ITS API User Manual

Table of Contents

        - Example: when an event occurs 3 times consecutively within 1 second, relay R01 on
    - **Audio**
        - **Format**
        - **Audio System**
        - Example: 사전정의된 음원 4번을 40%의 볼륨으로 플레이 요청
        - Example: Request to play predefined sound source 4 at 40% volume
        - Example: 사용자 오디오 Warning을 100%의 볼륨으로 10번 재생 요청
        - Example: Request to play user audio warning 10 times at 100% volume
        - Example: 웹 오디오 nocturne을 70%의 볼륨으로 2번 플레이 요청
        - Example: Request to play web audio nocturne 2 times at 70% volume
    - **Talk**
        - **Format**
        - Example: speaking
        - Example: listening
        - Example: disconnect
    - **Camera**
        - **Format**
        - Footprint(Blackbox) System
        - Example: Camera footprint Test
    - **Trigger**
        - **Format**
        - Example: trigger
    - **Messenger**
        - **Format**
        - SNS Messenger
        - Example: sendMessage
    - **Custom**
        - **Format - tcp_socket**
        - Example: tcp_sockert - JSON
        - Example: tcp_sockert - Non JSON
        - Example: tcp_sockert - Non JSON - IMS protocol
        - **Format - http_get/http_post**
        - Example: http_get
        - Example: http_post
    - **System**
        - **Format**
        - Example: sleep
        - Example: get_name
        - Example: set_name
        - Example: get_time
        - Example: set_time
        - Example: list_audio
        - Example: stop_audio
        - Example: enable_audio
        - Example: disable_audio
        - Example: trigger_io

ITS API는 Console(http://API_IP:28080)을 통해 관리된다.

인라인 명령(Inline command)이나 코멘드라인인터프리터(CLI) 또는 브라우저(URL)로 테스트 및 제어가 가능하다.

ITS API is managed through Console(http://API_IP:28080).

Inline command, CLI or URL can be used for testing and controlling.

FORMATS

기본적으로 ITS API는 3가지 형식의 이벤트에 대응한다.

In general, ITS API responds to 3 types of events.

Event Occurence

```
센서, 스케즐러, 타이머에 따른 이벤트가 발생된다.
Events occur according to sensor, scheduler and timer.
```

Sensor Event

- 8개의 Sensor : Dry Contact 신호기반
- 8 Sensors : Dry Contact Based on Alerts

> **IO Table**
>
> | Sensor | S01 | S02 | S03 | S04 | S05 | S06 | S07 | S08 |
> |---|---|---|---|---|---|---|---|---|
> | ID | io01 | io02 | io03 | io04 | io05 | io06 | io07 | io08 |

Scheduler Event

- 4개의 Crontab : Linux Crontab 기반
- 4 Crontabs : Linux Crontab based

> **Scheduler Table**
>
> | Cron | C01 | C02 | C03 | C04 |
> |---|---|---|---|---|
> | ID | am01 | am02 | am03 | am04 |

Timer Event

- 1개의 Heartbeat : Linux Timer 기반
- 1 Heartbeat : Linux Timer based

사용자는 이벤트 발생시 관련 항목에 JSON 형식의 명령문 요청이 가능한데 이러한 명령문을 ITS 액션스크립트(ITS Action Script)라 한다.

User can request commands in the form of JSON in the occurence of events, and these commands are referred to as ITS Action Script.

Action Script

Format

딕셔너리 형식의 JSON으로 구성은 host, port, data 이다.

JSON is in dictionary format and consists of host, port and data.

```
{
    "host":"",
    "port":"",
    "data": [ { command }, ... ]
}
```

- host(Option), port(Option) : Master Server
  - "data"내의 명령이 실행될 원격 IP 및 Port이며 미설정시 자신에게 요청한다.
  - ITS API port는 34001로 고정이다.
- data : Command List
  - 배열(list[ ])로 구성되며 명령문의 집합체 이다.

- host(Option), port(Option) : Master Server
  - This is the port that will be executed under the command in "data"; when it is not set, it will request on itself.
  - ITS API port is set at 34001
- data : Command List
  - This consists of array (list[]), and it is an array of commands.

data

리스트 형식의 JSON으로 다양한 명령문의 배열이다.

List type of JSON, and it is an array of commands.

```
{
    "data": [
        { command },
        { command },
        ...
    ]
}
```

command

gpio, audio, camera, messenger, custom, system ...

```
{
    "data": [
        {
            "command": {
                "name": value,
                ...
            },
            ...
        }
    ]
}

{
    "data": [
        {
            "command": { "name": value, ... }
        }
    ]
}
```

options

server, keycode, debug

```
{
    "data": [
        {
            "command": { "name": value, ... },
            "server": { host:addr, port:no. },
            "keyCode": sha256,
            "debug": true / false
        }
    ]
}
```

- server(Option) : remote IP and Port
  - Command단위 명령을 원격(IP:Port)으로 실행 요청한는 기능이다.
  - server": { "host":"192.168.0.100","port":"34001" }
- keyCode(Option) : keySource
  - 설정: API > Setup > API Key > keySource 값이 설정 되어 있으면 요청되는 모든 명령문에 API > Setup > API Key > keyCode(License) 값이 동반되어야 하며 그 값이 일치되어야만 명령이 실행된다.
- debug(Option) : true / false
  - 오류검증이 필요할때 사용되며 참이면 실행에 따른 결과를 반환한다.

- server(Option) : remote IP and Port
  - This is a function to request execution of commands remotely (IP:port)
  - server": { "host":"192.168.0.100","port":"34001" }
- keyCode(Option) : keySource
  - The command will execute only if keySource value (setting: API > setup > API Key > keySource) matches with keyCode(License) value in API > Setup > API Key > keyCode(License).
- debug(Option) : true / false
  - This command is needed for debugging, and if it is true, it will return value when the program is executed.

Complexed Action Script

단일명령을 콤마로 구분하여 복수의 명령을 동시에 요청할수 있다.

명령문수는 이론적으로 무제한으로 복수명령문의 실행 순서는 정열순으로 진행된다.

각각의 명령문은 subprocess 및 threading 기반으로 실행되어 이미 요청된 명령은 사용자및 시스템으로부터 독립적으로 행동한다.

Complex action script can be called with multiple simple action scripts separated with commas.

The number of scripts can theoretically be infinite, and the complex action script is executed by the order the script is in.

Each script is executed based on its subprocess and threading, and the requested script is independent of the user or the system.

Format

```
{
    "data": [
        { command_01 },
        { command_02 },
        { command_03 },
        ...
    ]
}
```

Commands

릴레이제어(GPIO), 방송(Audio), 스넵샷(Camera), 문자메시지(Messenger), 네트워크통신(Custom), 시스템제어(System), 원격실행(Server), 디버그(Debug)모드등 다양한 기능을 포함한다.

특히 해시코드를 통한 보안키(KeyCode)명령과 센서전원제어, 핼스체크 및 하트비트 발생기능도 포함한다.

A variety of functions are included such as relay control (GPIO), broadcast (Audio), snapshot (Camera), text message (Messager), network communication (Custom), system control (System), remote execution (Server), debug (Debug).

Specifically, it also includes security code (KeyCode) commands through hashcode, sensor power control, health check, and heartbeat occurence.

GPIO

ITS Device는 4개의 Relay Out과 1개의 12v 전원을 제어 한다.

ITS Device controls 4 Relay Outs and 1 12V power.

> **Table**
>
Relay	**R01**	**R02**	**R03**	**R04**
ID	io09	io10	io11	io12

Power	**P01**
ID	pw01

Format

```
{
    "data":[
        {
            "gpio":{
                "status":"status",
                "id":"id",
                "hold":"Second",
                "count":"Number",
                "interval":"Second"
            }
        }
    ]
}
```

- status :
  - '0:Off', '1:On', '2:Toggle', '3:Status', '7:Status Power', '8:Status Relay and Sensor', '9:Status All'
- id : Relay ID
  - Table 참조
- hold (float) :
  - 대기시간 후 반전
  - 0인 경우 상태 유지
- count (int) :
  - 센서이벤트(감지) 횟수
- interval (float) :
  - 센서이벤트(감지) 대기시간

- status :
  - '0:Off', '1:On', '2:Toggle', '3:Status', '7:Status Power', '8:Status Relay and Sensor', '9:Status All'
- id : Relay ID
  - See Table
- hold (float) :
  - Reversal after waiting time
  - Keep the state when it's 0
- count (int) :
  - sensor event (detect) number
- interval (float) :
  - sensor event (detect) duration

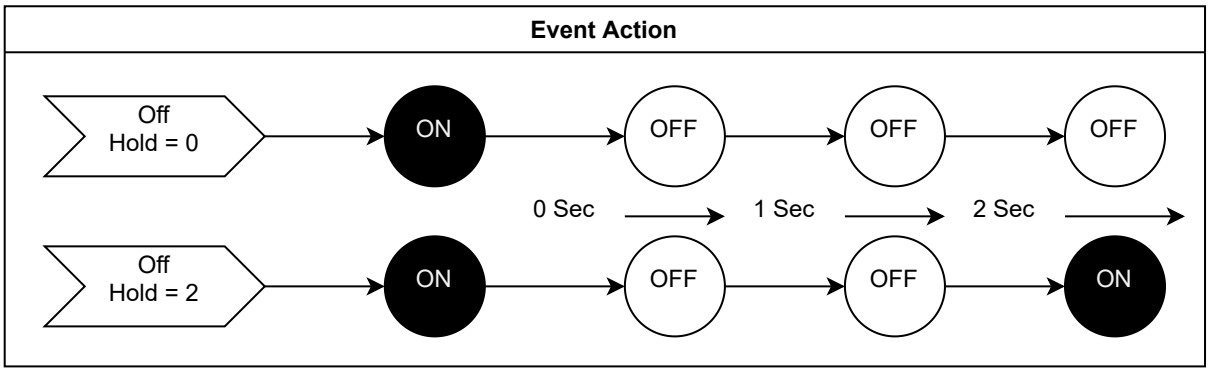
Hold

Hold는 출력단(Relay)의 기능이다.

