
IoT.own 사용 설명서

Revision 1.3



Hosted by (주)피엘네트웍스

PLNetworks

Powered by (주)코스랩

 CoXlab

Revision History

Revision	Date	Description
1.1	Aug. 17. 2017	1st revision
1.2	Feb. 20. 2020	2nd revision
1.3	Aug. 11 .2020	3rd revision

TABLE OF CONTENTS

개요	5
IoT.own 서비스의 목적.....	5
IoT.own 서비스의 구조 및 구성요소.....	5
Gateway.....	6
Node.....	6
내부 DB.....	6
Dashboard.....	6
Third-party API.....	6
IoT.own Version Type	7
IoT.own 제공 기능	8
Gateway.....	8
Gateway Import/Export/Delete By CSV File.....	11
LoRaWAN Gateway Statistics.....	14
Node	15
Node Import/Export/Delete By CSV File.....	18
LoRaWAN Node Statistics.....	20
Dashboard	21
Default Widget.....	22
Line Chart Widget.....	24
Gauge Widget	26
Thermometer Widget	28
DB Monitor.....	31
MAP	33
Event Log	34
Restful API	36
Open API Token	36
Error Code	38
API Refetence.....	38
GATEWAY/GET-LIST	38

<i>GATEWAY/GET-INFO</i>	39
<i>NODE/GET-LIST</i>	40
<i>NODE/GET-INFO</i>	42
<i>STORAGE/GET-DATA</i>	43
<i>SET-CALLBACK</i>	45
<i>GET-CALLBACK</i>	46
<i>CLEAR-CALLBACK</i>	47
IoT.own-BIZ	48
서비스 시작 및 종료.....	48
환경설정.....	48
게이트웨이 인증서 관리	48

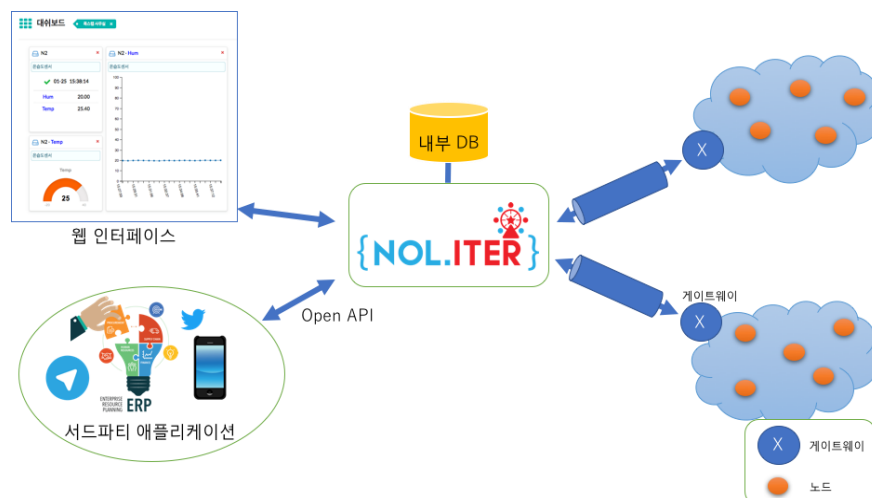
개요

IoT.own 은 사물인터넷(Internet of Things; IoT) 시스템을 쉽게 구축할 수 있도록 하는 웹 서비스입니다. 이 장에서 IoT.own 서버의 목적과 사용 시나리오를 소개합니다. 이와 관련하여 IoT.own 서비스의 구조와 구성요소를 소개하여 기본 개념에 대한 이해를 구합니다. 그리고 IoT.own 의 각 기능들에 대해 간략히 소개합니다.

IoT.own 서비스의 목적

사물인터넷은 무선 및 유선 기반의 센싱 시스템을 인터넷 및 웹과 연결을 의미하는 것으로 이를 통해 센싱 데이터의 활용을 용이하게 함으로써 그 가치가 극대화됩니다. 기존의 사물인터넷은 이를 구축하기 위해서는 일반적으로 사물(thing)의 성격, 내장된 마이크로프로세서, 통신 수단 등에 따라 연동을 위한 소프트웨어를 그때 그때 별도로 구현해야 합니다. IoT.own은 이를 하나의 기반 소프트웨어를 제공함으로써 별도 구현에 대한 시간적, 금전적 비용을 줄이는 것을 목표로 합니다.

IoT.own 서비스의 구조 및 구성요소



IoT.own은 Gateway와 Node, 내부 DB 그리고 대시보드 기능과 서드 파트 애플리케이션용 RESTful API 등으로 구성됩니다.

Gateway

Node와 IoT.own 시스템 중간에 위치하여 Node가 제공하는 센싱 데이터를 IoT.own 호환 포맷으로 변환 및 IoT.own에서 송신한 커맨드를 Node로 전달하는 역할, 통상 ARM Cortex-A 계열, MIPS 등 프로세서 기반에 유선(이더넷) 및 무선(Wi-Fi, Bluetooth, LoRa, ZigBee 등) 통신 인터페이스를 갖춘 리눅스 라우터입니다.

Node

실제 사용자가 필요한 현장에 설치되어 센서를 통해 각종 물리현상에 대한 데이터를 추출 및 이를 무선을 통해 데이터를 전송하는 장치 또는 무선을 통해 사용자로부터 명령을 수신 받아 현상의 장치를 제어하는 장치, 통상 ARM Cortex-M 계열, MSP430, AVR 등 마이크로프로세서 및 무선(Bluetooth, LoRa, ZigBee 등) 통신 인터페이스 그리고 센서 및 액추에이터로 구성된 장치(Thing) 입니다.

내부 DB

각 Node로부터 IoT.own로 수집된 센싱 데이터 및 각종 자료구조가 저장되는 곳입니다.

Dashboard

실시간으로 수집된 센싱 데이터를 시각적으로 표현하는 웹 페이지입니다.

Third-party API

내부 DB에 저장된 데이터를 서드파트에서 사용할 수 있도록 제공되는 HTTP 및 JSON 기반 RESTful API.

IoT.own Version Type

IoT.own은 서비스 제공 형태 및 구축 형태에 따라 다음 버전들이 있습니다.

- IoT.own : 클라우드에서 호스팅하는 공용 웹 서비스. 누구나 가입이 가능합니다. 무료 회원의 경우 내부 DB에 최장 일주일간 데이터가 저장됩니다. 유료 회원의 경우 내부 DB에 데이터가 한달간 저장됩니다.
- IoT.own-Biz : 설치형 IoT.own 소프트웨어. 기업용 유료 서비스로 기업이 보유하고 있는 서버에 직접 설치해 드립니다. 서버 당 Gateway 최대 100대, Node 최대 1만대까지 지원합니다.
- IoT.own-Lite : Gateway에 일체형으로 설치되어 로컬에서만 접속할 수 있는 경량형 서비스입니다. 서버 당 Gateway 최대 10대, Node 최대 200대까지 지원합니다.

IoT.own 제공 기능

Gateway

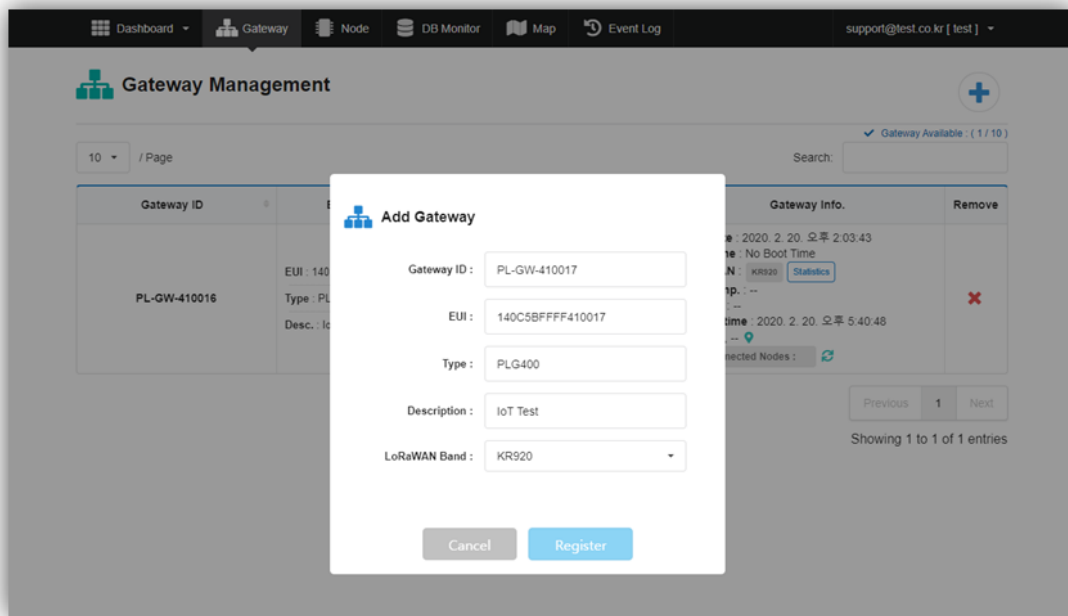
새로운 Gateway 를 추가하거나 기존 Gateway 를 삭제, 및 Gateway 현황을 살펴볼 수 있는 인터페이스입니다.

The screenshot shows the 'Gateway Management' page in the IoT.own dashboard. The top navigation bar includes 'Dashboard', 'Gateway', 'Node', 'DB Monitor', 'Map', and 'Event Log'. The user is logged in as 'support@test.co.kr [test]'. The main content area features a '+ Gateway Management' header and a search bar. Below is a table with the following columns: Gateway ID, EUI / Type / Description, Recent Act., Gateway Info., and Remove. A single gateway is listed with ID 'PL-GW-410016', EUI '140CSBFFFF410016', Type 'PLG400', and Description 'IoT Test'. The 'Recent Act.' column shows '2020. 2. 20. 오후 2:06:25'. The 'Gateway Info.' column provides details like 'Reg. Date', 'Boot Time', 'LoRaWAN', 'CPU Temp.', 'Local IP', 'System time', and 'GPS'. A 'Remove' button with a red 'X' icon is in the last column. At the bottom, it says 'Showing 1 to 1 of 1 entries'.

