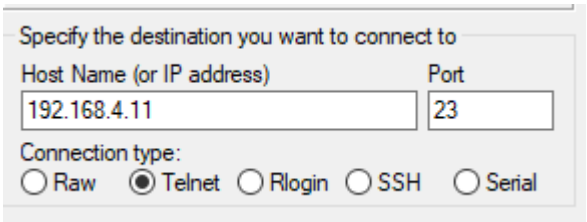
GATE HTTP

Instrukcja użytkownika

SPIS TREŚCI

1. MOŻLIWOŚĆ POŁĄCZENIA Z GATE ZA POMOCĄ TELNET

Dla modułu Gate Http możliwy jest podgląd skryptów Lua. W przypadku błędu konfiguracji (tryb emergency) możliwy jest podgląd miejsca błędu w utworzonej konfiguracji LUA. Połączenie jest nawiązywane za pomocą protokołu Telnet – w tym celu można wykorzystać np. program PuTTY. Przykładowe parametry do nawiązania połączenia:



Do wywołania połączenia po stronie Gate'a wykorzystane mogą być dwie metody:

- StartConsole – Uruchamia konsolę Lua. W momencie wywołania metody, użytkownik ma 10s na ustawienie połączenia z Gate. Przy poprawnym połączeniu, na terminalu (klient) zostanie zwrócona informacja o poprawnym połączeniu
  CLU SN Telnet session started.

  >
- StartConsoleOnReboot – umożliwia nawiązanie połączenia przy następnym restarcie Gate'a. Po restarcie , użytkownik ma 10s na ustawienie połączenia z Gate. Przy poprawnym połączeniu, na terminalu (klient) zostanie zwrócona informacja o poprawnym połączeniu
  CLU SN initializing...

  CLU: running user.lua...

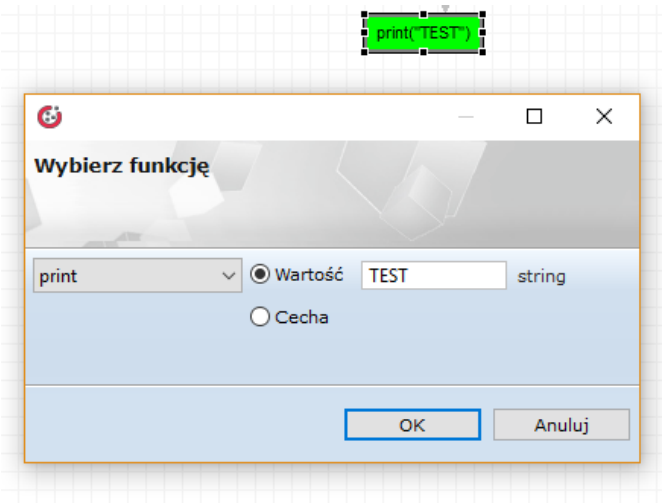
  CLU: running om.lua...

  CLU: running OnInit...

  CLU: Project loaded.

  >

Aby na konsoli wyświetlić, np. wartość danej cechy, należy wykorzystać komponent Blok funkcyjny i wybrać metodę **print** a następnie wybrać żądaną cechę.