- 요청된 상태를 Hold 시간만큼 유지한후 반전 한다.
- 0초로 선언된 경우엔 요청된 상태를 그대로 유지한다.

Hols is a function of output stage (Relay).

- Reveral occurs after maintaining the requested state as much as Hold time.
- If 0 seconds is declared, the requested state is maintained.



Example: 이벤트 발생시 릴레이 R01 Off

Example: when an event occurs, relay R01 off

```
{
    "data":[
        {
            "gpio":{
                "status":"0",
                "id":"io09",
                "hold":"0"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":{"status":"0","id":"io09","hold":"0"}}]
```

Example: 이벤트 발생시 릴레이 R01 On

Example: when an event occurs, relay R01 on

```
{
    "data":[
        {
            "gpio":{
                "status":"1",
                "id":"io09",
                "hold":"0"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":{"status":"1","id":"io09","hold":"0"}}]
```

Example: 이벤트 발생시 릴레이 R01 Toggle(즉각반응)

Example: when an event occurs, relay R01 Toggle (immediate reaction)

```
{
    "data":[
        {
            "gpio":{
                "status":"2",
                "id":"io09",
                "hold":"0"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":{"status":"2","id":"io09","hold":"0"}}]
```

Example: 이벤트 발생시 릴레이 R01 1.5초간 On 후 Off

Example: when an event occurs, relay R01 On for 1.5 seconds then off

```
{
    "data":[
        {
            "gpio":{
                "status":"1",
                "id":"io09",
                "hold":"1.5"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":{"status":"0","id":"io09","hold":"1.5"}}]
```

Example: 이벤트 발생시 릴레이 R01 0.5초간 반전 유지 후 재반전

Example: when an event occurs, relay R01 reversal maintained for 0.5 seconds then reversal again

```
{
    "data":[
        {
            "gpio":{
                "status":"2",
                "id":"io09",
                "hold":"0.5"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":{"status":"2","id":"io09","hold":"0.5"}}]
```

Count, Interval 간의 상관관계

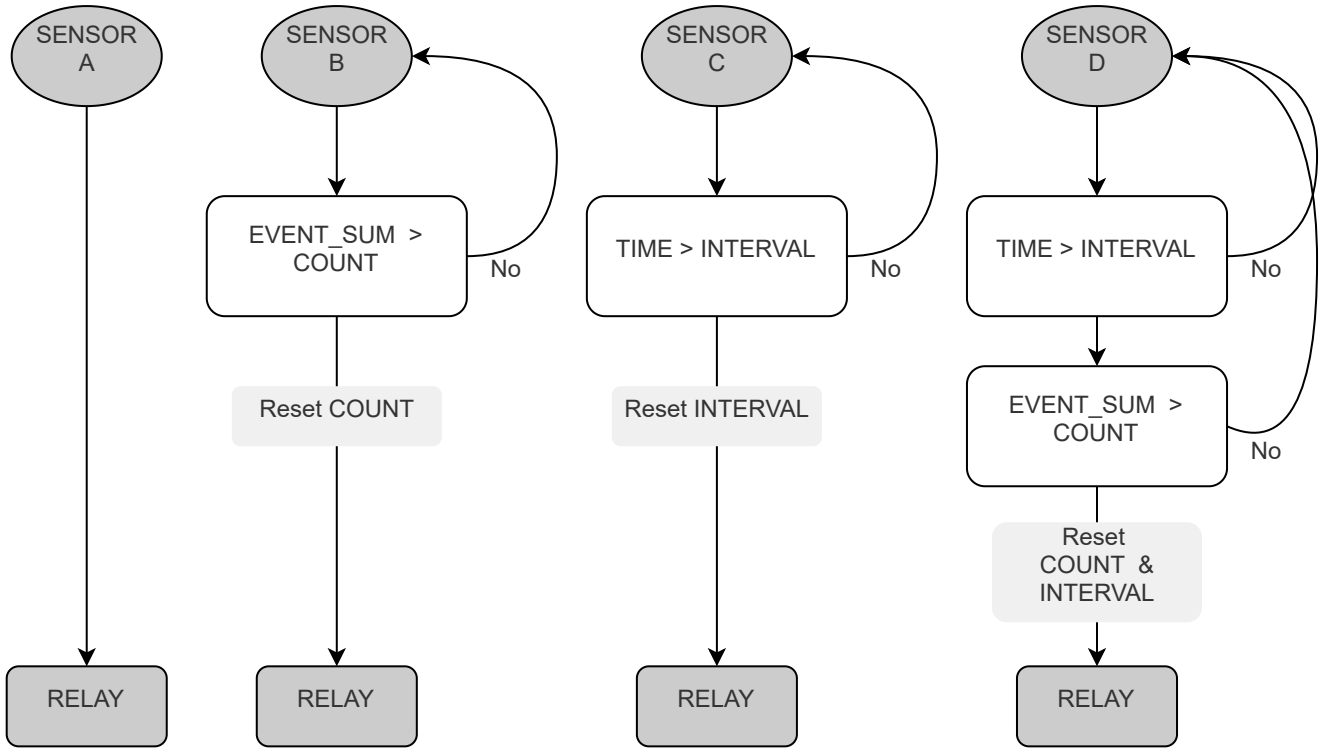
Correlation with Count and Interval

카운터와 인터벌은 입력단의 기능이다.

- A: 일반적으로 센서에서 감지신호가 오면 감지 유무에 따라 명령을 실행한다.
- B: Count(횟수)가 선언되면
    - 선언된 횟수 이상의 신호가 들어올 경우 명령을 실행한다.
    - 이후 카운터는 리셋된다.
- C: Interval(시간)이 선언되면
    - 감지되는 신호를 선언된 시간까지 무시하고 선언시간을 넘긴 신호가 감지될때 명령을 실행한다.
    - 이후 인터벌은 리셋된다.
- D: Count와 Interval이 같이 선언되면
    - Interval 시간안에 Count 횟수를 넘기면 남은 인터벌 시간동안 매 회 명령을 실행한다.
    - 인터벌을 주기로 카운터는 리셋된다.

Counter and interval are functions of the input stage.

- A: In general, when a detection signal is received from the sensor, the command is executed according to the presence or absence of detection.
- B: When Count is declared
    - Execute the command when more than the declared number of signals are received.
    - After that, the counter is reset.
- C: When Interval (time) is declared
    - Ignores the detected signal until the declared time and executes the command when the signal that exceeds the declared time is detected.
    - After that, the interval is reset.
- D: When Count and Interval are declared together
    - If the number of counts is exceeded within the interval time, the command is executed every time during the remaining interval time.
    - The counter is reset at intervals.


SENSOR A
SENSOR B
EVENT_SUM > COUNT
No
Reset COUNT
SENSOR C
TIME > INTERVAL
No
Reset INTERVAL
SENSOR D
TIME > INTERVAL
No
EVENT_SUM > COUNT
No
Reset COUNT & INTERVAL
RELAY
RELAY
RELAY
RELAY

Count 기능

센서 이벤트 횟수가 설정값 이상일때 명령실행

이벤트가 연속으로 감지될때는 카운터 설정값을 주기로 반복됨

Command execution when the number of sensor events exceeds the set value

When an event is continuously detected, the counter set value is repeated periodically.

Example: 이벤트 발생이 3회이상 발생시 릴레이 R01 On

Example: when an event occurs 3 times or more, relay R01 on

```
{
    "data":[
        {
            "gpio":{
                "status":"1",
                "id":"io09",
                "hold":"0",
                "count":"3"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":
{"status":"1","id":"io09","hold":"0","count":"3"}}]
```

Example: 이벤트 발생이 10회이상 발생시 릴레이 R01 반전

Example: when an event occurs 10 times or more, relay R01 reversal

```
{
    "data":[
        {
            "gpio":{
                "status":"2",
                "id":"io09",
                "hold":"0",
                "count":"10"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":
{"status":"2","id":"io09","hold":"0","count":"10"}}]
```

Interval 기능

최초 이벤트 발생후 인터벌시간 이내에 이벤트는 무시되고 이후 발생시 명령실행

이벤트가 연속으로 감지될때는 인터벌 설정값을 주기로 반복됨

The event is ignored within the interval time after the first event occurs, and the command is executed when subsequent occurrences occur.

When an event is continuously detected, the interval set value is repeated periodically.

Example: 이벤트 발생이 1초이상 연속으로 발생시 릴레이 R01 On

Example: when an event occurs consecutively for 1 second or more, relay R01 on

```
{
    "data": [
        {
            "gpio": {
                "status": "1",
                "id": "io09",
                "hold": "0",
                "interval": "1"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":
{"status":"1","id":"io09","hold":"0","interval":"1"}}]
```

Example: 이벤트 발생이 10초이상 연속으로 발생시 릴레이 R01 반전

Example: when an event occurs consecutively for 10 seconds or more, relay R01 reversal

```
{
    "data": [
        {
            "gpio": {
                "status": "2",
                "id": "io09",
                "hold": "0",
                "interval": "10"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":
{"status":"2","id":"io09","hold":"0","interval":"10"}}]
```

Count 와 Interval 조합 기능

인터벌 시간이내 카운터 설정값 이상 감지시 명령실행

이때 명령실행은 인터벌 종료시간 까지 매 회 발생

인터벌이 종료 되면 카운터도 초기화됨

Command execution when an event occurs over then counter setting within interval time

At this time, command execution occurs every time until the end of the interval.