Gateway ID	EUI / Type / Description	Recent Act.	Gateway Info.	Remove
PL-GW-410016	EUI : 140CSBFFFF410016 Type : PLG400 Desc. : IoT Test	2020. 2. 20. 오후 2:06:25	Reg. Date : 2020. 2. 20. 오후 2:03:43 Boot Time : No Boot Time LoRaWAN : KR920 CPU Temp. : -- Local IP : -- System time : 2020. 2. 20. 오후 2:06:25 GPS : --, -- > Connected Nodes :	

새로운 Gateway 를 추가하려면 우측 상단의 버튼을 클릭합니다.

This screenshot shows the same 'Gateway Management' page, but with the 'Add Gateway' dropdown menu open. The menu options are: 'Add Gateway', 'Import from CSV file', 'Export to CSV file', 'Delete by CSV file', and 'Delete All'. The table and other interface elements remain the same as in the previous screenshot.



Gateway 추가하기에서 아이디, EUI, 종류 및 설명을 입력한 후 '등록'을 클릭하면 Gateway가 추가됩니다.

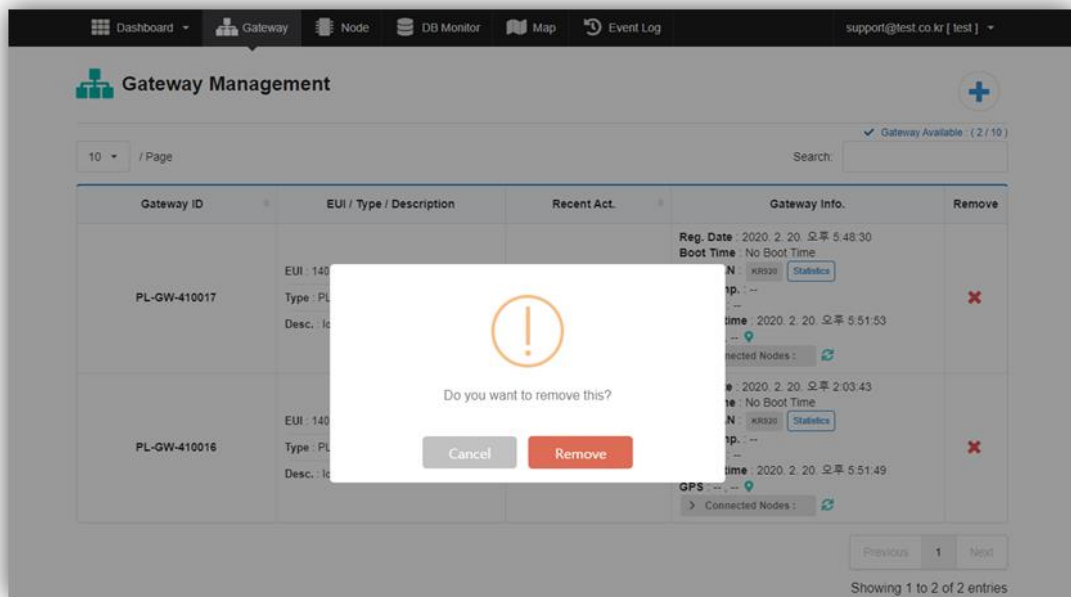
- 아이디 : Gateway의 정보를 활용하려면 Gateway 이름이 'gateway_id'의 이름과 일치해야 합니다.
- EUI : IEEE의 EUI-64를 의미하며, MSB first 순서를 입력해야 합니다. LoRaWAN을 지원하는 Gateway인 경우 필수로 입력해야 합니다.
- LoRaWAN 대역 : LoRa Gateway의 경우 다음 지원 대역 중 하나를 선택합니다.
 - KR920
 - KR920_FSK
 - AS923
 - AS923_INDONESIA
 - AS923_JAPAN

Gateway ID 우측의 ▼ 버튼을 클릭하면 상세한 정보를 볼 수 있습니다.

The screenshot shows the 'Gateway Management' interface. At the top, there's a navigation bar with 'Dashboard', 'Gateway', 'Node', 'DB Monitor', 'Map', and 'Event Log'. The 'Gateway' tab is active. Below the navigation bar, there's a 'Gateway Management' header with a search bar and a 'Gateway Available: (1 / 10)' indicator. The main content area displays a table with the following columns: Gateway ID, EUI / Type / Description, Recent Act., Gateway Info., and Remove. The table contains one entry for 'PL-GW-410016'. The 'Gateway Info.' column for this entry is expanded, showing details like 'Reg. Date: 2020. 2. 20. 오후 2:03:43', 'Boot Time: No Boot Time', 'LoRaWAN: KR920', 'CPU Temp: --', 'Local IP: --', 'System time: 2020. 2. 20. 오후 2:26:34', 'GPS: --', and 'Connected Nodes: 2'. The 'Remove' column has a red 'X' icon.

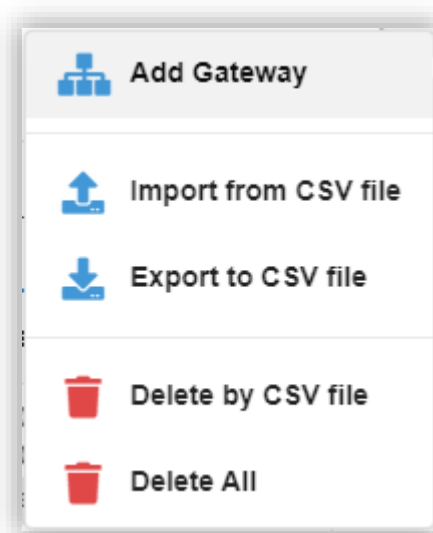
Gateway ID	EUI / Type / Description	Recent Act.	Gateway Info.	Remove
PL-GW-410016	EUI : 140C5BFFFF410016 Type : PLG400 Desc. : IoT Test	2020. 2. 20. 오후 2:26:51	Reg. Date : 2020. 2. 20. 오후 2:03:43 Boot Time : No Boot Time LoRaWAN : KR920 CPU Temp. : -- Local IP : -- System time : 2020. 2. 20. 오후 2:26:34 GPS : -- Connected Nodes : 2 LW140C5BFFFF30013A LW140C5BFFFF300154	

- Recent Act.: 마지막 활동 시간. 활동은 Node 에서 수신 한 데이터가 전달되고 연결 유지 신호가 수신됨을 의미합니다. 상태가 정상이면 파란색(1 분 내에 연결 유지 신호 수신), 그렇지 않으면 빨간색으로 표시됩니다.
- Reg. Date: Gateway 를 IoT.own 에 처음 등록된 날짜입니다.
- CPU Temp: Gateway 의 온도를 표시합니다.
- GPS: Gateway 위치 좌표를 표시합니다. 좌표는 Gateway 에 의해 자동으로 전송되거나 사용자가 수동으로 설정할 수 있습니다.
- Connected Nodes: Gateway 에 연결된 Node 수 및 목록을 표시합니다.
- LoRaWAN(Only for LoRaWAN Gateway): 지원되는 밴드가 표시되며 통계 버튼을 클릭하여 통계 정보를 볼 수 있습니다.



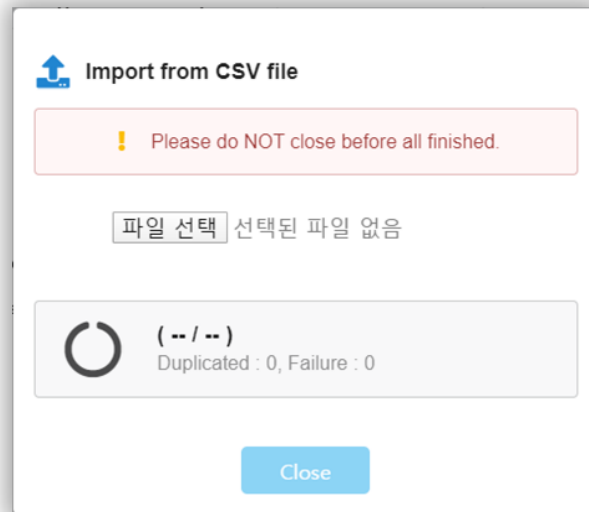
-  버튼을 누르면 Gateway 설명을 수정할 수 있습니다.
-  버튼을 누르면 Gateway 를 삭제할 수 있습니다.

Gateway Import/Export/Delete By CSV File



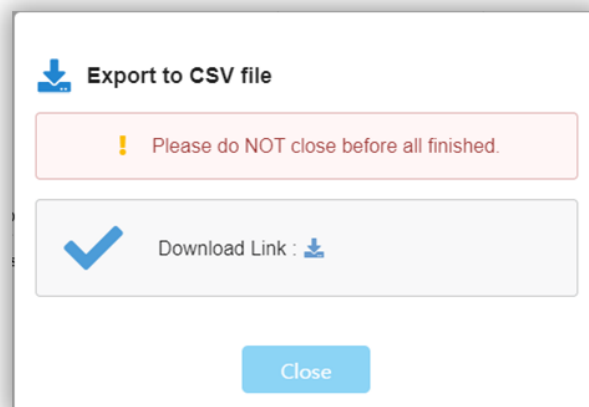
Import from CSV file

CSV 파일에서 Gateway 를 가져옵니다. CSV 파일에서 'Add Gateway', Gateway ID, EUI, Type, Description, and LoRaWAN 필드를 지정해야 합니다.



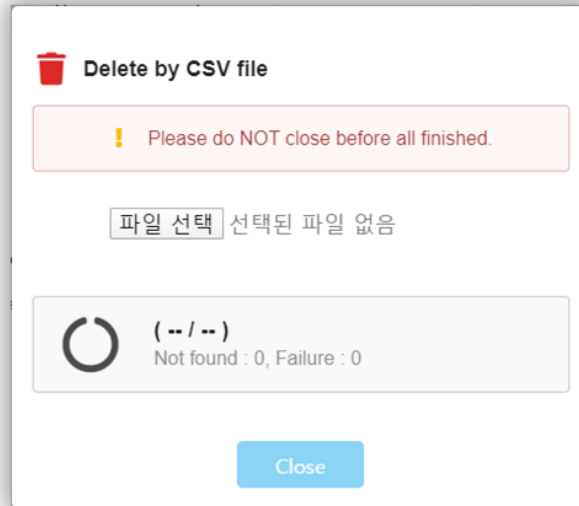
Export to CSV file

현재 Gateway 목록을 CSV 파일로 다운로드 합니다. 클릭하면 다음과 같은 다운로드 링크가 제공되며 다운로드 할 수 있습니다.