print("TEST")
Wybierz funkcję
print
Wartość
TEST
string
Cecha
OK
Anuluj

Grenton Sp. z o.o.
ul. Na Wierzchowinach 3
30-222 Kraków, Polska

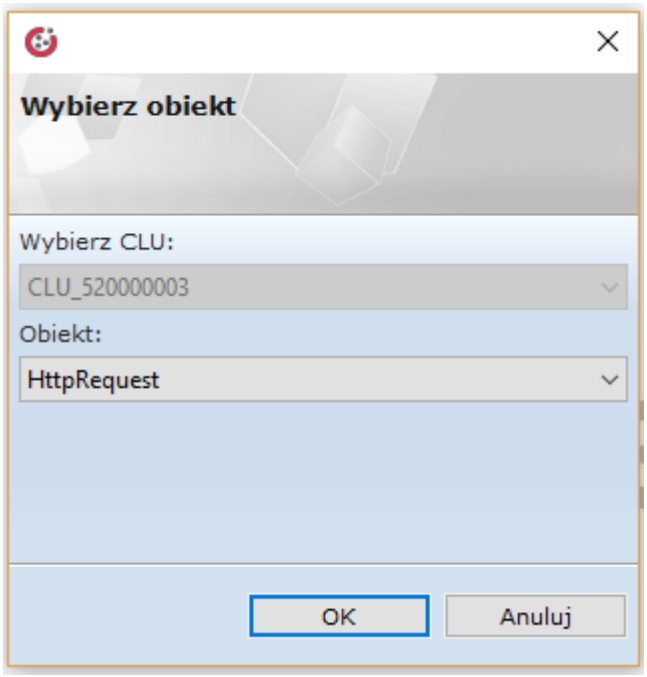
2. WYSYŁANIE POPRAWNEGO ZAPYTANIA REQUEST – PRZYKŁAD

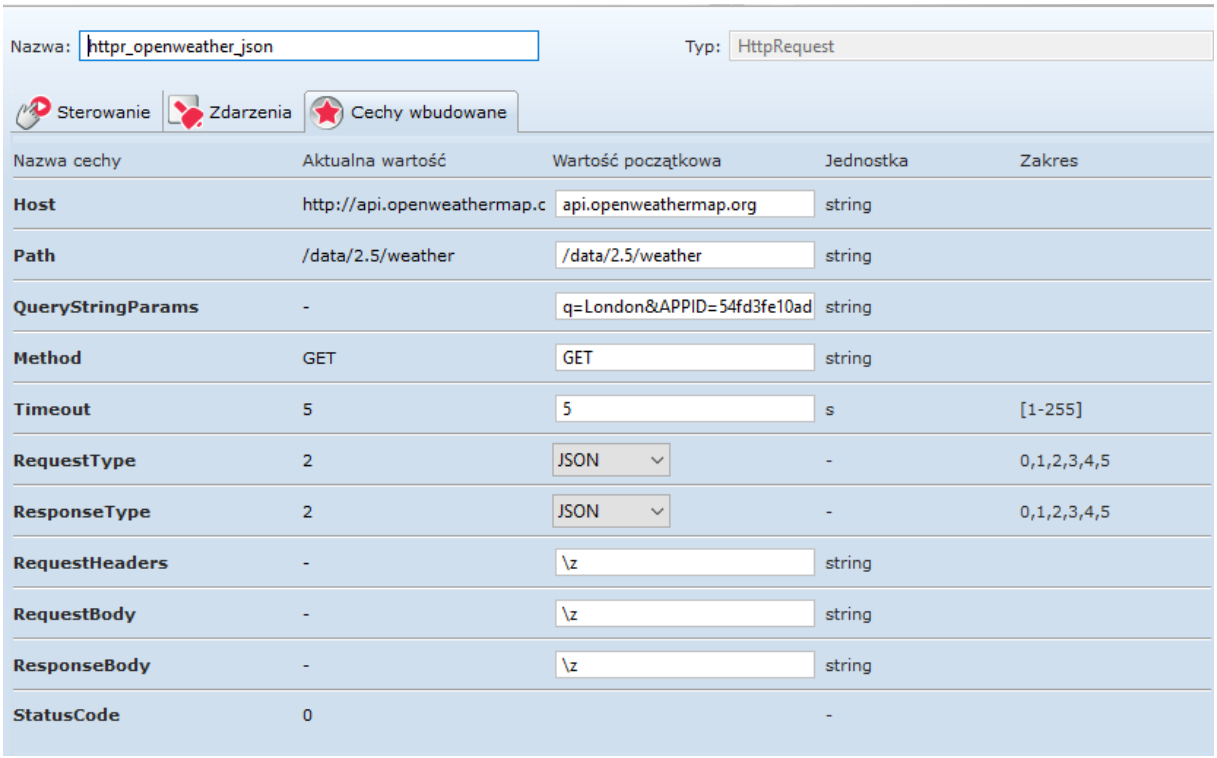
Według przykładu na stronie openweathermap.org, zapytanie API wygląda następująco:

API call:

http://api.openweathermap.org/data/2.5/weather?q=London&APPID={APIKEY}

Aby zastosować moduł Gate do odbierania zapytań, należy utworzyć obiekt wirtualny Request





W obiekcie HttpRequest należy ustawić następujące parametry:

Host api.openweathermap.org
Path /data/2.5/
QueryStringParams q=London&APPID={APIKEY}

Method GET
RequestType JSON
ResponseType JSON

Po wysłaniu konfiguracji i wywołaniu Metody SendRequest, StatusCode przyjmuje wartość 200 (OK).
Otrzymana odpowiedź na zapytanie jest przetrzymywana w cesze ResponseBody. Dla ustawionego ResponseType JSON, odpowiedź jest parsowana z json do tabeli. Wartość cechy jest niewidoczna z poziomu OM. Wartości odpowiedzi należy wyciągnąć z odpowiedzi z poziomu skryptu.