When the interval ends, the counter is also reset.

Example: 이벤트 발생이 1초이내에 연속으로 3회이상 발생시 릴레이 R01 On

Example: when an event occurs 3 times consecutively within 1 second, relay R01 on

```
{
    "data": [
        {
            "gpio": {
                "status": "1",
                "id": "io09",
                "hold": "0",
                "count":"3",
                "interval": "1"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":
{"status":"1","id":"io09","hold":"0","count":"3","interval":"1"}}]
```

> Count나 Interval만 선언된 경우 각각의 설정값을 주기로 명령이 실행되지만 Count와 Interval이 동시에 설정된경우 종료시간 까지 연속적으로 명령이 실행된다.

> If only Count or Interval is declared, the command is executed at the cycle of each set value, but if Count and Interval are set at the same time, the command is continuously executed until the end time.

Audio

ITS Device Audio 제어 한다.

- 방송중 연속적인 재생 요청은 무시된다.
- 제공되는 오디오 플레이어는 omxplayer와 mplayer가 있다.
- 플레이어 특성에 따라 볼륨레벨이 다를 수 있다.
- omxplayer 선택시 반복(loop)기능은 무시된다.
- mplayer는 omxplayer보다 재생 직전 초기화 시간이 길다.
- 플레이어에 따라 불륨레벨에 차이도 참고 해야한다.
- 음원은 사전정의된 순서로 재생 가능한 음원과 사용자 업로드 재생 기능이 있다.
- 설정: API > Config > File Manager > Audio Manager
- 사용자 업로드 한 음원을 이웃하는 ITS API(**server 기능** 참조)로 재생시 네트워크 복사후 재생된다.

Controls ITS Device Audio.

- Continuous playback requests during broadcast are ignored.
- The provided audio players are omxplayer and mplayer.
- Volume level may vary depending on player characteristics.
- When omxplayer is selected, the loop function is ignored.
- mplayer has a longer initialization time just before playback than omxplayer.
- You should also take note of the difference in volume level depending on the player.
- The sound source has a sound source that can be played in a predefined order and a user upload play function.
- Settings: API > Config > File Manager > Audio Manager
- When playing the sound source uploaded by the user with the neighboring ITS API (refer to **server function**), it is played after network copy.

Format

```
{
    "data":[
        {
            "audio":{
                "source":"Internal(No./Full Path), External(URI)",
                "volume":"volume(%): 0 - 100",
                "loop":"0 ~ Int"
            }
        }
    ]
}
```

- source: Internal(No/Full Path), External(URI)
- volume(%): 0 - 100
- loop: 0 ~ Int.

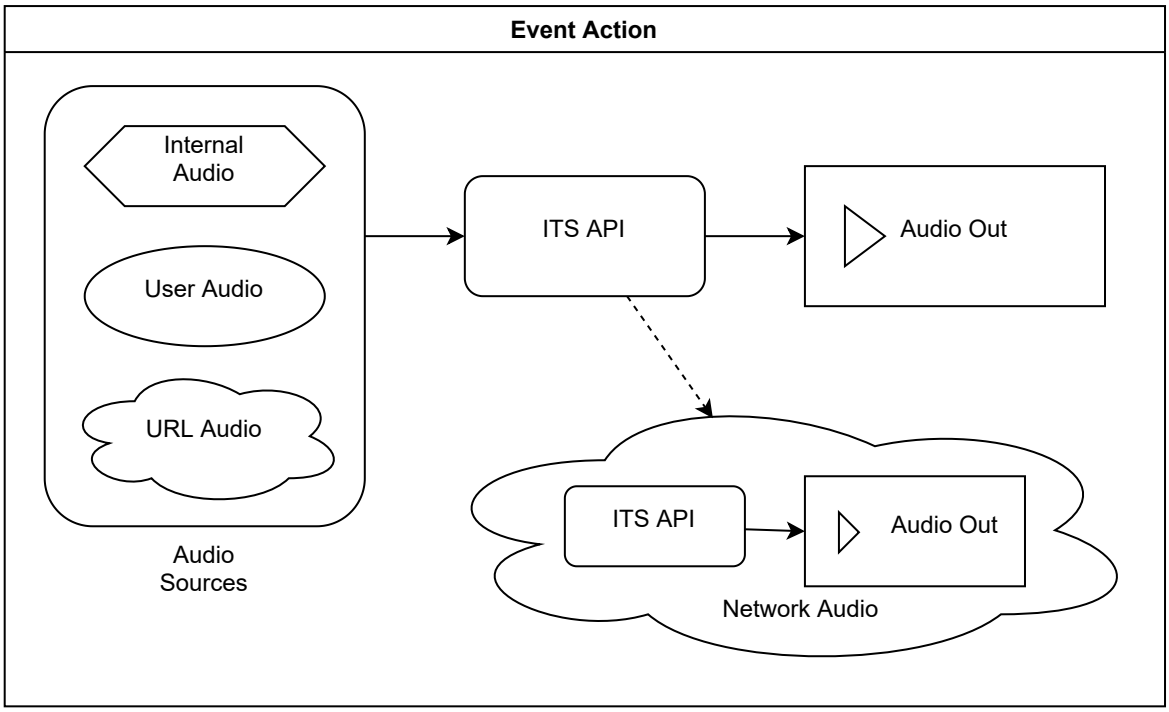
Audio System

모든 오디오형식은 MP3이다.

- Internal Audio : 빠른 응답속도
  - 사전에 등록된 음원으로 번호로 호출 가능 하다.
  - 환경설정(Config > File Manager > Audio Manager)을 통해 검색 및 등록 가능 하다.
  - 순서는 등록된 파일명의 정렬순이다.
- User Audio : 음원에 따라 재생시간 지연가능
  - 사용자가 업로드한 오디오의 전체경로(Full Path)를 지정해야 한다.
- URL Audio
  - 네트워크(또는 웹)상에 있는 오디오파일의 URL을 지정해야 한다.
- Network Audio : 재생 속도가 상대적으로 느림
  - 이웃하는 스피커로 오디오 재생이 필요할때 관련 디바이스로 명령문 전송이 가능하다.
  - 이는 브로드캐스팅 기능으로 재생속도를 감안해 Internal Audio 재생을 권장 한다.

All audio formats are MP3.

- Internal Audio: Fast response speed
  - It is possible to call by number with a sound source registered in advance.
  - Search and registration are possible through the environment setting (Config > File Manager > Audio Manager).
  - The order is the sort order of the registered file names.
- User Audio: Playback time can be delayed depending on the sound source
  - The full path of the audio uploaded by the user must be specified.
- URL Audio
  - You must specify the URL of the audio file on the network (or web).
- Network Audio : Relatively slow playback speed
  - When it is necessary to play audio to a neighboring speaker, it is possible to send a command to the relevant device.
  - This is a broadcasting function. Considering the playback speed, internal audio playback is recommended.


Event Action
Internal Audio
User Audio
URL Audio
Audio Sources
ITS API
Audio Out
ITS API
Audio Out
Network Audio

Example: 사전정의된 음원 4번을 40%의 볼륨으로 플레이 요청

Example: Request to play predefined sound source 4 at 40% volume

```
{
    "data":[
        {
            "audio":{
                "source":"4",
                "volume":"40",
                "loop":"0"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"audio":{"source":"4","volume":"40","loop":"0"}}]
```

Example: 사용자 오디오 Warning을 100%의 볼륨으로 10번 재생 요청

Example: Request to play user audio warning 10 times at 100% volume

```
{
    "data":[
        {
            "audio":{
                "source":"./audio/Warning.mp3",
                "volume":"100",
                "loop":"10"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"audio":
{"source":"./audio/Warning.mp3","volume":"100","loop":"10"}}]
```

Example: 웹 오디오 nocturne을 70%의 볼륨으로 2번 플레이 요청

Example: Request to play web audio nocturne 2 times at 70% volume

```
{
    "data":[
        {
            "audio":{
                "source":"https://www.mfiles.co.uk/mp3-downloads/chopin-nocturne-
op9-no2.mp3",
                "volume":"70",
                "loop":"1"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"audio":{"source":"https://www.mfiles.co.uk/mp3-
downloads/chopin-nocturne-op9-no2.mp3","volume":"70","loop":"1"}}]
```

Talk

마이크를 통한 음성을 원격으로 실시간 전송하는 기능이다.

- 서버 클라이언트간 파이프라인(Redirection)을 연결하는 방식으로 1:n은 불가능하다.
- [speaking]과 [listening]을 개별 및 동시 실행이 가능하다
- 동시에 실행하는 경우 상대방과의 실시간 대화가 가능하게 된다.
- [speaking]은 마이크를 통한 자신의 음성을 원격으로 송신하는 방식이다.
- [listening]은 원격의 마이크를 구동시켜 음성을 수신하는 방식이다.
- [disconnect]는 실행되고 있는 [speaking] 또는 [listening]프로세서를 죽인다.
- 구동되는 processor 종류는 arecord, oggenc, sshpass, mplayer 이다.
- 종료는 threading기능을 이용해 버퍼링시간(3~4초)을 대기한 후 관련프로세서를 죽인다.
- 접속을 위한 클라이언트 IP는 환경설정을 통해 사전등록 되어야 한다.

This is a function that remotely transmits voice through a microphone in real time.