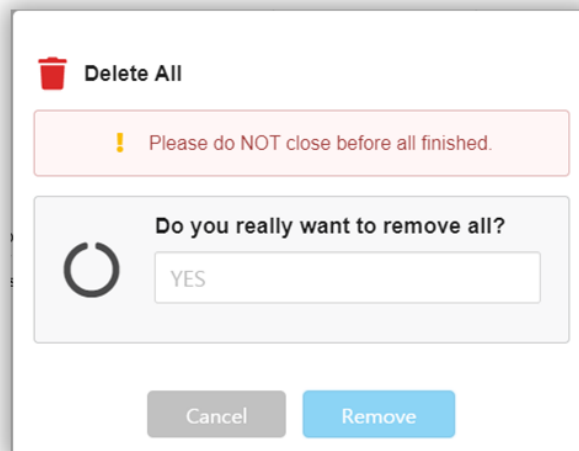
Delete by CSV file

삭제할 Gateway 목록은 CSV 파일에서 제공되고 삭제됩니다. 삭제 'Not found' 또는 'Failure'로 표시됩니다.



Delete All

모든 등록된 Gateway 를 삭제합니다. 사용자의 실수를 방지하기 위해 'YES'를 한번 더 입력해야 합니다.



LoRaWAN Gateway Statistics

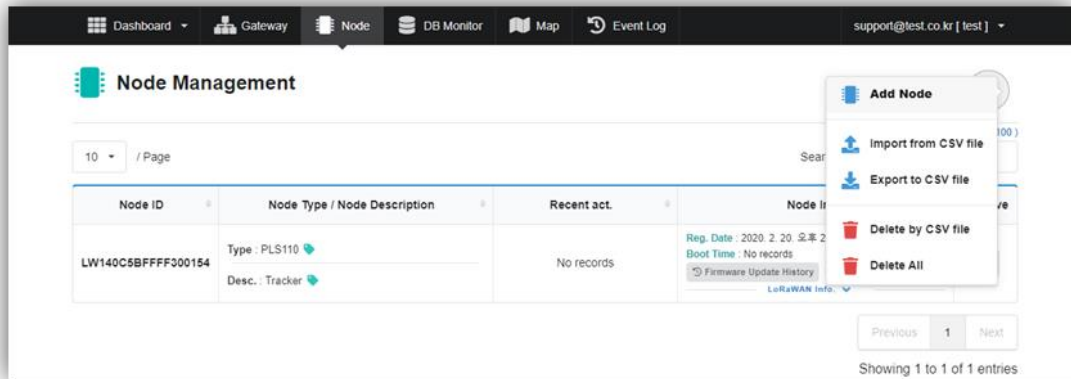
LoRaWAN Gateway 의 Statistics 버튼을 클릭하면 다음과 같은 팝업 창이 열립니다.

이를 통해 해당 Gateway 의 과거 시간 대별 통신 상태(송/수신 총 횟수, 성공 송/수신 횟수)를 확인할 수 있습니다.

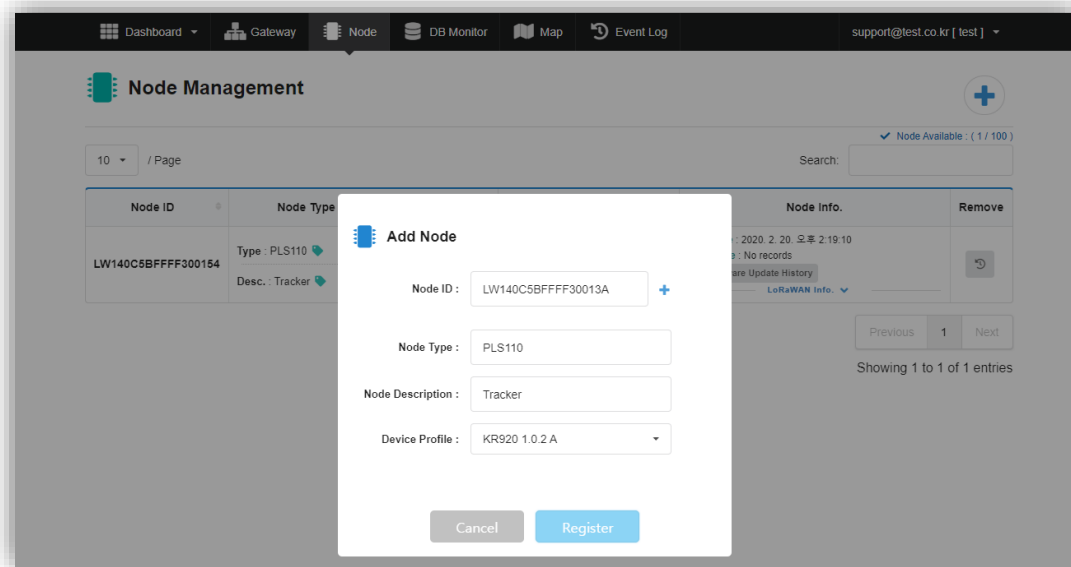


Node

새로운 Node 를 추가하거나 기존 Node 를 삭제 및 Node 현황을 살펴볼 수 있는 인터페이스입니다.



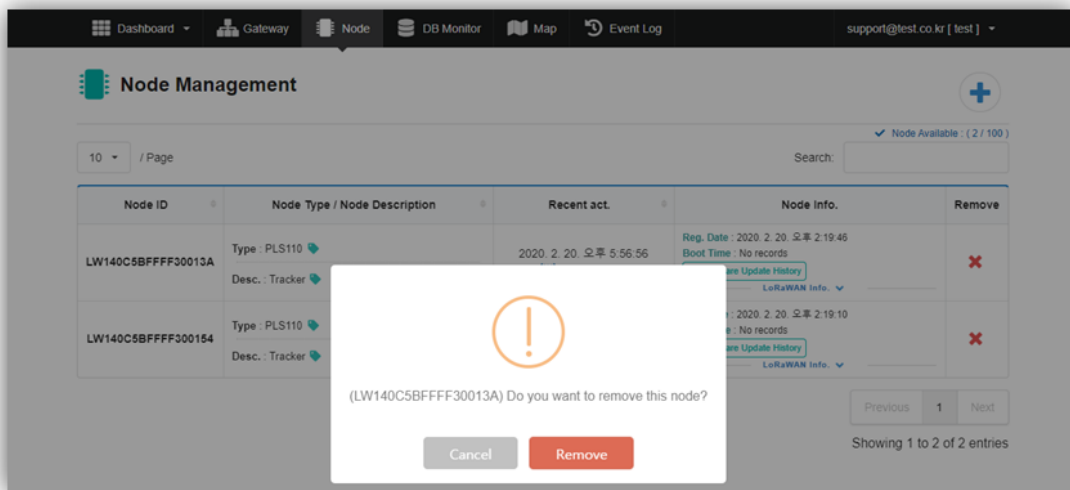
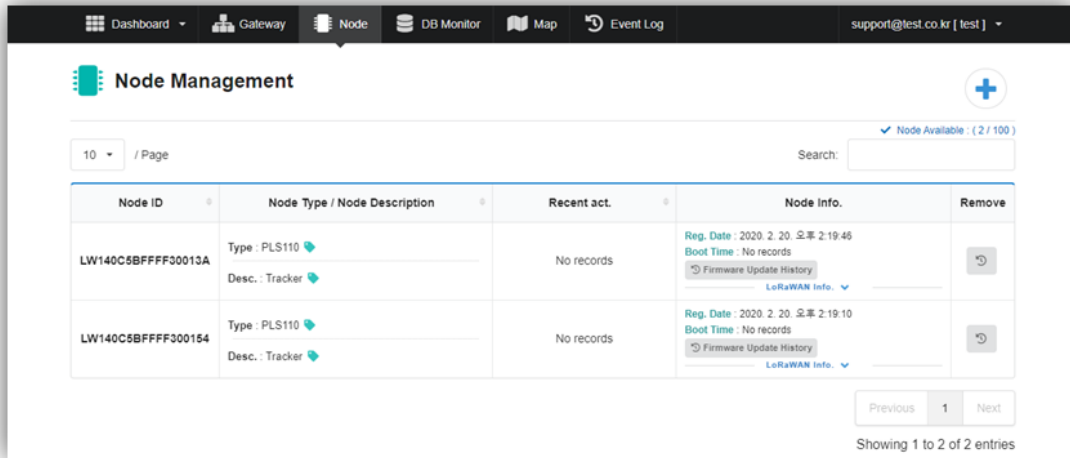
새로운 Node 를 추가하려면 우측 상단의  버튼을 클릭합니다.



'Node 추가하기'에서 Node 아이디, 종류 및 설명을 입력 후 '등록'을 클릭하면 새로운 Node 가 추가됩니다.

- Node ID : 숫자가 아닌 영문으로 시작해야 합니다. LoRaWAN 단말인 경우, LW+'EUI-64' 형태여야 하며, EUI-64 는 MSB-first 순서여야 합니다.

- Node Type : Node 유형을 입력합니다.
- Node Description : Node 에 대한 설명을 입력합니다.
- Device Profile : LoRaWAN 대역과 Class 등을 지정합니다.(예: 한국-KR920, 일본-AS923)



- 'Node 설명'의 [Edit] 버튼을 누르면 Node 설명을 수정할 수 있습니다.
- [X] 버튼을 누르면 Node 가 삭제됩니다. 연결 기록에서는 부팅 기록과 최근 통신 시간을 확인할 수 있습니다.