3. WYCIĄGANIE WARTOŚCI Z OTRZYMANEJ ODPOWIEDZI Z SERWERA – PRZYKŁADY

Uwaga1
Uzyskaną odpowiedź ResponseBody należy przypisać do zmiennej lokalnej (w skrypcie)

Przykładowo:
local resp = GATE->httpr_openweather_json->ResponseBody

Następnie w skryptach należy wykonywać operację na zmiennej resp

Przykład odczytu odpowiedzi. Przykłady dla XML, JSON

Otrzymane odpowiedzi w zależności od ich typu (ResponseType) są odpowiednio parsowane do postaci tabeli. Poniżej przykładowe odpowiedzi w formacie XML oraz JSON oraz sposób odczytania danej wartości. Przykładowe odczyty wartości są zapisywane do zmiennych lokalnych (wewnątrz skryptu). Aby była możliwość wykorzystania zmiennej (np. do wyświetlania w aplikacji), należy przypisywać do zmiennych globalnych (cechy użytkownika).
Poniższe przykłady wykorzystują odpowiedzi z serwisu pogodowego openweathermap.org

JSON:
Przykładowa odpowiedź (openweathermap.org):

```
resp = [[
{"coord":
{"lon":145.77,"lat":-16.92},
"weather":[{"id":803,"main":"Clouds","description":"broken clouds","icon":"04n"}],
"base":"cmc stations",
"main":{"temp":293.25,"pressure":1019,"humidity":83,"temp_min":289.82,"temp_max":295.37},
"wind":{"speed":5.1,"deg":150},
"clouds":{"all":75},
"rain":{"3h":3},
"dt":1435658272,
"sys":{"type":1,"id":8166,"message":0.0166,"country":"AU","sunrise":1435610796,"sunset":1435650870},
"id":2172797,
"name":"Cairns",
"cod":200}
]]
```

Jak odczytać:

- Wartość parametru lon
  {"coord":
  {"lon":**145.77**,"lat":-16.92},
  "weather":[{"id":803,"main":"Clouds","description":"broken clouds","icon":"04n"}],
  "base":"cmc stations",
  "main":{"temp":293.25,"pressure":1019,"humidity":83,"temp_min":289.82,"temp_max":295.37},

  W skrypcie:
  local lon = resp.coord.lon
  Po wywołaniu skryptu do zmiennej lokalnej (zmienna skryptu) zostanie przypisana wartość 145.77

- Wartość parametru description
  {"coord":
  {"lon":145.77,"lat":-16.92},
  "weather":[{"id":803,"main":"Clouds","description":**"broken clouds"**,"icon":"04n"}],
  "base":"cmc stations",
  "main":{"temp":293.25,"pressure":1019,"humidity":83,"temp_min":289.82,"temp_max":295.37},

W skrypcie:
local description = resp.weather[1].description
Po wywołaniu skryptu do zmiennej lokalnej (zmienna skryptu) zostanie przypisana wartość „broken clouds”

XML:

Przykładowa odpowiedź (openweathermap):

```
resp= [[
<current>
   <city id="2643741" name="City of London">
    <coord lon="-0.09" lat="51.51">
    <country>GB</country>
    <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
   </city>
   <temperature value="72.34" min="66.2" max="79.88" unit="fahrenheit"/>
   <humidity value="43" unit="%">
   <pressure value="1020" unit="hPa">
   <wind>
    <speed value="7.78" name="Moderate breeze">
    <direction value="140" code="SE" name="SouthEast">
   </wind>
   <clouds value="0" name="clear sky">
   <visibility value="10000">
   <precipitation mode="no">
   <weather number="800" value="Sky is Clear" icon="01d">
   <lastupdate value="2015-06-30T08:36:14">
</current>
]]
```