- 1:n is not possible as it connects the pipeline (redirection) between server and client.
- [speaking] and [listening] can be executed individually or simultaneously.
- When executing simulataneouly, real-time conversation with the other party is possible.
- [speaking] is a method of remotely transmitting one's own voice through a microphone.
- [listening] is a method of receiving voice by driving a remote microphone.
- [disconnect] kills the running [speaking] or [listening] processor.
- The types of processors running are arecord, oggenc, sshpass, and mplayer.
- Termination uses the threading function to wait for the buffering time (3~4 seconds) and then kills the related processor.
- Client IP for connection must be pre-registered through environment setting.

Format

```
{
    "data":[
        {
            "talk":{
                "command":"",
                "remoteIP":""
            }
        }
    ]
}
```

- command :
  - speaking, listening, disconnect
- remoteIP : Target IP

Example: speaking

Local System의 마이크를 통한 Audio Streamming을 원격으로 송신한다. Remotely transmits Audio Streaming through the microphone of the Local System.

```
{
    "data":[
        {
            "talk":{
                "command":"speaking",
                "remoteIP":"ip_address"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"talk":
{"command":"speaking","remoteIP":"192.168.0.2"},"debug":true}]
```

Example: listening

Remote System의 마이크를 통한 Audio Streamming을 Local System으로 수신한다. Receives audio streaming through the microphone of the remote system to the local system.

```
{
    "data":[
        {
            "talk":{
                "command":"listening",
                "remoteIP":"ip_address"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"talk":
{"command":"listening","remoteIP":"192.168.0.2"},"debug":true}]
```

Example: disconnect

실행되고있는 관련 프로세서를 죽인다. Kill the associated processor that is running.

```
{
    "data":[
        {
            "talk":{
                "command":"disconnect",
                "remoteIP":"ip_address"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"talk":
{"command":"disconnect","remoteIP":"192.168.0.2"},"debug":true}]
```

Camera

이벤트 발생시점을 기준으로 이전과 이후 영상을 저장하는 기능이다.(Blackbox)

- ITS microDVR(mDVR)과 연동
- 설정: API > Setup > Camera(mDVR)
- CCTV의 스트리밍을 일정량(maxCntPrev)의 스넵샷으로 저장하는 기능
- 스넵샷을 maxCntPrev 횟수까지 저장하는 주기는 환경에 따라 시간이 변동적이다.
- CCTV의 성능에 따라 저장속도나 크기가 가변적임으로 상황에 따른 튜닝이 필요하다.
- 이미지는 http://its.ip/mDVR/ 에 저장되며 확인 가능 하다.
- 스트리밍 프로토클은 Local Camera(/dev/video0) 외에 RTSP, AVI, MJPG가 가능 하다.
- 웹캠인 경우 기본 640 X 480 초당 20장 정도를 일반적으로 저장 한다.
- 현재 가능한 기능은 footprint(mDVR) 이다.
- 그외 still_shot, motion_shot(due_time), list_shot(interval, count), download_shot 기능은 추후 기능

It is a function to save before and after images based on the time of event occurrence. (Blackbox)

- Interworking with ITS microDVR (mDVR)
- Settings: API > Setup > Camera (mDVR)
- Ability to save CCTV streaming as snapshots of a certain amount (maxCntPrev)
- The period of saving snapshots up to maxCntPrev times varies depending on the environment.
- Depending on the performance of the CCTV, the storage speed or size is variable, so tuning is required according to the situation.
- Images are saved in http://its.ip/mDVR/ and can be checked.
- Streaming protocol can be RTSP, AVI, MJPG in addition to Local Camera (/dev/video0).
- In case of Webcam, about 20 frames per second of 640 X 480 are generally saved.
- The currently available function is footprint (mDVR).
- Other still_shot, motion_shot(due_time), list_shot(interval, count), and download_shot functions will be available later.

Format

```
{
    "data":[
        {
            "camera":{
                "command":"",
                "value":""
            }
        }
    ]
}
```

- command :
  - footprint, still_shot, motion_shot, list_shot, download_shot
- value : Not Use(Current)

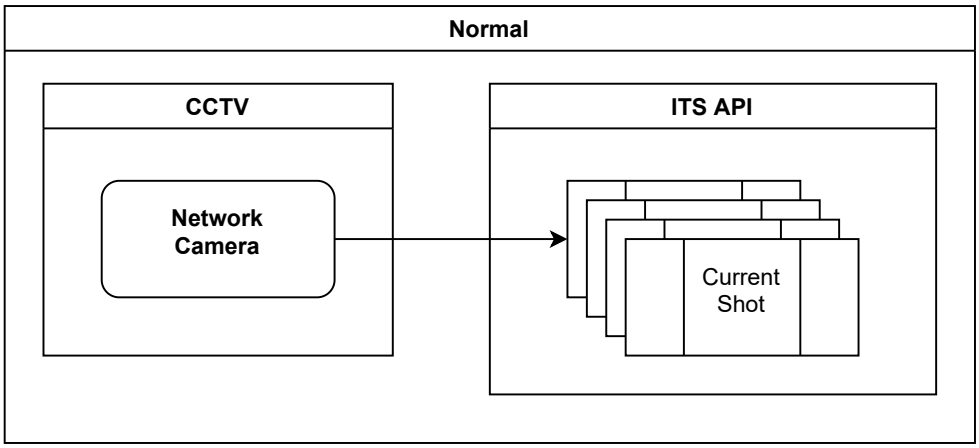
Footprint(Blackbox) System

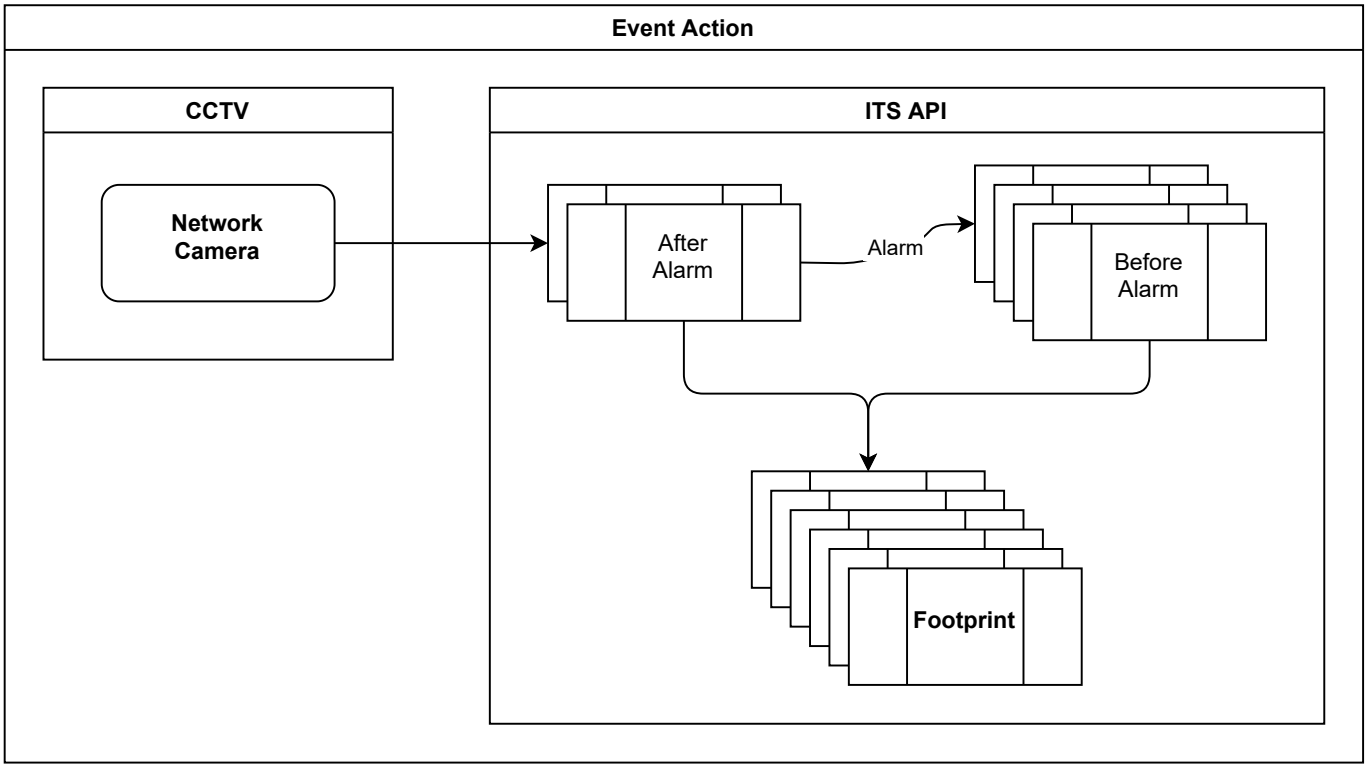
센서이벤트 발생 시점이전의 일정량(사전정의됨)과 시점이후의 일정량은 저장보관 하는 블랙박스 기능이다.

- Normal
  - 사전정의된 일정량의 스넵샷을 지속적으로 저장(FIFO)한다.
  - 설정량을 넘은 스냅샷을 버려서 설정량을 유지한다.
- Event
  - 센서이벤트 발생시 그 시점까지 저장된 스넵샷(cntPreShotMax:20)과 그 이후 일정시간(cntPostShotMax:10)의 스넵샷을 저장하는 기능이다.
  - 이기능은 이벤트 이전시점(cntPreShotMax) + 이벤트이후시간(cntPostShotMax) 동안 저장된 스넵샷이다.

It is a black box function that stores and stores a certain amount (pre-defined) before the sensor event occurs and a certain amount after the point of time.