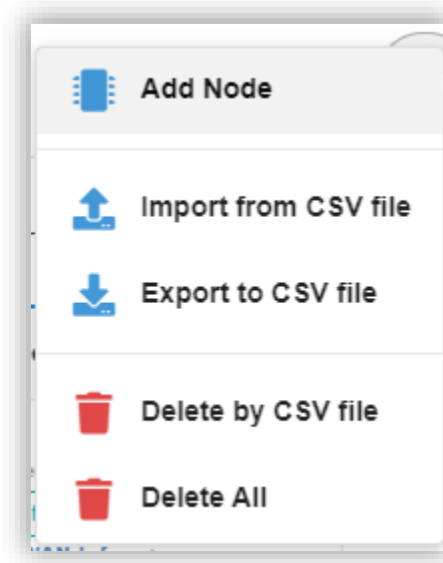
LoRaWAN 단말인 경우, LoRaWAN Info ▼ 를 클릭하면 LoRaWAN 관련 추가 정보를 확인할 수 있습니다.

The screenshot shows a 'Node Info.' window with the following sections:

- Registration Info:**
 - Reg. Date : 2020. 7. 7. 오후 12:09:47
 - Boot Time : No records
 - Firmware Update History (button)
- LoRaWAN Info. ▲** (expandable section):
 - Node Load (button)
 - AppKey : 140c5bffff300308140c5bffff300308
 - Device Profile : KR920 1.0.2 A
 - Battery Status : 68 %
 - Link Margin : -28 dB
 - Skip fCnt Check : ☒
 - Session Load (button) Reset (button)
 - AppSKey : d23e2d5cedcc71acfa323ffb9b8676f7
 - NwkSKey : 7d79ff4246f552df427fc2b642f361fd
 - DevAddr : 001be7ed
 - FCntUp : 980
 - FCntDown : 0
 - Downlink Queue Refresh (button) Clear (button)
 - Queue Length : 0
 - Statistic Chart (button)

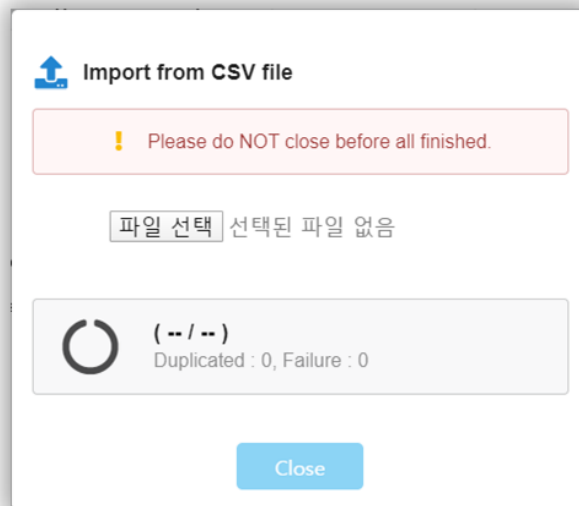
- Node
 - Node 등록 시 자동으로 생성되며 여기서 획득 한 키는 NoI.A-SDK 의 LoRaMac API 에서 사용해야 합니다. IoT.own 의 AppEUI 는 '0000000000000000'입니다.
 - Device Profile: 사용할 대역, LoRaWAN 사양 버전, Class 등이 표시됩니다.
 - **Skip fCnt Check: 활성화 시 중복된 fCnt 를 처리합니다. 비활성화 시 중복된 fCnt 는 처리하지 않습니다.**
- Session: Node 가 OTAA 를 완료 한 후 세션 정보를 확인하거나 초기화 할 수 있습니다.
 - Load: Node 에서 OTAA 가 성공한 후에 버튼을 클릭하여 생성된 세션 정보와 업데이트 된 FCntU 값을 확인합니다.
 - Reset: 세션 정보를 초기화합니다. 초기화 후에는 Node 와의 통신이 불가능 해지며 OTAA 를 다시 수행해야 합니다.
 - AppSKey, NekSKey, DevAddr: 생성된 세션에 대한 정보를 표시합니다.
 - FCntUP, FCntDown: Node 와의 통신에서 사용하고 있는 현재의 uplink, downlink 에 대한 frame counter 값입니다.

Node Import/Export/Delete By CSV File



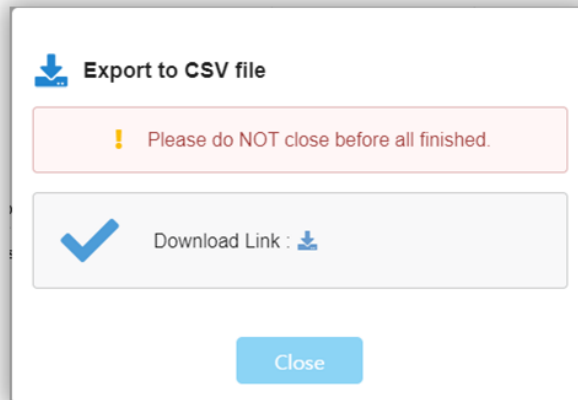
Import from CSV file

CSV 파일에서 Node 를 가져옵니다. CSV 파일에서 'Add Node', Node ID, Node Type, Node Description, AppKey, Device Profile 필드를 지정해야 합니다.



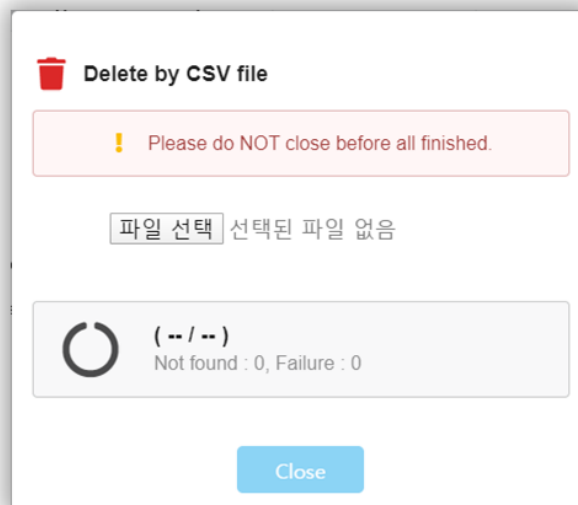
Export to CSV file

현재 Node 목록을 CSV 파일로 다운로드 합니다. 클릭하면 다음과 같은 다운로드 링크가 제공되며 다운로드 할 수 있습니다.



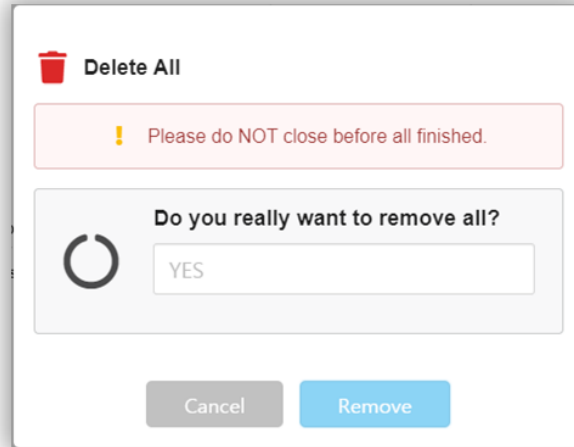
Delete by CSV file

삭제할 Node 목록은 CSV 파일에서 제공되고 삭제됩니다. 삭제 시 'Not found' 또는 'Failure'로 표시됩니다.



Delete All

모든 등록된 Node 를 삭제합니다. 사용자의 실수를 방지하기 위해 'YES'를 한번 더 입력해야 합니다.



LoRaWAN Node Statistics

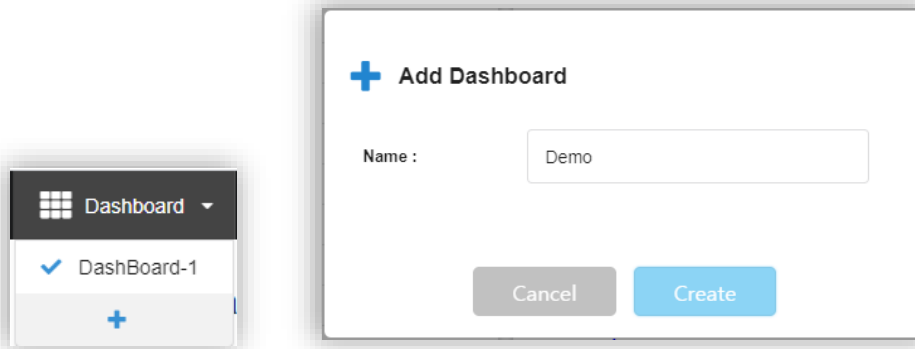
노드의 통계 'Chart' 버튼을 클릭하면 다음과 같은 팝업 창이 열립니다. 이를 통해 해당 노드의 과거 시간대 통신 상황 (uplink traffic, RSSI, SNR)을 확인할 수 있습니다.



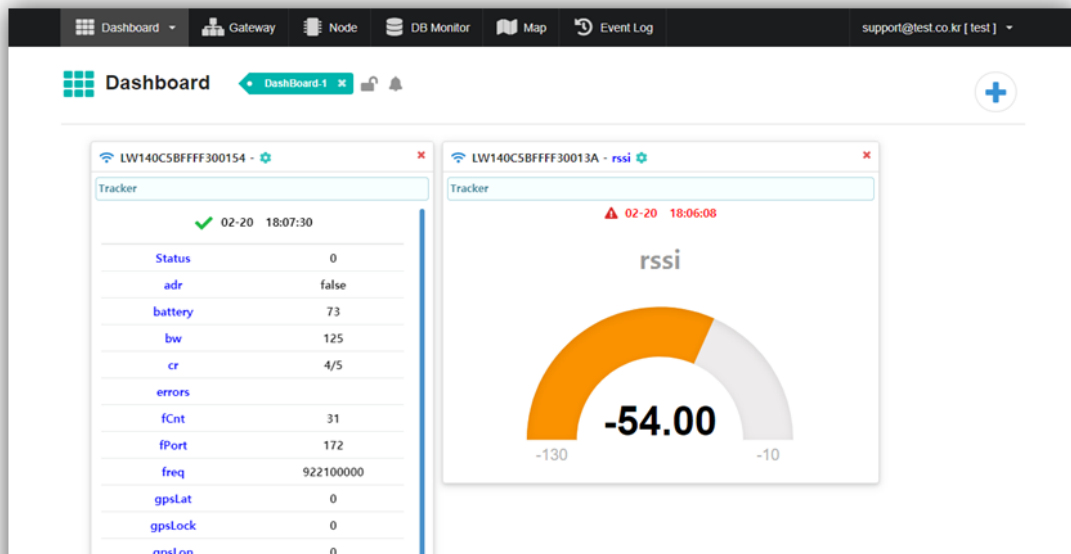
Dashboard

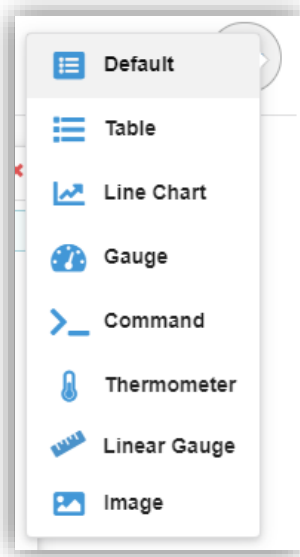
대시보드(Dashboard)는 실시간 수집 센싱 데이터를 시각화하여 사용자에서 제공하는 인터페이스입니다. 사용자는 하나 이상의 대시 보드를 만들고 사용할 수 있습니다.