Jak odczytać:

- Wartość atrybutu id w tagu city

```
<current>
        <city id="2643741" name="City of London">
                <coord lon="-0.09" lat="51.51">
                <country>GB</country>
                <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
        </city>
```

W skrypcie:
local city_id = resp[1].id

Po wywołaniu skryptu do zmiennej lokalnej (zmienna skryptu) zostanie przypisana wartość 2643741

- Wartość znajdująca się pomiędzy tagiem <country>

```
<current>
        <city id="2643741" name="City of London">
                <coord lon="-0.09" lat="51.51">
                <country>GB</country>
                <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
```

</city>

W skrypcie:

local country = resp[1][2][1]
Po wywołaniu skryptu do zmiennej lokalnej (zmienna skryptu) zostanie przypisana wartość „GB”

- Nazwa tagu
  <current>
  <city id="2643741" name="City of London">
  <coord lon="-0.09" lat="51.51">
  <**country**>GB</country>
  <sun rise="2015-06-30T03:46:57" set="2015-06-30T20:21:12">
  </city>

  W skrypcie:
  local nameTag = resp[1][2].xmlTag

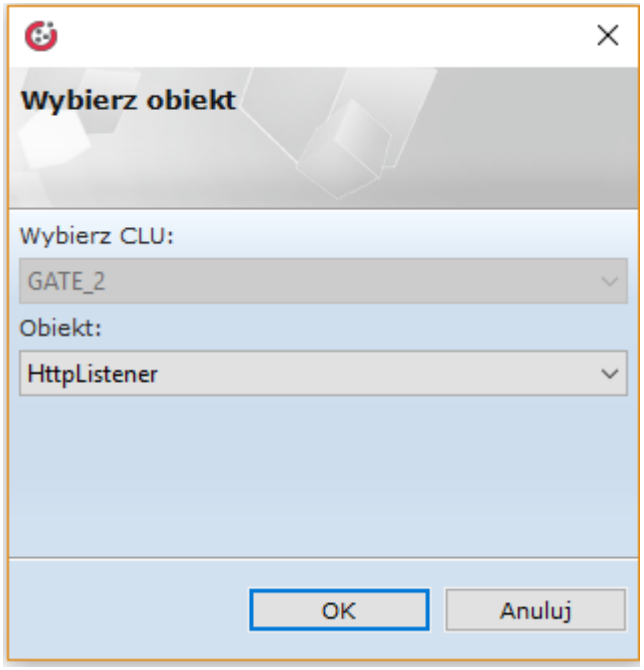
  Po wywołaniu skryptu do zmiennej lokalnej (zmienna skryptu) zostanie przypisana wartość „country”

