatu w syrenie**

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „AUTO-SYRKOM”

Klucz – „data\_send” -> dane:

- potwierdzenie (1 – poprawnie, 0 – syrena uszkodzona)
- data (format *date*),
- identyfikator syreny

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „, ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## **7.8 Inne zdarzenia gdzie indziej nie sklasyfikowane**

Ze względu na ciągły rozwój oprogramowania SAOL MUW lista obsługiwanych zdarzeń stale rośnie. Obsługa zdarzeń innych jest analogiczna do schematów połączeń zawartych w rozdziale 7.



## 8 Zarządzanie systemami podrzędnymi

Komunikacja między serwerem a klientem odbywa się za pomocą metody POST(curl). Serwer wysyła żądania na określone adresy IP, kontroler i funkcję. System jest uniwersalny, można wszystkie poniższe kontrolery i funkcje przekierować na własne w systemie klienta. Poniższa lista żądań zostaje wysyłana na wiele adresów IP, adres IP systemu powiatowego (jeżeli jest wpiętą centrala powiatowa w system) oraz na wszystkie adresy IP central miejskich, na których ma zostać uruchomiony alarm.

Centrala powiatowa podejmuje żądanie centrali MUW i również wysyła żądanie do central miejskich, centrala miejska podejmuje żądanie, które pierwsze do niej dotrze, czy to z centrali MUW czy z centrali POWIATOWEJ, dodatkowo potwierdza odebranie żądania na adres IP centrali powiatowej oraz adres IP centrali MUW.

### 8.1 Uruchomienie alarmów z MUW na wszystkich syrenach w powiecie/mieście

Adres odbioru:

IP\_klient/eth\_api/alarm\_on

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „ON-ALARM”

Klucz – „data\_send” -> dane:

- jaki alarm (od A1 do A8)

- czy zostanie nadany komunikat po alarmie (T – tak, N – nie jeśli tak to jaki np. TK1)

- głośność (1 do 9),

- data (format date),

- w ilu miastach podjęta próba uruchomienia

- w przypadku uruchomienia alarmu na kilku wybranych miastach w powiecie zostanie przestany ciąg identyfikatorów miast/obszarów oddzielonych od siebie separatorem <mst>, gdy w całym powiecie string „CALE”.

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „,” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin

Klucz – „check” -> dane -> string utworzony z danych: session\_id,date,zadanie,data\_send,PIN

### 8.2 Uruchomienie alarmów z MUW na wybranych syrenach w powiecie/mieście

Adres odbioru:

IP\_klient/eth\_api/alarm\_on

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „ON-SYRALARM”

Klucz – „data\_send” -> dane:

- jaki alarm (od A1 do A8)

- czy zostanie nadany komunikat po alarmie (T – tak, N – nie jeśli tak to jaki np. TK1)

- głośność (1 do 9),

- data (format *date*),
  - na ilu syrenach podjęta próba uruchomienia
  - `<cen>numer_identyfikacyjny_cetralli</cen>` identyfikatory syren na których została podjęta próba uruchomienia oddzielone separatorem „`<syr>`” (gdy wszystkie syreny są zaznaczone w mieście podajemy string `ALL<syr>`), obiekty oddzielamy od siebie parametrem `<spacja>`
- Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „`,`” (przecinek).

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „`,`” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### 8.3 Uruchomienie komunikatów głosowych z MUW na wszystkich syrenach w powiecie/mieście

Adres odbioru:

IP\_klient/eth\_api/alarm\_on

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „ON-KOM”

Klucz – „data\_send” -> dane:

- jaki komunikat (od K1 do K8) dla mikrofonu UU

- głośność (1 do 9),

- data (format *date*),

- na ilu obiektach podjęta próba uruchomienia

- w przypadku uruchomienia komunikatu na kilku wybranych miastach w powiecie zostanie przesłany ciąg identyfikatorów miast/obszarów oddzielonych od siebie separatorem `<mst>`, gdy w całym powiecie string „CALE”.