새 Dashboard 를 추가하려면 'Dashboard' 메뉴를 선택하고 아래  버튼을 클릭하십시오.



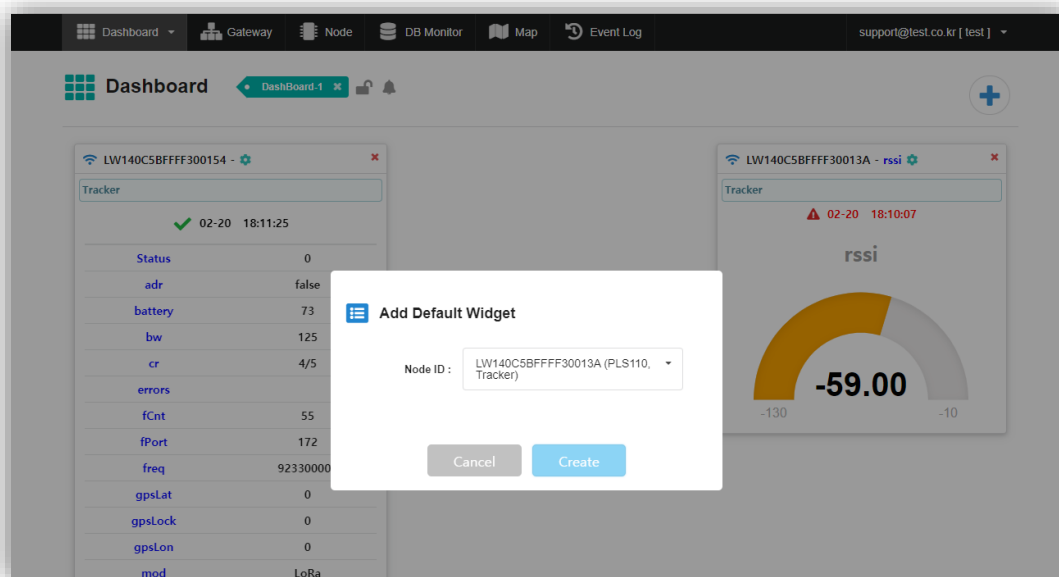
새로운 위젯을 추가하기 위해서는 우측 상단의  버튼을 클릭하면 추가할 위젯의 종류를 선택할 수 있습니다.

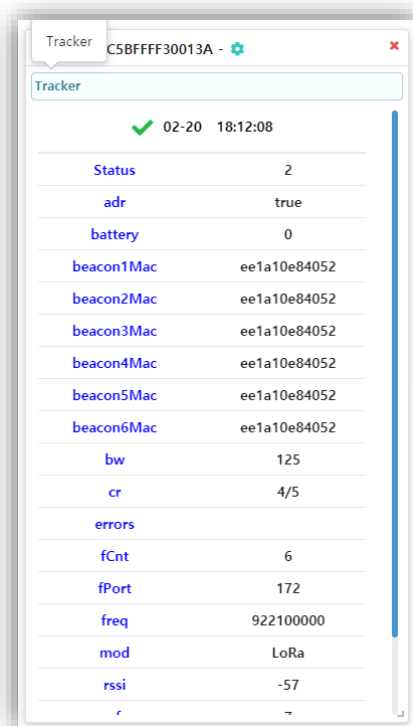




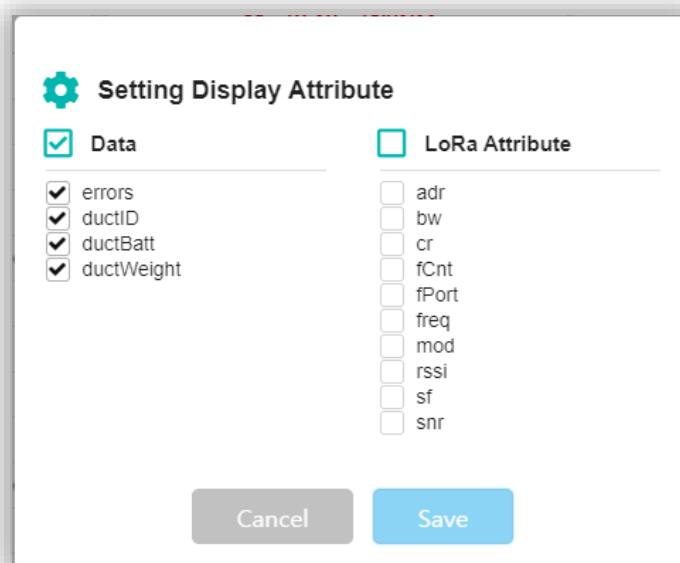
Default Widget

기본 보기는 단순히 데이터 수신 시간과 데이터를 나타냅니다.



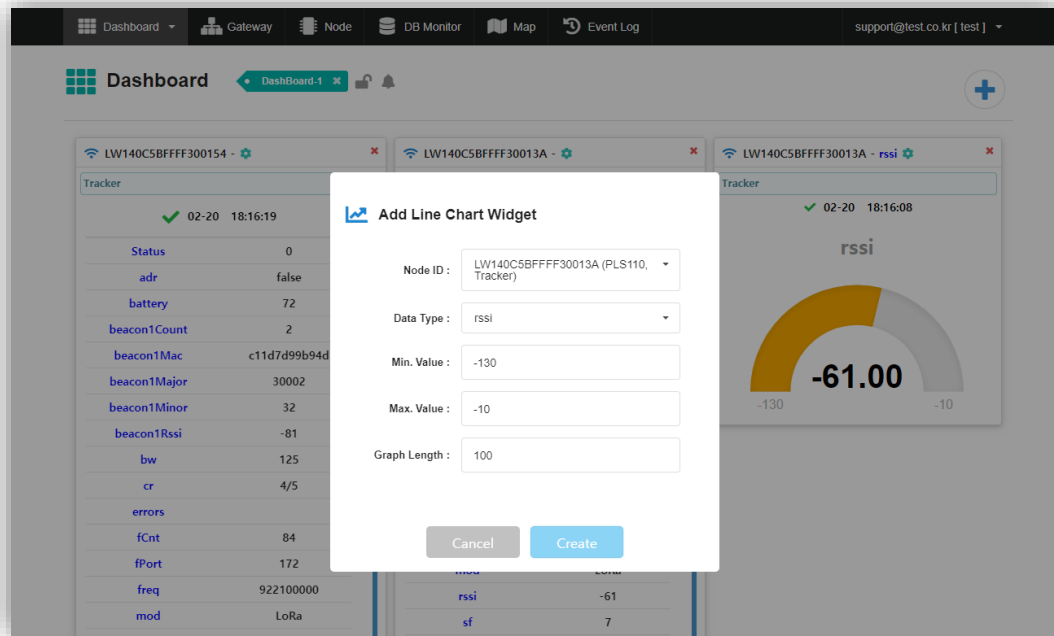


위젯 제목 오른쪽에 있는  아이콘을 클릭하여 표시할 데이터를 선택할 수 있습니다. 아래와 같이 LoRa 속성을 끄면 전송한 데이터만 선택적으로 표시할 수도 있습니다.

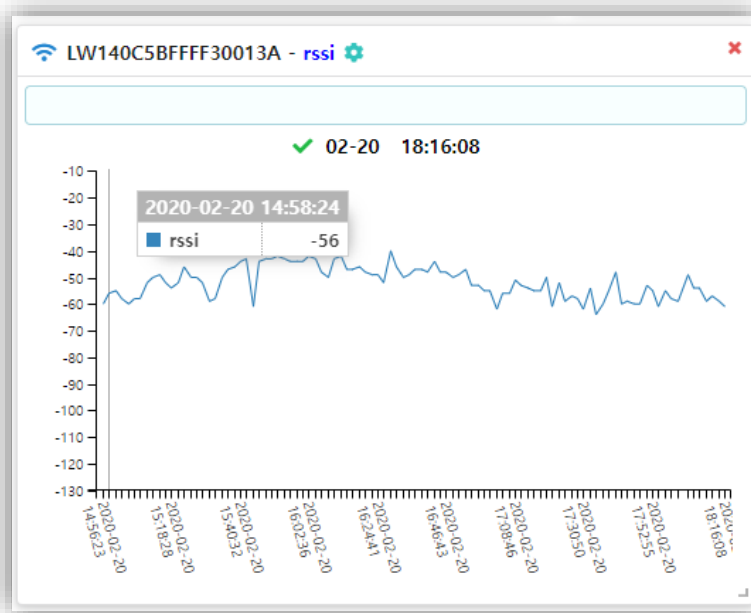


Line Chart Widget

선형 차트를 추가 할 때 표시할 데이터 유형과 최대 값 및 최소값을 지정해야 합니다.



- Node ID: 데이터를 전송하여 표현하고자 하는 Node ID 를 선택합니다.
- Data Type: 표시할 데이터 이름을 선택합니다.
- Min. Value: 데이터의 최소값을 입력합니다. 이것은 Y 축의 최소값입니다.
- Max. Value: 데이터의 최대값을 입력합니다. 이것은 Y 축의 최대값입니다.
- Graph Length: 표시할 데이터 수를 입력합니다.