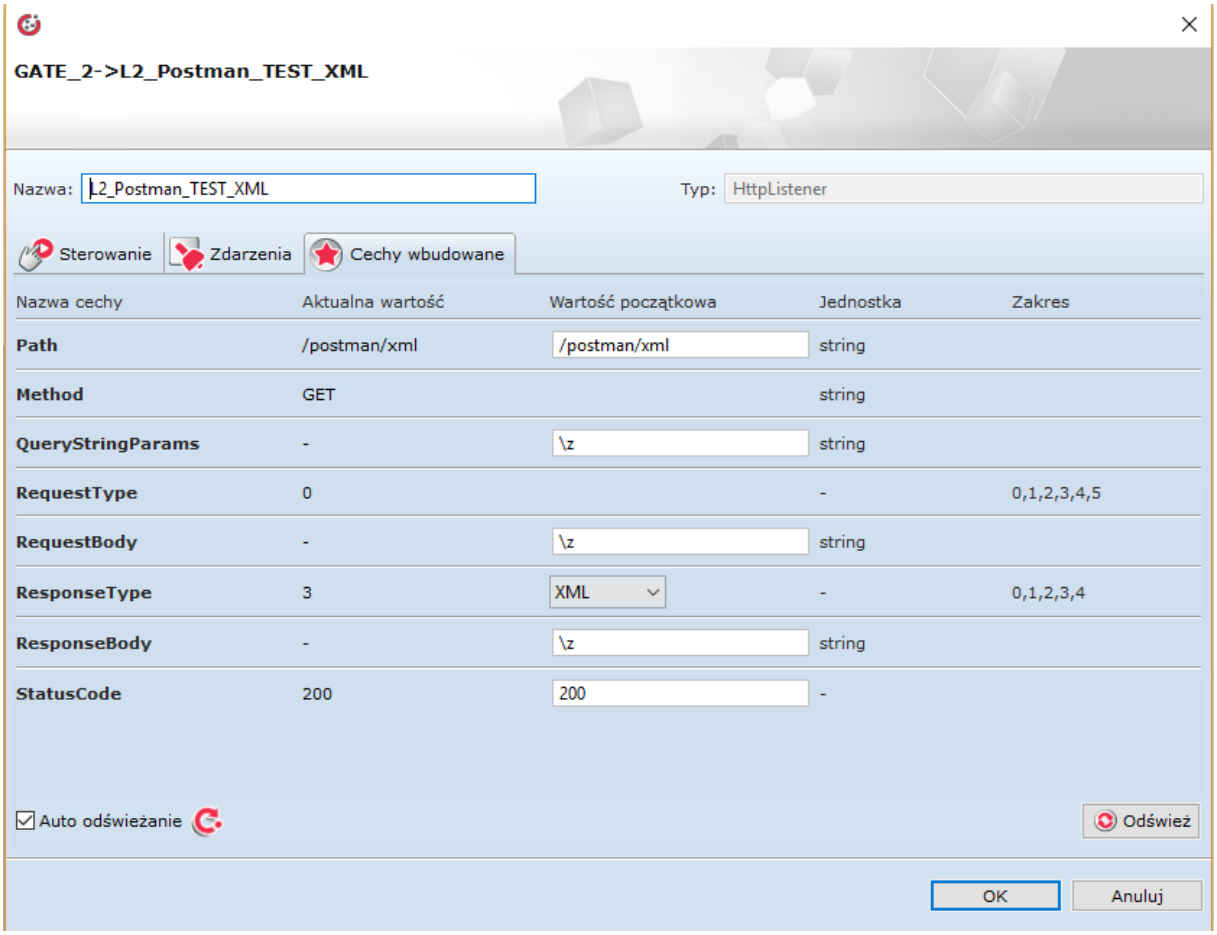
4. NASŁUCHIWANIE NA WYSŁANE ZAPYTANIE – LISTNER - PRZYKŁAD

W przypadku nasłuchiwania na zapytanie Request do modułu Gate – przykładowo:

GET 192.168.4.12/postman/xml

Należy utworzyć obiekt wirtualny HttpListner





W obiekcie HttpRequest należy ustawić następujące parametry:

Path /postman/xml
ResponseType XML
StatusCode 200
Do zdarzenia **OnRequest** należy utworzyć skrypt, który będzie tworzył poprawną odpowiedź i wysyłał ją zwrotnie.

5. PRZYGOTOWANIE ODPOWIEDZI NA ZAPYTANIE – PRZYKŁADY

Odpowiedź jest tworzona w zmiennej lokalne resp.

Po przygotowaniu odpowiedzi należy ją ustawić dla cechy ResponseBody(resp) a następnie wysłać za pomocą metodą SendRepsone()

XML

Aby w odpowiedzi wysłać wartość danej cechy:

```
local resp ="<clu><temperature>" ..CLUZ->x103478262_ONEW_SENSOR1-
>Value.."</temperature></clu>"


GATE_2->Listner_XML->SetResponseBody(resp)

GATE_2->Listner_XML->SendResponse()
```

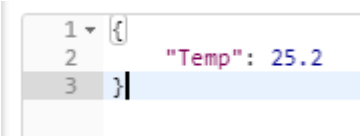
Przesłana odpowiedź wygląda następująco:

```
1 <clu>
2     <temperature>25.3</temperature>
3 </clu>
```

JSON

```
local resp = {

      Temp = CLUZ->x103478262_ONEW_SENSOR1->Value

}


GATE_2->Listner_JSON->SetResponseBody(resp)

GATE_2-> Listner_JSON->SendResponse()
```

Przesłana odpowiedź wygląda następująco:



6. PARAMETRY KONFIGURACYJNE

OBIEKT GATE_HTTP

CECHY:

- **UpTime** Czas pracy urządzenia od ostatniego resetu (w sekundach)
- **UnixTime** Zwraca aktualny uniksowy znacznik czasu
- **FirmwareVersion** Wersja oprogramowania Gate
- **ClientReportInterval** Okres raportowania o zmianach cech

METODY:

- **SetDateTime** Ustawia datę i czas
- **SetClientReportInterval** Ustawia okres raportowania o zmianach cech
- **SetUpdateTime** Ustawia czas, co jaki stan centralki jest uaktualniany
- **StartConsole** Uruchamia konsolę Lua
- **StartConsoleOnReboot** Uruchamia konsolę Lua przy kolejnym uruchomieniu modułu

ZDARZENIA:

- **OnInit** Zdarzenie wywolywane jednorazowo w momencie inicjalizacji urządzenia

OBIEKT HTTPREQUEST

Uwaga. Cechy opisane jako nieustawialne są cechami zawierające odpowiedzi. Wartości początkowe tych cech należy pozostawić niezmienione. Wszelkie operacje na tych zmiennych należy wykonywać na skryptach (oraz zmiennych lokalnych)

CECHY:

- **Host** Adres hosta
- **Path** Ścieżka zapytania
- **QueryStringParams** Parametry zapytania. \z oznacza brak
- **Method** Typ metody wysyłanej w zapytaniu np. GET, POST
- **Timeout** Dopuszczalny czas odpowiedzi
- **RequestType** Typ zawartości wysyłanego zapytania. Definiuje parametr content-type w nagłówku zapytania. W zależności od wybranego typu zawartość cechy RequestBody jest odpowiednio serializowana:
  - 0 - None - niezdefiniowany. W nagłówku nie jest wysyłane content-type. Zawartość cechy RequestBody nie jest serializowana.
  - 1 - Text - content-type: text/plain. Zawartość cechy RequestBody nie jest serializowana.
  - 2 - JSON - content-type: application/json. Zawartość cechy RequestBody jest serializowana do formatu JSON.
  - 3- XML - content-type: text/xml. Zawartość cechy RequestBody jest serializowana do formatu XML.

    - 4 - FormData - content-type: application/x-www-form-urlencoded. Zawartość cechy RequestBody jest serializowana do tabeli.
    - 5- Other - typ zawartości (content-type) jest inny niż wbudowany. Typ można zdefinować umieszczając go w nagłówku (cecha RequestHeaders). Zawartość nie jest serializowana.
- **ResponseType** Typ oczekiwanej odpowiedzi. Definiuje parametr Accept w nagłówku zapytania. W zależności od wybranego typu zawartość otrzymanej odpowiedzi (cechy ResponseBody) jest odpowiednio parsowana do tabeli:
    - 0 - None - parametr Accept nie jest wysyłany w nagłowku wysyłanego zapytania. Odpowiedź (cecha ResponseBody) nie jest parsowana.
    - 1 - Text - Accept: text/plain. Odpowiedź (cecha ResponseBody) nie jest parsowana.
    - 2 - JSON - Accept: application/json. Odpowiedź (cecha ResponseBody) jest parsowana z JSON.
    - 3 - XML - Accept: text/xml. Odpowiedź (cecha ResponseBody) jest parsowana z XML.
    - 4 - FormData - Accept: application/x-www-form-urlencoded. Odpowiedź (cecha ResponseBody) jest parsowana.
    - 5 - Other - parametr Accept nagłówka jest inny niż wbudowany. Parametr można zdefinować umieszczając go w nagłówku (cecha RequestHeaders)
- **RequestHeaders** Dodatkowe nagłówki zapytania HTTP. \z oznacza brak zawartości.
- **RequestBody** Zawartość wiadomości wysyłanej w zapytaniu. \z oznacza brak zawartości
- **ResponseBody** Zawartość wiadomości otrzymanej po wysłaniu zapytania. (cecha wykorzystywana do odczytu w skryptach - nieustawialna)
- **StatusCode** Status odpowiedzi HTTP

METODY:

- **SendRequest** Wysyła zapytanie
- **AbortRequest** Przerywa obsługę zapytania
- **Clear** Usuwa treść zapytania
- **SetHost** Ustawia adres hosta
- **SetPath** Ustawia ścieżkę zapytania
- **SetQueryStringParams** Ustawia parametry zapytania
- **SetMethod** Ustawia metodę zapytania
- **SetTimeout** Ustawia dopuszczalny czas odpowiedzi
- **SetResponseType** Ustawia typ wiadomości
- **SetResponseBody** Ustawia typ odpowiedzi
- **SetRequestHeaders** Ustawia dodatkowe nagłówki HTTP
- **SetRequestBody** Ustawia zawartość wiadomości

ZDARZENIA:

- **OnRequestSent** Zdarzenie wywoływane w momencie wysłania zapytania
- **OnResponse** Zdarzenie wywoływane w momencie otrzymania odpowiedzi

OBIEKT HTTPLISTNER

Uwaga. Cechy opisane jako nieustawialne są cechami zawierające odpowiedzi. Wartości początkowe tych cech należy pozostawić niezmienione. Wszelkie operacje na tych zmiennych należy wykonywać na skryptach (oraz zmiennych lokalnych)

CECHY:

- **Path** Ścieżka zapytania
- **Method** Typ metody otrzymanej w zapytaniu np. GET, POST
- **QueryStringParams** Zwraca parametry zapytania HTTP (cecha wykorzystywana do odczytu w skryptach - nieustawialna)
- **RequestType** Typ otrzymanego zapytania. W zależności od wybranego typu, zawartość otrzymanego zapytania (cechy RequestBody) jest odpowiednio parsowana do tabeli:
  - 0 - None - Odpowiedź nie jest parsowana.
  - 1 - Text - Odpowiedź nie jest parsowana.
  - 2 - JSON - Odpowiedź jest parsowana z JSON.
  - 3 - XML - Odpowiedź jest parsowana z XML.
  - 4 - FormData - Odpowiedź jest parsowana.
  - 5 - Other - Odpowiedź nie jest parsowana
- **RequestBody** Zwraca treść zapytania HTTP (cecha wykorzystywana do odczytu w skryptach - nieustawialna)
- **ResponseType** Typ zawartości wysłanej odpowiedzi na zapytanie. Definiuje parametr content-type w nagłówku odpowiedzi. W zależności od wybranego typu, zawartość cechy ResponseBody jest odpowiednio serializowana:
  - 0 - None - niezdefiniowany. W nagłówku nie jest wysyłane content-type. Zawartość nie jest serializowana.
  - 1 - Text - content-type: text/plain. Zawartość nie jest serializowana.
  - 2 - JSON - content-type: application/json. Zawartość RequestBody jest serializowana do formatu JSON.
  - 3 - XML - content-type: text/xml. Zawartość RequestBody jest serializowana do formatu XML.
  - 4 - FormData - content-type: application/x-www-form-urlencoded. Zawartość RequestBody jest serializowana
  - 5 - Other - parametr Accept nagłówka jest inny niż wbudowany. Parametr można zdefinować umieszczając go w nagłówku (cecha RequestHeaders)
- **ResponseBody** Zwraca treść odpowiedzi HTTP (cecha wykorzystywana do odczytu w skryptach
- **StatusCode** Status wysyłanej odpowiedzi HTTP. Obsługiwane statusy:
  - 200 - OK
  - 201 - Utworzono
  - 202 - Przyjęto
  - 204 - Brak zawartości
  - 205 - Przywróć zawartość
  - 400 - Nieprawidłowe zapytanie
  - 403 - Zabroniony
  - 404 - Nie znaleziono
  - 405 - Niedozwolona metoda
  - 406 - Niedozwolone
  - 408 - Koniec czasu oczekiwania na żądanie
  - 409 - Konflikt
  - 410 - Zniknął (usunięto)

METODY:

- **SendResponse** Wysyła odpowiedź na zapytanie
- **Clear** Usuwa treść odpowiedzi
- **SetPath** Ustawia ścieżkę zapytania
- **SetResponseType** Ustawia typ odpowiedzi

- **SetResponseBody** Ustawia treść odpowiedzi
- **SetStatusCode** Ustawia status odpowiedzi

ZDARZENIA:

- **OnRequest** Zdarzenie wywoływane w momencie otrzymania zapytania