- Normal
  - Continuously stores (FIFO) a predefined amount of snapshots.
  - The set amount is maintained by discarding the snapshots that exceed the set amount.
- Event
  - When a sensor event occurs, it is a function to save the snapshot (cntPreShotMax:20) saved up to that point and the snapshot for a certain time (cntPostShotMax:10) thereafter.
  - This function is a snapshot saved during the time before the event (cntPreShotMax) + the time after the event (cntPostShotMax).


Normal
CCTV
Network Camera
ITS API
Current Shot


Event Action
CCTV
Network Camera
ITS API
After Alarm
Alarm
Before Alarm
Footprint

Example: Camera footprint Test

```
{
    "data":[
        {
            "camera":{
                "command":"footprint",
                "value":""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"camera":{"command":"footprint","value":""}}]
```

- 저장된 자료 조회경로: http://ip_address/mDVR
- Saved data inquiry path: http://ip_address/mDVR

Trigger

인위적으로 센서포트를 트리거링 하는 기능이다.

- Soft Sensing (Hardware Event가 아닌 인라인 명령)
- 주의: Action Script 내에 자신을 Call하게되면 무한루프 오류 발생 우려
- System Command의 trigger_io와 유사한 기능
- No Reponse

This is a function that artificially triggers the sensor port.

- Soft Sensing (inline commands, not hardware events)
- Caution: If you call yourself within Action Script, you may get an infinite loop error.
- Function similar to trigger_io of System Command
- No response

Format

```
{
    "data":[
        {
            "trigger":{
                "id":"Sensor ID"
            }
        }
    ]
}
```

- Sensor ID: io01 ~ io08

Sensor	S01	S02	S03	S04	S05	S06	S07	S08
ID	io01	io02	io03	io04	io05	io06	io07	io08

Example: trigger

```
{
    "data":[
        {
            "trigger":{
                "id":"io02"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"trigger":{"id":"io02"},"debug":true}]
```

Messenger

본서비스를 통해 관리영역의 감지이벤트를 ITS API를 통해 문자로 통보 받는다.

- 텔레그램을 통한 SMS Report Service 사전작업
  - 설정: API > Telegram > Token, ChatID
  - Telegram Bot Token은 ITS API단에서 제공하는 키값
  - Group ChatID는 사용자가 만들어 제공해야하는 그룹ID
    - 취득방법: 관리자에게 문의

Through this service, the detection event of the management area is notified by text message through the ITS API.

- SMS Report Service pre-work through Telegram
  - Settings: API > Telegram > Token, ChatID
  - Telegram Bot Token is a key value provided by the ITS API
  - Group ChatID is a group ID that the user must create and provide.
    - How to obtain: Ask the manager

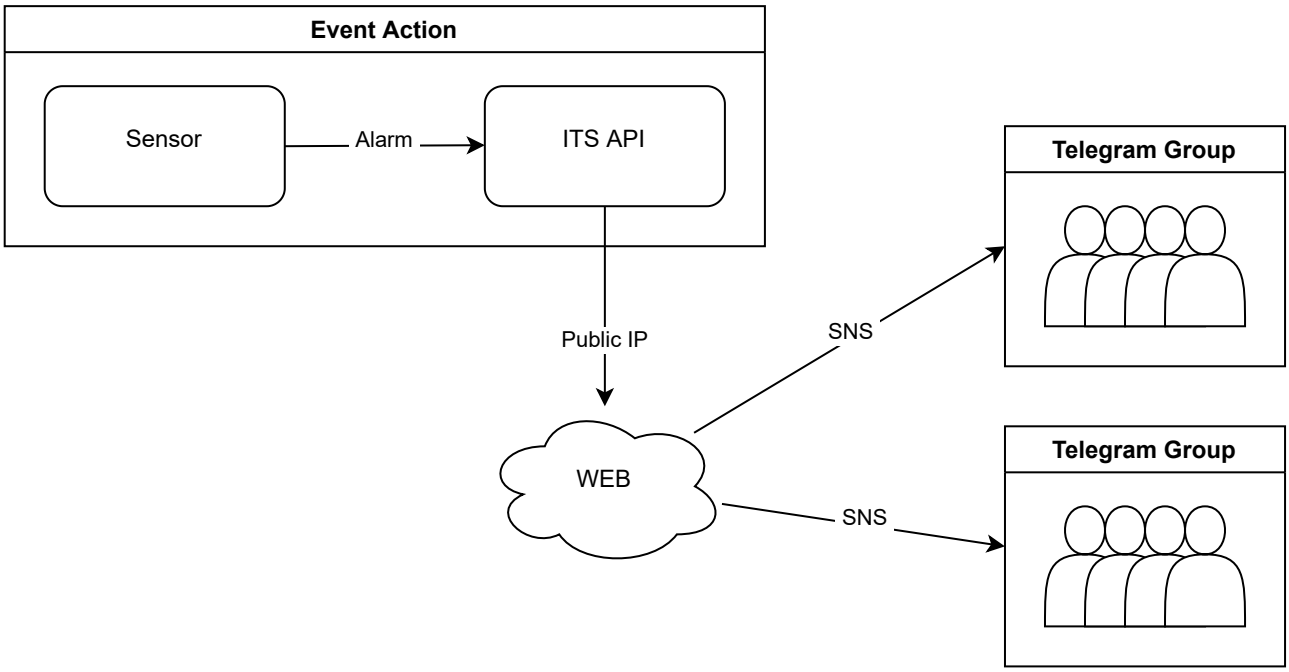
Format

```
{
    "data":[
        {
            "messenger":{
                "sendMessage":"any Message"
            }
        }
    ]
}
```

- sendMessage : 전송할 Message
- sendMessage : Message to send

SNS Messenger

- 텔레그램
- Telegram



Example: sendMessage

```
{
    "data":[
        {
            "messenger":{
                "sendMessage":"ITS API System Message"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"messenger": {"sendMessage": "http request
Message"}}]
```

```
echo '[{"messenger": {"sendMessage": "Commend Line Message"}}]' | nc ip_address
34001 -q 0
```

Custom

tcp_socket, http_get, http_post를 통한 네트워크상의 이기종 서버에 명령을 전송함

본 기능은 API가 제공되는 독립된 디바이스나 시스템의 직접제어가 가능하다.

예를 들면 IP Camera, IP Audio, Monitoring System, Factory Automation System, etc..

Sends commands to heterogeneous servers on the network through tcp_socket, http_get, and http_post

This function enables direct control of an independent device or system for which API is provided.

For example, IP Camera, IP Audio, Monitoring System, Factory Automation System, etc..

Format - tcp_socket

```
{
    "data":[
        {
            "custom":{
                "method":"tcp_socket",
                "isJson": true / false,
                "data": {}
            },
            "count": "0",
            "interval": "0"
        },
        "server": {
            "host": "server ip",
            "port": "server port"
        }
    ]
}
```

- method
- isJson : data type
- count
- interval
- server : Remote Server Info

Example: tcp_sockert - JSON

```
{
    "data": [
        {
            "custom": {
                "method": "tcp_socket",
                "isJson": true,
                "data": {
                    "id_01": "name_01",
                    "id_02": "name_02"
                },
                "count": "3",
                "interval": "3"
            },
            "server": {
                "host": "ip_address",
                "port": "34001"
            },
            "debug": true
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"custom":
{"method":"tcp_socket","isJson":true,"data":
{"id_01":"name_01","id_02":"name_02"},"count":"3","interval":"3"},"server":
{"host":"192.168.0.50","port":"34001"},"debug":true}]
```

Example: tcp_sockert - Non JSON

```
{
    "data": [
        {
            "custom": {
                "method": "tcp_socket",
                "isJson": false,
                "data": "id_01=name_01, id_02=name_02",
                "count": "0",
                "interval": "0"
            },
            "server": {
                "host": "ip_address",
                "port": "34001"
            },
            "debug": true
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"custom":
{"method":"tcp_socket","isJson":false,"data":"id_01=name_01,
id_02=name_02","count":"3","interval":"3"},"server":
{"host":"192.168.0.50","port":"34001"},"debug":true}]
```

Example: tcp_sockert - Non JSON - IMS protocol

```
{
    "data": [
        {
            "custom": {
                "method": "tcp_socket",
                "isJson": false,
                "data":
"id=g300t100_192_168_0_90_0012,name=R01,beep=1,shot=,latS=0.0,lngS=0.0,latE=0.0,ln
gE=0.0,count=1,block=0,status=1,msg=Active_Event",
                "count": "0",
                "interval": "0"
            },
            "server": {
                "host": "192.168.0.90",
                "port": "38087"
            },
            "debug": true
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"custom":
{"method":"tcp_socket","isJson":false,"data":"id=g300t100_192_168_0_90_0012,name=R
01,beep=1,shot=,latS=0.0,lngS=0.0,latE=0.0,lngE=0.0,count=1,block=0,status=1,msg=A
ctive_Event","count":"3","interval":"3"},"server":
{"host":"192.168.0.90","port":"38087"},"debug":true}]
```

Format - http_get/http_post

```
{
    "data":[
        {
            "custom":{
                "method":"http_get/http_post",
                "data": {
                    data
                    ...
                },
                "count": "0",
                "interval": "0"
            },
        },
        "server":{
            "url":"server url",
            "username":"",
            "password":""
        }
    ]
}
```

- method : http_get or http_post
- count
- interval
- server : Remote Server URL, Username, Password