위젯 제목 오른쪽에 있는  아이콘을 클릭하여 데이터 유형과 최소, 최대 값을 변경할 수 있습니다.

Line Chart Widget Setting

Data Type : battery

Min. Value : 0

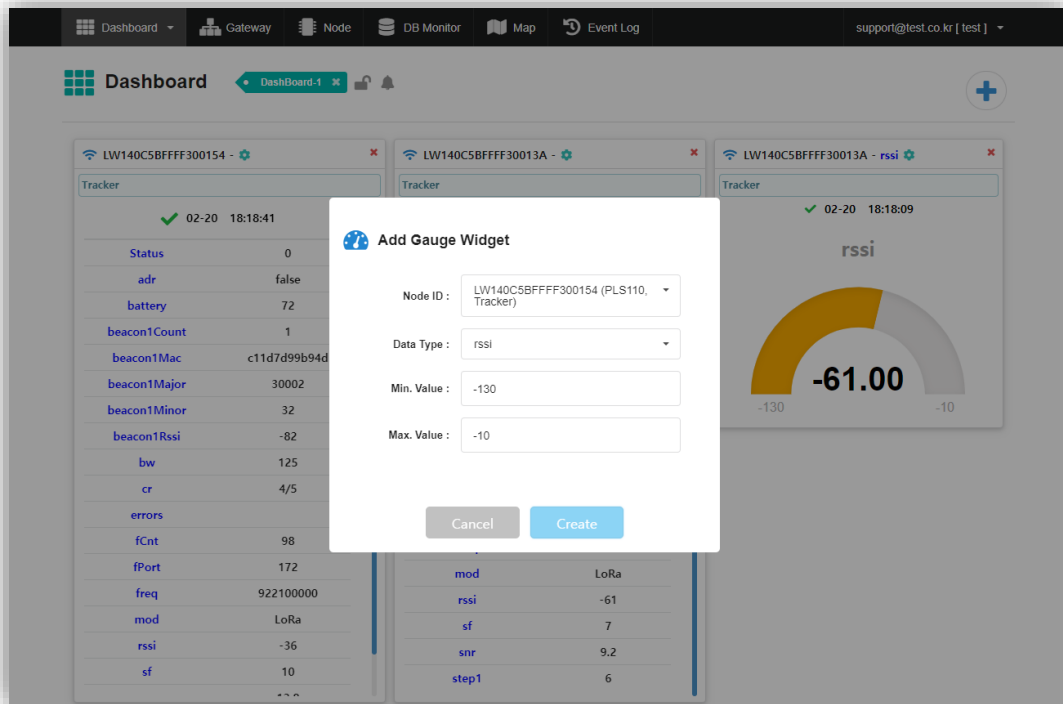
Max. Value : 100

Graph Length : 100

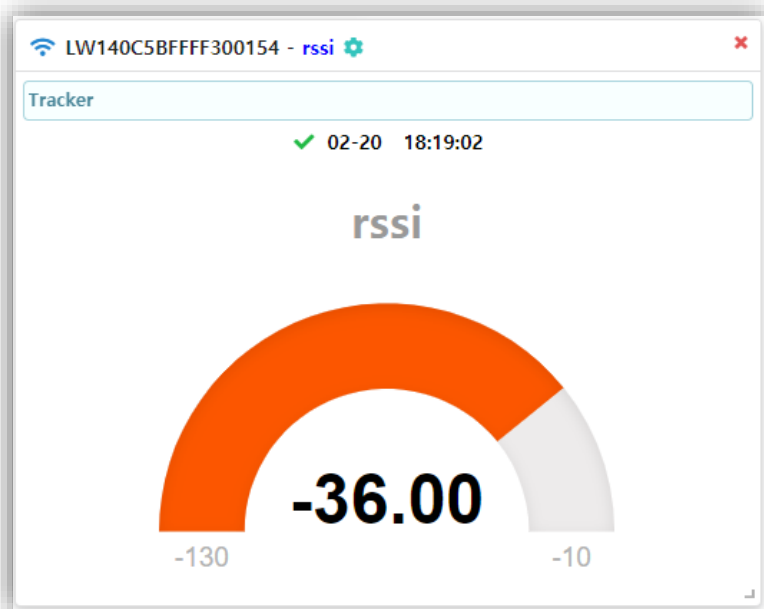
Cancel Save

Gauge Widget

Gauge 를 추가 할 때 표시할 데이터 유형과 최소 값 및 최대값을 지정 합니다.



- Node ID: 데이터를 전송하여 표현하고자 하는 Node ID 를 선택합니다.
- Data Type: 표시할 데이터 이름을 선택합니다.
- Min. Value: 데이터의 최소값을 입력합니다.
- Max. Value: 데이터의 최대값을 입력합니다.



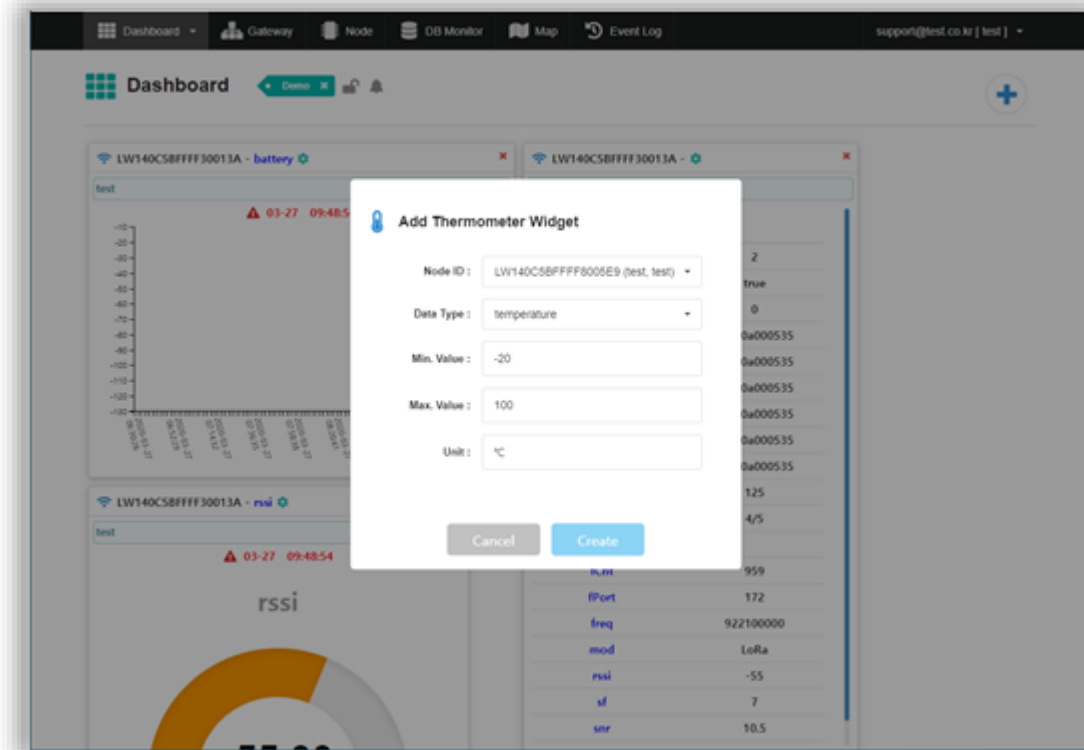
위젯 제목 오른쪽에 있는  아이콘을 클릭하여 데이터 유형과 최소, 최대 값을 변경할 수 있습니다.

The 'Gauge Widget Setting' dialog box contains the following fields and buttons:

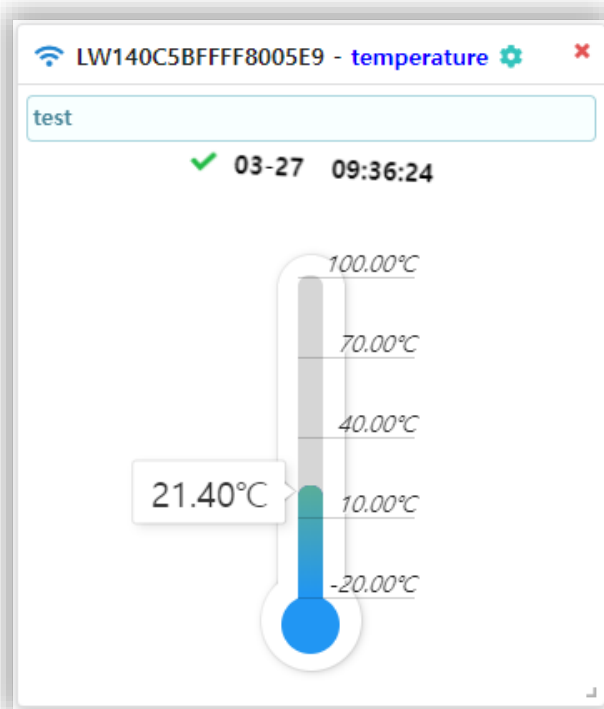
- Data Type :** A dropdown menu currently showing 'fCnt'.
- Min. Value :** A text input field containing '0'.
- Max. Value :** A text input field containing '1000'.
- Buttons:** 'Cancel' and 'Save' buttons at the bottom.

Thermometer Widget

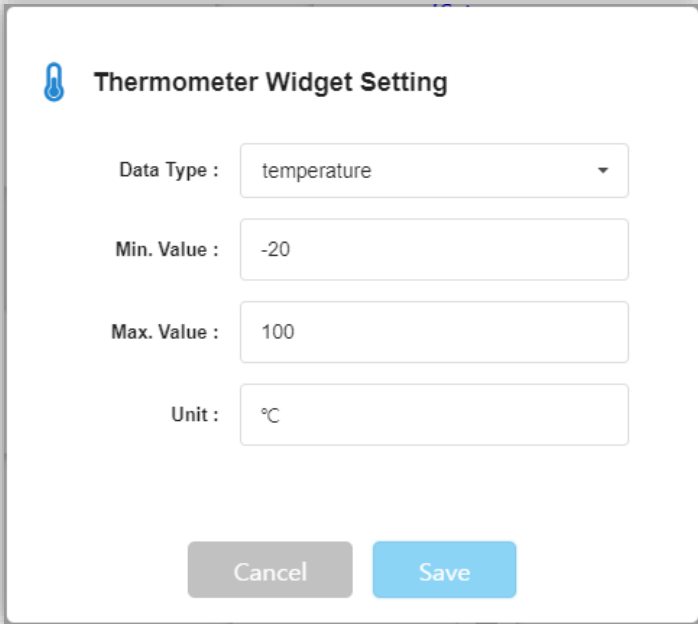
온도계 위젯을 추가할 때 표시할 데이터 유형과 최대 및 최소 단위를 지정합니다.