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „`,`” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### 8.4 Uruchomienie komunikatów głosowych z MUW na wybranych syrenach w powiecie/mieście

Adres odbioru:

IP\_klient/eth\_api/alarm\_on

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „ON-SYRKOM”

Klucz – „data\_send” -> dane:

- jaki komunikat (od K1 do K8) dla mikrofonu UU
  - głośność(1 do 9),
  - data (format *date*),
  - na ilu syrenach podjęta próba uruchomienia
  - `<cen>numer_identyfikacyjny_cetr</cen>` identyfikatory syren na których została podjęta próba uruchomienia oddzielone separatorem „ <sy> ” (gdy wszystkie syreny są zaznaczone w mieście podajemy string ALL<sy> ), obiekty oddzielamy od siebie parametrem <spacja>
- Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## **8.5 Ogłaszanie komunikatów głosowych nadawanych przez mikrofon z centrali MUW na wszystkich syrenach w powiecie/mieście**

Adres odbioru:

IP\_klient/eth\_api/alarm\_on

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „ON-KOM”

Klucz – „data\_send” -> dane:

- UU
- głośność(1 do 9),
- data (format *date*),
- na ilu obiektach podjęta próba uruchomienia
- w przypadku uruchomienia komunikatu na kilku wybranych miastach w powiecie zostanie przesłany ciąg identyfikatorów miast/obszarów oddzielonych od siebie separatorem <mst>, gdy w całym powiecie string „CALE”.

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## **8.6 Ogłaszanie komunikatów głosowych nadawanych przez mikrofon z centrali MUW na wybranych syrenach w powiecie/mieście**

Adres odbioru:

IP\_klient/eth\_api/alarm\_on

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „ON-SYRKOM”

Klucz – „data\_send” -> dane:

- UU

- głośność(1 do 9),

- data (format date),

- na ilu syrenach podjęta próba uruchomienia

- <cen>numer\_identyfikacyjny\_cetrali</cen> identyfikatory syren na których została podjęta próba uruchomienia oddzielone separatorem „ <syr> ” (gdy wszystkie syreny są zaznaczone w mieście podajemy string ALL<syr> ), obiekty oddzielamy od siebie parametrem <spacja>

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin

Klucz – „check” -> dane -> string utworzony z danych: session\_id,date,zadanie,data\_send,PIN

## 8.7 Zatrzymanie ogłaszania alarmów i komunikatów alarmowych

### 8.7.1 Zatrzymanie alarmów z MUW na wszystkich syrenach w powiecie/mieście

Adres odbioru:

IP\_klient/eth\_api/stop

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „STOP-ALARM”

Klucz – „data\_send” -> dane:

- data (format date),

- na ilu obiektach podjęta próba zatrzymania

- w przypadku zatrzymania alarmu na kilku wybranych miastach w powiecie zostanie przestany ciąg identyfikatorów miast/obszarów oddzielonych od siebie separatorem <mst>, gdy w całym powiecie string „CALE”.

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin

Klucz – „check” -> dane -> string utworzony z danych: session\_id,date,zadanie,data\_send,PIN

### 8.7.2 Zatrzymanie alarmów z MUW na wybranych syrenach w powiecie/mieście

Adres odbioru:

IP\_klient/eth\_api/stop

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „STOP-SYRALARM”

Klucz – „data\_send” -> dane:

- data (format *date*),

- na ilu syrenach podjęta próba zatrzymania

- <cen>numer\_identyfikacyjny\_cetrali</cen> identyfikatory syren na których została podjęta próba uruchomienia oddzielone separatorem „ <syr> ” (gdy wszystkie syreny są zaznaczone w mieście podajemy string ALL<syr> ), obiekty oddzielamy od siebie parametrem <spacja>

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### 8.7.3 Zatrzymanie komunikatów głosowych z MUW na wszystkich syrenach w powiecie/mieście

Adres odbioru:

IP\_klient/eth\_api/alarm\_on

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „STOP-KOM”

Klucz – „data\_send” -> dane:

- na ilu obiektach podjęta próba zatrzymania

- w przypadku zatrzymania komunikatu w kilku wybranych miastach w powiecie zostanie przesłany ciąg identyfikatorów miast/obszarów oddzielonych od siebie separatorem <mst>, gdy w całym powiecie string „CALE”.