Example: http_get

```
{
    "data":[
        {
            "custom":{
                "method":"http_get",
                "data":{
                    "id_01":"name_01",
                    "id_02":"name_02"
                },
                "count": "0",
                "interval": "0"
            },
            "server":{
                "url":"http://ip_address/api.php"
                "username":"",
                "password":""
            },
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"custom":{"method":"http_get","data":
{"id_01":"name_01","id_02":"name_02"},"count":"0","interval":"0"},"server":
{"url":"http://ip_address/api.php"},"debug":true}]
http://ip_address/api.php?api=[{"custom":{"method":"http_get","data":
{"id_01":"name_01","id_02":"name_02"},"count":"0","interval":"0"},"server":
{"url":"http://ip_address/api.php","username":"","password":""},"debug":true}]
http://192.168.0.50/api.php?api=[{"custom":{"method":"http_get","data":
{"id_01":"name_01","id_02":"name_02"},"count":"0","interval":"0"},"server":
{"url":"http://192.168.0.50/api.php","username":"","password":""},"debug":true}]
```

Example: http_post

```
{
    "data":[
        {
            "custom":{
                "method":"http_post",
                "data":{
                    "id_01":"name_01",
                    "id_02":"name_02"
                },
                "count": "3",
                "interval": "3"
            },
            "server":{
                "url":"http://ip_address/api.php",
                "username":"",
                "password":""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"custom":{"method":"http_post","data":
{"id_01":"name_01","id_02":"name_02"},"count":"3","interval":"3"},"server":
{"url":"http://ip_address/api.php"},"debug":true}]
http://192.168.0.50/api.php?api=[{"custom":{"method":"http_post","data":
{"id_01":"name_01","id_02":"name_02"},"count":"0","interval":"0"},"server":
{"url":"http://192.168.0.50/api.php","username":"","password":""},"debug":true}]
```

System

부가 명령어 기능이다.

- 주로 ITS API Device 관련 명령들로 디바이스명, 오디오, 센서, 시간 설정및 시그템 재부팅 이나 프로그램 재실행관련 명령으로 이루어져 있다.
- 사용자의 피드백에 따른 필요한 명령을 지속적으로 개발 중이다.

This is an additional command function.

- Commands related to ITS API Device are mainly composed of device name, audio, sensor, time setting, system reboot or program re-execution related commands.
- We are continuously developing necessary commands according to user feedback.

Format

```
{
    "data":[
        {
            "system":{
                "command":"[names]",
                "value": ""
            }
        }
    ]
}
```

- names :
    - sleep, set_name, get_name, set_time, get_time, stop_audio, list_audio, enable_audio, disable_audio, health_check, disable_io, enable_io, trigger_io, restart, reboot

Example: sleep

- 특정시간 대기
- Wait for a specific time

```
{
    "data":[
        {
            "system":{
                "command":"sleep",
                "value": "1.5"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"sleep","value":"1.5"},"debug":true}]

Return:
{
    "ip": "ip_address",
    "category": "system",
    "command": "sleep",
    "msg": "sleep 1sec"
}
```

Example: get_name

- 디바이스명 불러오기
- Get device name

```
{
    "data":[
        {
            "system":{
                "command":"get_name",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"get_name","value":""},"debug":true}]

Return:
{
    "ip": "ip_address",
    "category": "system",
    "command": "get_name",
    "msg": "Location"
}
```

Example: set_name

- 디바이스명 설정하기
- Set device name

```
{
    "data":[
        {
            "system":{
                "command":"set_name",
                "value": "New Title"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":{"command":"set_name","value":"New
Title"},"debug":true}]

Return:
{
    "ip": "ip_address",
    "category": "system",
    "command": "set_name",
    "msg": "New location name is New Title"
}
```

Example: get_time

- 디바이스 시간 불러오기
- Get device time

```
{
    "data":[
        {
            "system":{
                "command":"get_time",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"set_name","value":""},"debug":true}]

Return:
{
    "ip": "ip_address",
    "category": "system",
    "command": "set_name",
    "msg": "2022-06-29 22:02:52"
}
```

Example: set_time

- 디바이스 시간 설정하기
- Set device time

```
{
    "data":[
        {
            "system":{
                "command":"set_time",
                "value": "2021-10-18 10:12:40"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":{"command":"set_time","value":"2021-10-18
10:12:40"},"debug":true}]

Return:
{
    "category": "system",
    "ip": "ip_address",
    "command": "set_time",
    "msg": "Success set_time Mon 18 Oct 10:12:40 KST 2021\n"
}
```

Example: list_audio

- 내부음원 목록 반환
- Return the internal sound source list

```
{
    "data":[
        {
            "system":{
                "command":"list_audio",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"list_audio","value":""},"debug":true}]

Return:
{
    "category": "system",
    "ip": "ip_address",
    "command": "list_audio",
    "msg": [
        "Air_Horn.mp3",
        "Fire_Truck.mp3",
        "Industrial.mp3",
        "Siren.mp3",
        "Smoke.mp3",
        "Whistle.mp3"
    ]
}
```

Example: stop_audio

- 오디오 출력 중지
- Stop audio output

```
{
    "data":[
        {
            "system":{
                "command":"stop_audio",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"stop_audio","value":""},"debug":true}]

Return:
{
    "category": "system",
    "ip": "ip_address",
    "command": "stop_audio",
    "msg": "Success stop_audio"
}
```

Example: enable_audio

- 오디오 기능 활성
- Audio function active

```
{
    "data":[
        {
            "system":{
                "command":"enable_audio",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"enable_audio","value":""},"debug":true}]

Return:
{
    "category": "system",
    "ip": "ip_address",
    "command": "enable_audio",
    "msg": "Now audio is enabled"
}
```

Example: disable_audio

- 오디오 기능 비활성
- Disable audio function

```
{
    "data":[
        {
            "system":{
                "command":"disable_audio",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"disable_audio","value":""},"debug":true}]

Return:
{
    "category": "system",
    "ip": "ip_address",
    "command": "disable_audio",
    "msg": "Now audio is disabled"
}
```

Example: trigger_io

- Soft Sensing (Dry Contact이 아닌 인라인 명령)
- Soft Sensing (inline command, not dry contact)
- io01 ~ io08

Sensor	S01	S02	S03	S04	S05	S06	S07	S08
ID	io01	io02	io03	io04	io05	io06	io07	io08

- 주의: Action Script 내에 자신을 Call하게되면 무한루프 오류 발생
- Caution: Infinite loop error occurs if you call yourself in Action Script

```
{
    "data":[
        {
            "system":{
                "command":"trigger_io",
                "value": "io01"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"trigger_io","value":"io01"},"debug":true}]

Return:
{
    "category": "system",
    "ip": "ip_address",
    "command": "trigger_io",
    "msg": "trigger io01"
}
```

Example: health_check

- ITS API 시스템 정보 반환
- Return ITS API system information
- IP Address, CPU Temp., CPI Useage, Storage Useage, Memory Useage, Systeme Time, Last Start Time, Live Time, License Key

```
{
    "data":[
        {
            "system":{
                "command":"health_check",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"health_check","value":""},"debug":true}]

Return:
{
  "ip": "ip_address",
  "category": "system",
  "command": "health_check",
  "msg": {
    "cpuPcent": {
      "idle": "94.9",
      "system": "3.8",
      "user": "1.3"
    },
    "cpuTemp": 52.078,
    "diskGb": {
      "avail": "10G",
      "pcent": "28%",
      "size": "15G",
      "used": "3.8G"
    },
    "fixed": {
      "dateTime": "2022-06-29 23:04:19.697769",
      "diskSize": "15G",
      "execTime": "0:00:00.284039",
      "ioBoard": "ITS STD",
      "ipAddr": "ip_address",
      "lastStart": "2022-06-02 18:16:14",
      "licenseStatus": "Approved",
      "liveTime": "2347848.96",
      "macAddr": "0xb827eb2af733L",
      "noLicense": 2592000,
      "run": "SRF",
      "serialKey": "000000000f2af733",
      "systemTitle": "ecos"
    },
    "memUseKb": {
      "free": "73.7",
      "total": "924.2"
    }
  }
}
```

Example: enable_io

- GPIO 기능 활성
- GPIO function active
  - io01 ~ io08

nsor	S01	S02	S03	S04	S05	S06	S07	S08
io01	io02	io03	io04	io05	io06	io07	io08	

  - io01 ~ io08

lay	R01	R02	R03	R04
io09	io10	io11	io12	

  - pw01

wer	P01
pw01	

```
{
    "data":[
        {
            "system":{
                "command":"enable_io",
                "value": "io01"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"enable_io","value":"io01"},"debug":true}]

Return:
{
    "category": "system",
    "ip": "ip_address",
    "command": "enable_io",
    "msg": "Now io01 is enabled"
}
```

Example: disable_io

- GPIO 기능 비활성
- Disable GPIO function

```
{
    "data":[
        {
            "system":{
                "command":"disable_io",
                "value": "io01"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"disable_io","value":"io01"},"debug":true}]

Return:
{
    "category": "system",
    "ip": "ip_address",
    "command": "disable_io",
    "msg": "Now io01 is disabled"
}
```

Example: restart

- API 재실행
- API restart

```
{
    "data":[
        {
            "system":{
                "command":"restart",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"restart","value":""},"debug":true}]

No Return:
```

Example: reboot

- 디바이스 재부팅
- device reboot

```
{
    "data":[
        {
            "system":{
                "command":"reboot",
                "value": ""
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":
{"command":"reboot","value":""},"debug":true}]

No Return:
```

Debug

명령문에 결과갑을 반환한다.

Returns the result of the statement.