- Node ID: 데이터를 전송하여 표현하고자 하는 Node ID 를 선택합니다.
- Data Type: 표시할 데이터 이름을 선택합니다.
- Min. Value: 데이터의 최소값을 입력합니다.
- Max. Value: 데이터의 최대값을 입력합니다.
- Unit: 단위를 입력합니다.



위젯 제목 오른쪽에 있는  아이콘을 클릭하여 최소, 최대 값을 변경할 수 있습니다.



Thermometer Widget Setting

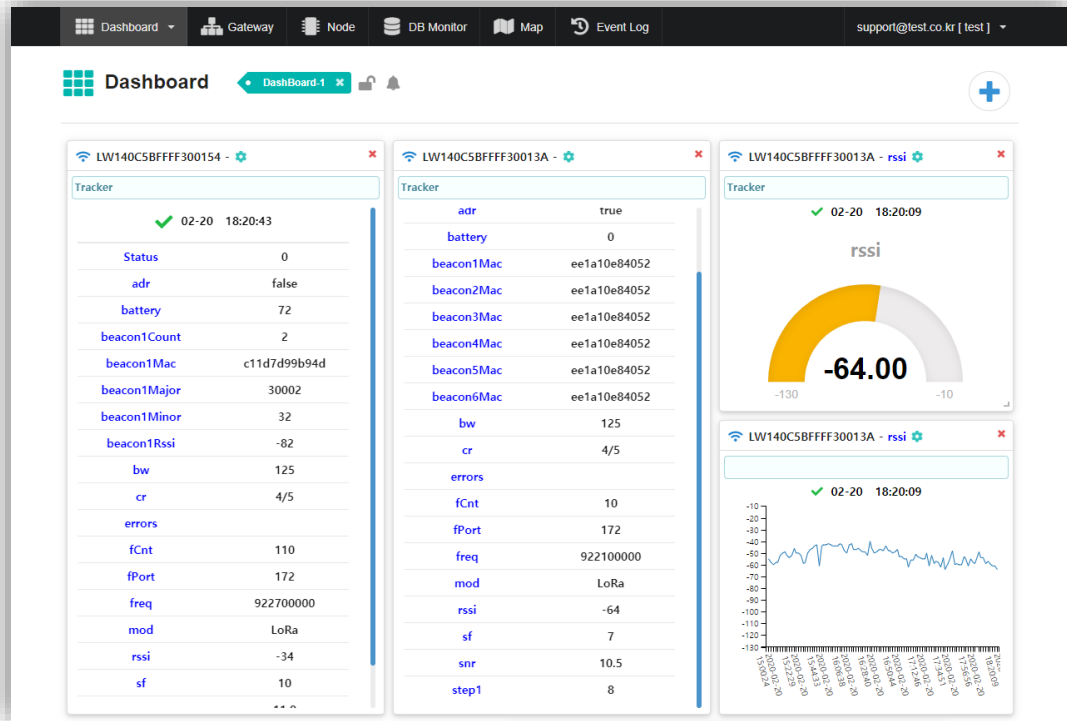
Data Type :

Min. Value :

Max. Value :

Unit :

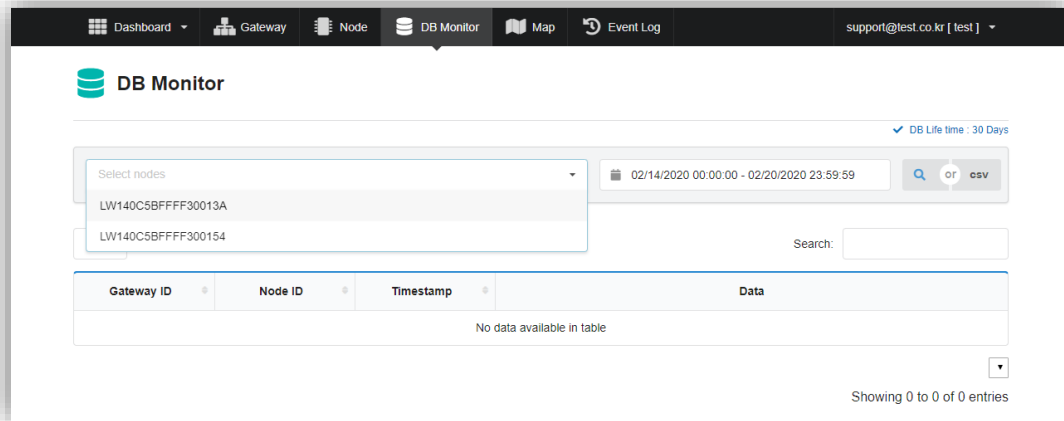
위젯은 Dashboard 내에서 이동하고 크기를 조정할 수 있습니다. 위젯을 이동하려면 위젯 제목을 드래그하면 됩니다. 위젯의 오른쪽 하단 모서리를 드래크하여 창 크기를 조정할 수 있습니다.



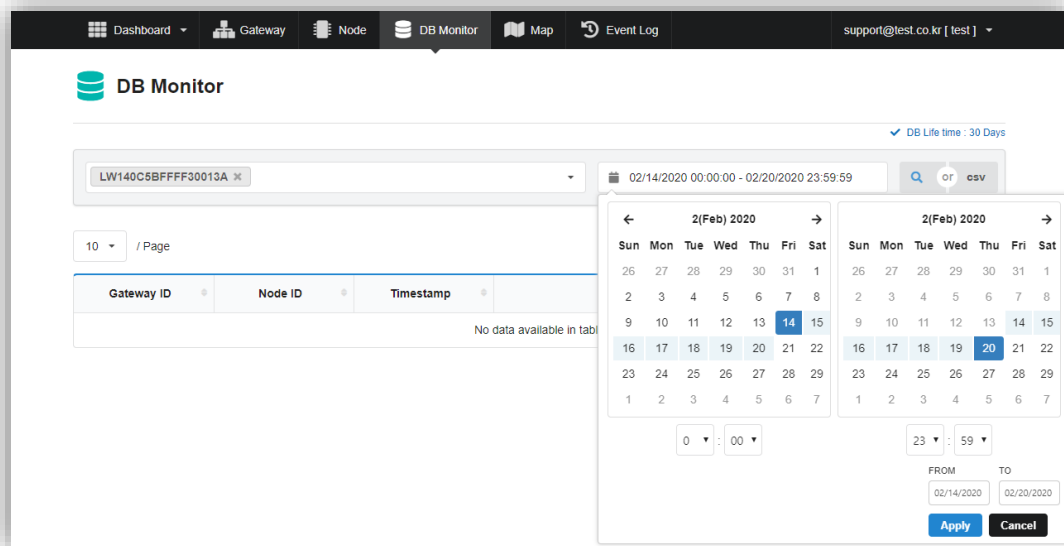
DB Monitor

노드에서 보낸 데이터는 IoT.own 내부 DB에 저장됩니다. DB Monitor는 Node 및 날짜/시간 조건에 따라 DB에 저장된 데이터를 조회할 수 있습니다.

Node 조건



날짜/시간 조건

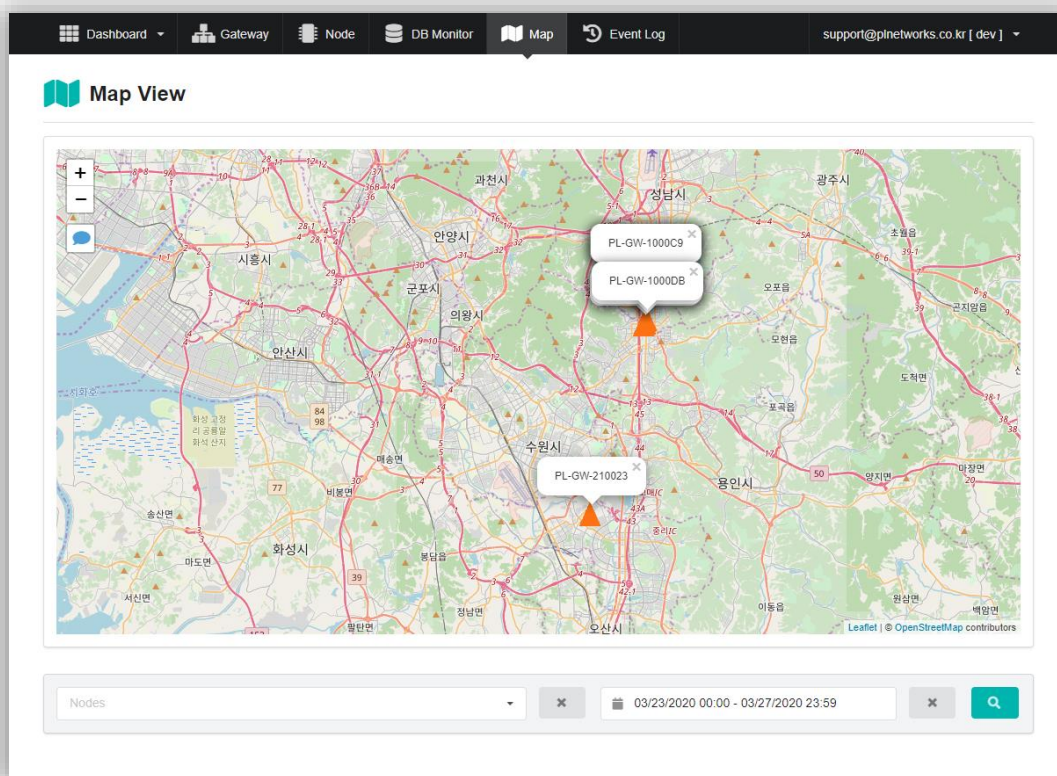


조건을 지정하지 않으면 모든 Node 및 데이터가 검색됩니다.