- data (format *date*),

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych: *klucz\_licencyjny,session\_id, date,zadanie, data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date, zadanie,data\_send,PIN*

### 8.7.4 Zatrzymanie komunikatów głosowych z MUW na wybranych syrenach w powiecie/mieście

Adres odbioru:

IP\_klient/eth\_api/alarm\_on

Klucz – „session\_id” -> dane: session\_id

Klucz – „date” -> dane: date

Klucz – „zadanie” -> dane: „STOP-SYRKOM”

Klucz – „data\_send” -> dane:

- data (format *date*),

- na ilu syrenach podjęta próba zatrzymania

- *<cen>numer\_identyfikacyjny\_cetralli</cen>* identyfikatory syren na których została podjęta próba uruchomienia oddzielone separatorem „ <syr> ” (gdy wszystkie syreny są zaznaczone w mieście podajemy string ALL<syr> ), obiekty oddzielamy od siebie parametrem <spacja> Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## 8.8 Kasowanie pamięci alarmów

W przypadku ogłoszenia pełnego alarmu syreny w systemie MUW zostają oznaczone specjalną flagą. System SAOL MUW dysponuje funkcjami, które umożliwiają zarządzanie tym statusem.

### 8.8.1 Serwer -> klient

#### 8.8.1.1 Na wszystkich syrenach

Adres odbioru:

IP\_klient/eth\_api/pamiec

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „STOP-PAMIEC”

Klucz – „data\_send” -> dane:

- data (format *date*),

- string ALL,

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ”(przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

#### 8.8.1.2 Na wybranych syrenach

Adres odbioru:

IP\_klient/eth\_api/pamiec

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „STOP-PAMIEC”

Klucz – „data\_send” -> dane:

- data (format *date*),

- identyfikatory syren na których usuwamy informację o pamięci alarmu oddzielone separatorem

„ <syr> ”

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## 8.8.2 Klient -> serwer

### 8.8.2.1 Na wszystkich syrenach

Adres odbioru:

IP/eth\_api\_inne/pamiec

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „STOP-PAMIEC”

Klucz – „data\_send” -> dane:

- data (format *date*),

- string *ALL*,

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

### 8.8.2.2 Na wybranych syrenach

Adres odbioru:

IP/eth\_api\_inne/pamiec

Klucz – „session\_id” -> dane: *session\_id*

Klucz – „date” -> dane: *date*

Klucz – „zadanie” -> dane: „STOP-PAMIEC”

Klucz – „data\_send” -> dane:

- data (format *date*),

- identyfikatory syren na których usuwamy informację o pamięci alarmu oddzielone separatorem „ <syr> ”



Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Z powyższych danych tworzymy string, a parametry rozdzielamy między sobą znakiem interpunkcyjnym „ , ” (przecinek).

Klucz – „PIN” -> dane -> string utworzony z danych:

*klucz\_licencyjny,session\_id,date,zadanie,data\_send,OB\_pin*

Klucz – „check” -> dane -> string utworzony z danych: *session\_id,date,zadanie,data\_send,PIN*

## **9 Bezpieczeństwo odbierania powiadomień URLC**

Wygenerowany kod uwierzytelniający dla klienta jest podstawą do wyliczenia sumy kontrolnej powiadomień, którą klient dostaje w powiadomieniach URLC. Skrypt odbierający powiadomienia URLC powinien sprawdzić czy w parametrze nadstąnym z serwera jest taka sama wartość jak ta wyliczona przez skrypt klienta. System odbierający powiadomienia/żądania powinien sprawdzić IP serwera z którego otrzymuje dane. IP serwera jest w posiadania MUW. Należy również oprócz sumy kontrolnej porównać PIN przesłany z serwera.