- debug(Option) : true or false

Example: Debug

```
{
    "data":[
        {
            "gpio":{
                "status":"2",
                "id":"io09",
                "hold":"0.6"
            },
            "debug":true
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":
{"status":"2","id":"io09","hold":"0.6"},"debug":true}]

Return:
{
    "ip": "ip_address",
    "category": "gpio",
    "status": "2",
    "response": {
        "io09": 0
    }
}
```

KeyCode

모든 명령문의 암호화(Encryption) 인증(Authentication) 기능

- 키(Keycode) 생성은 관리자 권한이며 생성후 모든 명령문에 적용됨
- 전송시 명령어 레벨에 [keyCode]항목에 첨부 (SHA256 - Hex)
- 키값의 위치는 API > Setup > API Key > keyCode(License)이다.

Encryption of all statements Authentication function

- Keycode generation is an administrator privilege and is applied to all statements after creation.
- Attached to the [keyCode] item at the command level during transmission (SHA256 - Hex)
- The location of the key value is API > Setup > API Key > keyCode(License).

Example: KeyCode

```
{
    "data":[
        {
            "gpio":{
                "status":"2",
                "id":"io09",
                "hold":"0.6"
            },
            "keyCode":"e3b0c44298fc1c...",
            "debug":true
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"gpio":
{"status":"2","id":"io09"},"keyCode":"e3b0c44298fc1c...","debug":true}]

Return:
{
    "ip": "ip_address",
    "category": "keyCode",
    "msg": "Missing keyCode value"
}
```

Complexed Script

하나 이상의 명령을 동시에 요청할수 있다. 이는 콤마로 구분 명령문의 배열로 조합수는 무제한이다.

이벤트 발생시 주변 환경에 따라 경고 또는 안내방송과 동시에 경광등을 조작해야 하는등 복수의 작업이 필요한 때 사용한다.

또는 관제시스템에 알람을 보냄과 동시에 관련자에게 문자전송이나 Camera 제어등 다양한 응용이 가능 하다.

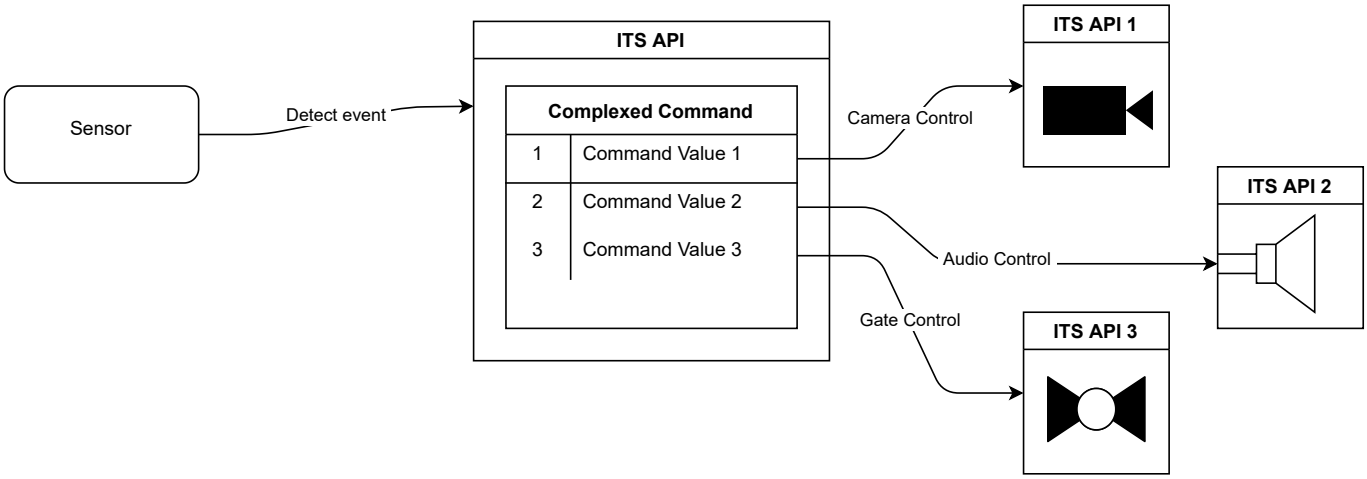
- 단일명령외 복합명령(Complexed Action Script) 사용시 중복되는 Debug mode 선언은 권장하지 않는다.

You can request more than one command at the same time. This is an array of comma-separated statements, and the number of combinations is unlimited.

When an event occurs, it is used when multiple tasks are required, such as to operate a warning light at the same time as a warning or announcement depending on the surrounding environment.

Alternatively, various applications such as sending an alarm to the monitering system and sending text messages or camera control to related parties are possible.

- It is not recommended to declare duplicate Debug mode when using Complexed Action Script other than a single command.



Example: 이벤트 발생시 릴레이 n초 대기 후 Toggle

Example: Toggle after waiting n seconds for relay when an event occurs

```
{
    "data": [
        {
            "system":{
                "command":"sleep",
                "value":"1.0"
            }
        },
        {
            "gpio":{
                "status":"2",
                "id":"io09",
                "hold":"0.1"
            }
        }
    ]
}
```

```
http://ip_address/api.php?api=[{"system":{"command":"sleep","value":"1.0"}},
{"gpio":{"status":"2","id":"io09","hold":"0.1"}},{"system":
{"command":"sleep","value":"1.0"}},{"gpio":
{"status":"2","id":"io09","hold":"0.1"},"debug":true}]

Return: 1초후에 반환값이 표시된다.
Return: The return value is displayed after 1 second.
{
    "ip": "ip_address",
    "category": "gpio",
    "status": "2",
    "response": {
        "io09": 0
    }
}
```

Example: 이벤트 발생시 릴레이(R01), 릴레이(R02) 동시 반전, n초 후 복귀

Example: Relay (R01) and relay (R02) simultaneously invert when an event occurs, return after n seconds

- http://ip_address/api.php?api=[{"gpio":{"status":"2","id":"io09","hold":"1"}},{"gpio":{"status":"2","id":"io10","hold":"1"}}]

Example: 이벤트 발생시 릴레이(R01) 반전, n초 후 릴레이(R02) 반전

Example: Relay (R01) reversed when an event occurs, relay (R02) reversed after n seconds

- http://ip_address/api.php?api=[{"gpio":{"status":"2","id":"io09","hold":"0.1"},"debug":true},{"system":{"command":"sleep","value":"1.0"}},{"gpio":{"status":"2","id":"io10","hold":"0.1"},"debug":true}]

Example: 단일 이벤트 릴레이(R01, R0, R03, R04) 전체 반전

Example: Single event relay (R01, R0, R03, R04) full inversion

- http://ip_address/api.php?api=[{"gpio":{"status":"2","id":"io09","hold":"0.2"}},{"gpio":{"status":"2","id":"io10","hold":"0.4"}},{"gpio":{"status":"2","id":"io11","hold":"0.6"}},{"gpio":{"status":"2","id":"io12","hold":"0.8"}}]

Example: 재생되는 오디오를 n초후, 정지시키고 새 음원 재생

Example: Stop playing audio after n seconds and play a new sound source

- http://ip_address/api.php?api=[{"system":{"command":"sleep","value":"2"},"debug":true},{"system":{"command":"stop_audio","value":""},"debug":true},{"audio":{"source":"5","volume":"40","loop":"0"},"debug":true}]
-

Example: 재생되는 오디오를 즉시 정지시키고 새 음원 재생(**비상경보**)

Example: Immediately stop the playing audio and play a new sound source (**Emergency Alarm**)

- http://ip_address/api.php?api=[{"system":{"command":"stop_audio","value":""},"debug":true},{"audio":{"source":"5","volume":"40","loop":"0"},"debug":true}]

Server

- Server 기능을 통해 각기다른 API(ITS)로 명령 요청
- Master와 Sub("server"명 소속)로 구분

- Request commands with different APIs (ITS) through the Server function
- Divided into Master and Sub (belonging to "server" name)

Format

```
{
    "host":"master ip",
    "port":"master port",
    "data":[
        {
            "gpio":{
                ...
            },
            "server":{
                "host":"sub ip",
                "port":"sub port"
            }
        }
    ]
}
```

- Master Server측에 Data를 전송
- Data측에서 Sub Server측에 GPIO 명령을 요청 한다.
- Transmit data to Master Server
- The data side requests a GPIO command from the sub server side.

Example: Server

```
{
    "host":"master ip",
    "port":"master port",
    "data":[
        {
            "gpio":{
                "status":"2",
                "id":"io12",
                "hold":"0.6",
                "interval":"4"
            },
            "server":{
                "host":"A_ip_address",
                "port":"34001"
            },
            "debug":true
        },
        {
            "audio":{
                "source":"5",
                "volume":"40",
                "loop":"0"
            },
            "server":{
                "host":"B_ip_address",
                "port":"34001"
            },
            "debug":true
        }
    ]
}
```

- 설명: 마스터서버가 모든 명령을 일괄로 접수한다.
- 접수된 명령을 슬레이브 서버 A_ip_address에는 gpio 명령을 요청하고
- 서버 B_ip_address에는 audio 명령을 요청한다.

- Description: The master server receives all commands at once.
- The received command requests the gpio command to the slave server A_ip_address and
- The audio command is requested from the server B_ip_address.

```
http://ip_address/api.php?api=[{"gpio":
{"status":"2","id":"io12","hold":"0.6","interval":"4"},"server":
{"host":"192.168.0.50","port":"34001"},"debug":true},{"audio":
{"source":"5","volume":"40","loop":"0"},"server":
{"host":"192.168.0.80","port":"34001"},"debug":true}]

Return:
- Ignore "SyntaxError: JSON.parse ..."
{
    "ip": "192.168.0.50",
    "category": "gpio",
    "status": "2",
    "response": {
        "io12": 0
}
}{
    "ip": "192.168.0.50",
    "category": "audio",
    "response": {
        "sent": "5"
    }
}
```

Example: CLI Test

이상의 모든 명령은 CLI에서도 실행 가능하다. 다음의 예를 참조

All of the above commands can also be executed from the CLI. See the following example

```
$ echo '[{"gpio":{"status":"2","id":"io12","hold":"0.6","interval":"4"},"server":
{"host":"192.168.0.50","port":"34001"},"debug":true},{"audio":
{"source":"5","volume":"40","loop":"0"},"server":
{"host":"192.168.0.80","port":"34001"},"debug":true}]' | nc 192.168.0.70 34001 -q
0
```

- 설명(CLI Test): 서버 A_ip_address에는 gpio 명령은 보내고 서버 B_ip_address에는 audio 명령을 실행하는 예 이다.
- Description (CLI Test): This is an example of sending the gpio command to the server A_ip_address and executing the audio command to the server B_ip_address.

Example: CLI vs Inline vs URI

다음은 CLI명령과 인라인 명령(Inline command), 브라우저(URI)요청간 명령어 차이로 3가지 모두 같은 명령문이다.

- 인라인 명령(Inline command)

The following is the command difference between CLI command, inline command, and browser (URI) request. All three commands are the same.

- Inline command

```
{
    "data":[
        {
            "gpio":{
                "status":"2",
                "id":"io09",
                "hold":"0"
            }
        }
    ]
}
```

- CLI

```
$ echo '[{"gpio":{"status":"2","id":"io12","hold":"0"}}]' | nc ip_address 34001 -q
0

<div style="page-break-after: always"></div>
```

- 브라우저(URI)
- browser(URI)

```
http://192.168.0.50/api.php?api=[{"gpio":{"status":"2","id":"io12","hold":"0"}}]
```

Scheduler(Crontab)

콘솔 윈도 내에 알람 설정을 실행 한다.

본 기능은 리눅스 Crontab 룰을 따른다.

실행명령(스크립트)은 기존 센서룰과 같다.

설정된 명령문은 **프로그램 재실행 시 적용** 된다.

최소 1분단위 설정 가능하다.

오류가 발생된 명령은 무시 한다.

Execute the alarm setting in the console window.

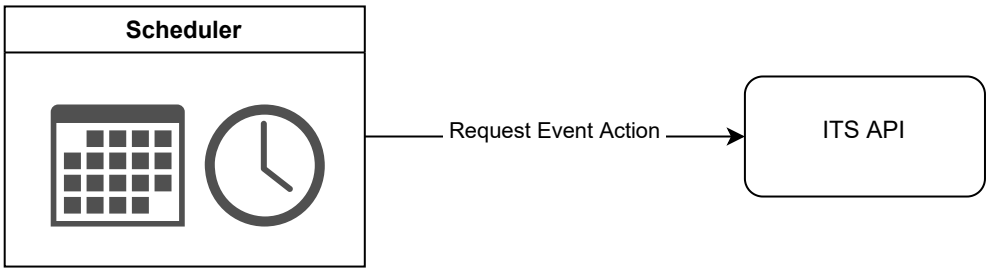
This function follows Linux Crontab rules.

The execution command (script) is the same as the existing sensor rule.

The set statement is applied when re-executing the program.

It can be set in units of at least 1 minute.

The command in which an error occurred is ignored.



```
* * * * * Commands
┬ ┬ ┬ ┬ ┬
│ │ │ │ │
│ │ │ │ │
│ │ │ │ └──────── Weeks (0 - 6) (0:sun, 1:mon, 2:tue, ..., 6:sat)
│ │ │ └────────── Months (1 - 12)
│ │ └──────────── Days (1 - 31)
│ └────────────── Hours (0 - 23)
└──────────────── Minutes (0 - 59)
```

Example: Scheduler Time Set

Time Set	Comment
* * * * *	매 분 마다
45 5 * * 5	매 주 금요일 오전 5시 45분에
0,20,40 * * * *	매 일 매 시간 0분, 20분, 40분에
0-30 1 * * *	매 일 1시 0분 부터 30분까지 매 분 마다
*/10 * * * *	매 10분 마다
*/10 2,3,4 5-6 * *	5일, 6일간 2,3,4시 대 매 10분 마다

Time Set	Comment
* * * * *	every minute
45 5 * * 5	Every Friday at 5:45 am
0,20,40 * * * *	Every day every hour at 0 minutes, 20 minutes, 40 minutes
0-30 1 * * *	Every minute from 1:00 to 30 minutes every day
*/10 * * * *	every 10 minutes
*/10 2,3,4 5-6 * *	5 days, 6 days 2, 3, 4 hours every 10 minutes

```
설정값 확인 방법
How to check the set value
$ crontab -l
* * * * * echo '[{"gpio":{"status":"2","id":"io09","hold":"0"}}]' | nc ip_address
34001 -q 0 > /dev/null 2>&1
0 * * * * echo '[{"debug": true, "audio": {"volume": "77", "source": "1", "loop":
"0"}}]' | nc ip_address 34001 -q 0 > /dev/null 2>&1
```

Timer(Threading)

콘솔 윈도 내에 타이머 설정을 실행 한다.

실행명령은 기존 센서룰과 같다.

설정된 명령문은 **프로그램 재실행 시 적용** 된다.

최소 초단위 설정이 가능하다.

타이머 갯수는 config.json -> timerCmds 내 항목 추가로 가능 하다.

[타이머 버튼]의 이름을 'Heartbeat'라고 선언 하면 콘솔화면애 정해진 시간을 주기로 깜빡인다.

Execute timer setting in console window.

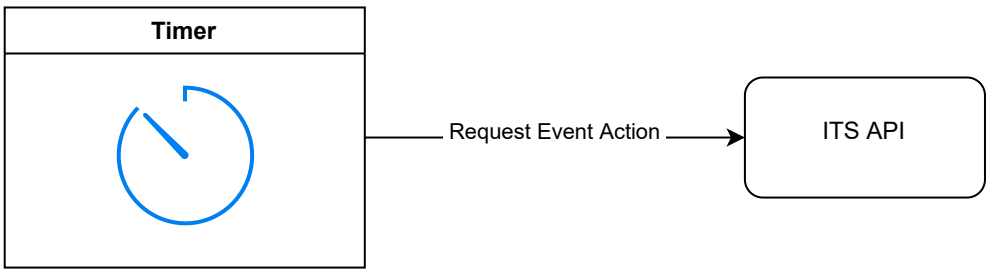
The execution command is the same as the existing sensor rule.

The set statement is applied when re-executing the program.

It is possible to set the minimum unit of seconds.

The number of timers can be added by adding an item in config.json -> timerCmds.

If the name of the [Timer Button] is declared as 'Heartbeat', the console screen will blink at a set period of time.



Log

- 실시간 로그는 콘솔 상단 우측 [log]를 통해 확인 가능 하다.
- 로그테이터는 파일 기반으로 대략 10만 라인기준으로 10개의 파일로 보관 된다.
- 약 100만 라인 이상의 로그는 자동 삭제된다.
- URL: http://ip_address/data/log/API3/API3.log

- Real-time logs can be checked through [log] on the upper right of the console.
- The log data is stored as 10 files based on approximately 100,000 lines on a file basis.
- Logs over 1 million lines are automatically deleted.
- URL: http://ip_address/data/log/API3/API3.log.

Database Query

- Database

Programming Examples

python Code Example - **pyCode.py**

```
# 특정 릴레이(io09)를 토글링 한후 1초후 되돌린다.
# 디버그 모드를 활성화("debug":True)해서 반환값을 받는다.

#!/usr/bin/env python
# -*- coding: utf-8 -*-

import socket
import json

host = 'ip_address'
port = 34001
obj = [{"gpio":{"status":"2","id":"io09","hold":"1"},"debug":True}]

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(1) # settimeout
try:
    s.connect((host,port))
    s.send(json.dumps(obj).encode('utf-8'))
    print(s.recv(1024))
except socket.error:
    print('socket.error')
except socket.timeout:
    print('socket.timeout')
finally:
    s.close()
```

- Python Code Test (Linux Terminal)

```
$
$ python pyCode.py
{"category": "gpio", "ip": "ip_address", "status": "2", "response": {"io09_desc":
0}}
$
```

PHP Code Example - **phpCode.php**

```
// 특정 릴레이(io11)를 활성화(1:On) 한후 3초 후 비 활성화("hold":"3" -> Off) 시킨다.
// 디버그 모드를 활성화("debug":True)해서 반환값을 받는다.

// After activating (1:On) a specific relay (io11), disable it after 3 seconds
("hold":"3" -> Off).
// Enable debug mode ("debug":True) and get the return value.

<?php
    $obj = '[{"gpio":{"status":"1","id":"io11","hold":"3"},"debug":true}]';

    try {
        // socket_create
        $socket = socket_create(AF_INET, SOCK_STREAM, SOL_TCP);
        if ($socket === false) {
            echo "socket_create() failed: " .
socket_strerror(socket_last_error());
        }
        // socket_connect
        $result = socket_connect($socket, "ip_address", "34001");
        if ($result === false) {
            echo "socket_connect() failed. " .
socket_strerror(socket_last_error($socket));
        }
        // socket_write
        socket_write($socket, $obj, strlen($obj));
        // socket_read when "debug":true
        print socket_read($socket, 1024);
    } finally {
        socket_close($socket);
    }

?>
```

- PHP Code Test (Linux Terminal)

```
$ php phpCode.php
{"category": "gpio", "ip": "ip_address", "status": "2", "response": {"io11_desc":
1}}
$
```

```
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
/_/   Daivoc Kim (Developer)  /_/
/_/         daivoc@gmail.com  /_/
/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/
```