[illegible]

MAP

설치된 게이트웨이의 GPS 위치를 지도에 표시합니다.

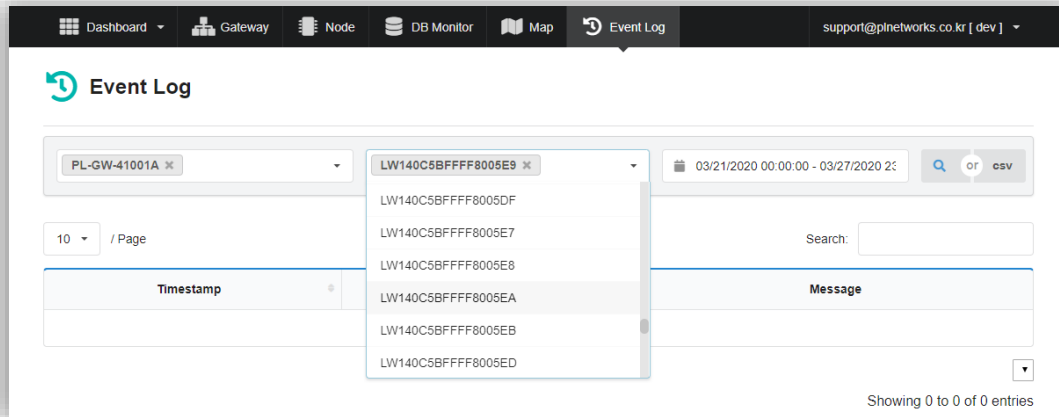


PL-GW-1000DB	EUI : 140C5BFFFF1000DB Type : PLG100M Desc. : PLG100M	2019. 8. 22. 오후 5:57:03	Reg. Date : 2019. 8. 21. 오후 7:05:17 Boot Time : 2019. 8. 22. 오후 5:50:44 LoRaWAN : KR920 CPU Temp. : --- Local IP : --- System time : --- GPS : 37.34235 , 127.10883 > Connected Nodes :
--------------	---	-------------------------	--

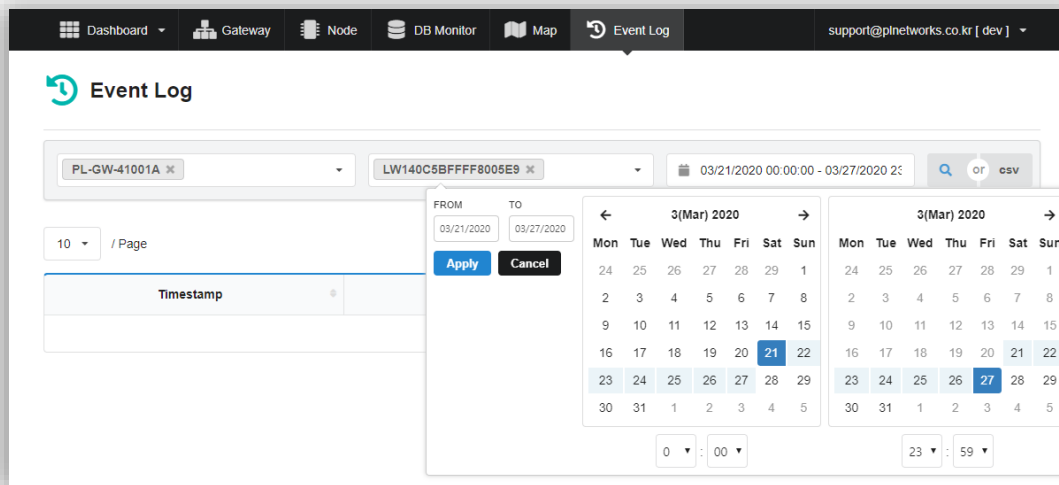
Event Log

Gateway 와 Node 가 보낸 데이터는 IoT.own 내부 DB 에 저장됩니다. Event Log 는 Gateway 의 keep alive 수신 및 Node 의 상태 데이터를 날짜/시간 조건에 따라 조회할 수 있습니다.

Gateway 및 Node 조건



날짜/시간 조건



조건을 지정하지 않으면 모든 데이터가 검색됩니다.

[Dashboard](#) [Gateway](#) [Node](#) [DB Monitor](#) [Map](#) [Event Log](#) [support@plnetworks.co.kr \[dev \]](#)

Event Log

PL-GW-41001A x

LW140C5BFFFF8005E9 x

03/21/2020 00:00:00 - 04/29/2020 23:59:59

Q or csv

10 / Page

Search:

Timestamp	Source	Message
2020. 3. 27. 오후 12:32:16	Node [LW140C5BFFFF8005E9]	Status received. - (battery: 100 %, margin: 27 dB)
2020. 3. 27. 오후 12:32:13	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)
2020. 3. 27. 오후 12:31:43	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)
2020. 3. 27. 오후 12:31:12	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)
2020. 3. 27. 오후 12:30:42	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)
2020. 3. 27. 오후 12:30:12	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)
2020. 3. 27. 오후 12:29:42	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)
2020. 3. 27. 오후 12:29:12	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)
2020. 3. 27. 오후 12:28:42	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)
2020. 3. 27. 오후 12:28:12	Gateway [PL-GW-41001A]	Keep alive received. - (CPU Temp.: --°C, GPS: [--, --], local IP: --)

1 of 26

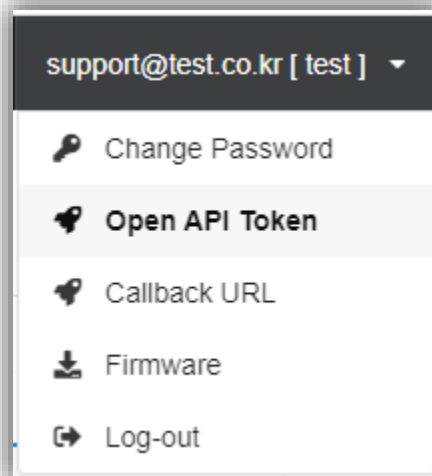
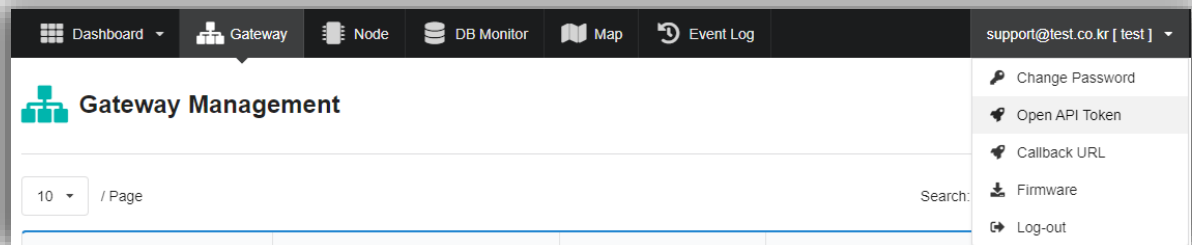
Showing 1 to 10 of 253 entries

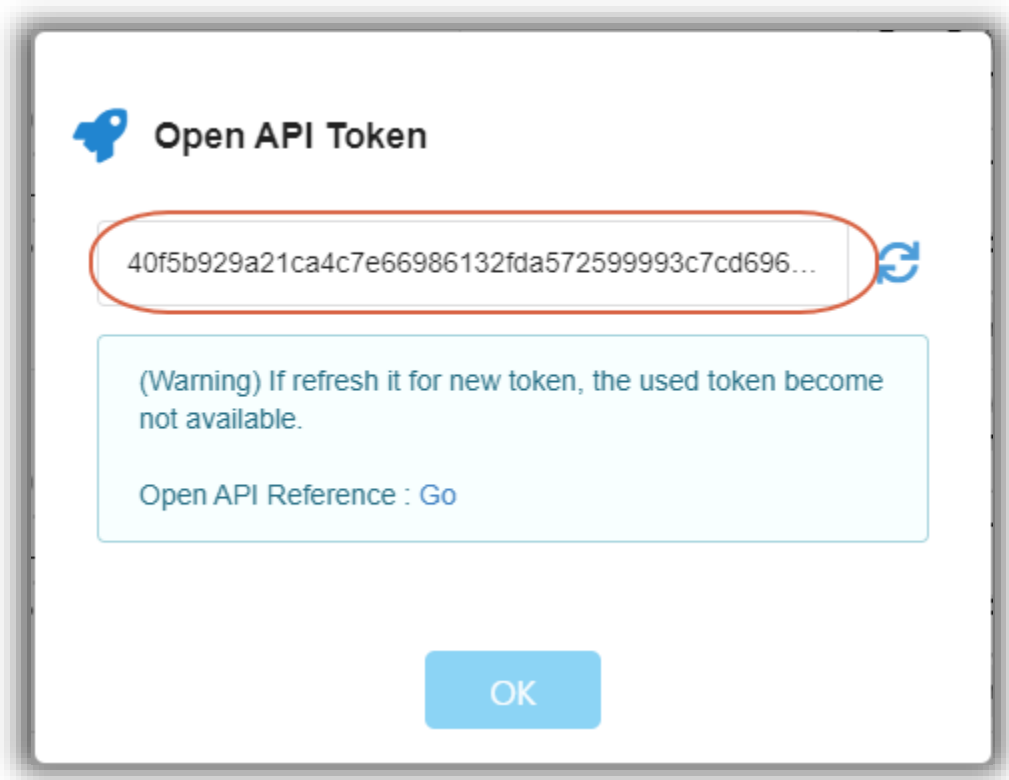
Restful API

IoT.own 은 Restful API 를 지원하여 내부 DB 에 축적 된 데이터를 제 3 자가 접근하여 사용할 수 있도록 합니다.

Open API Token

API 를 사용하려면 Token 이 필요하며, 우측 상단의 로그인 계정을 클릭하여 'Open API Token' 메뉴에서 Token 을 얻을 수 있습니다.





※ Token 이 있는 경우 Restful API 를 통해 Gateway, Node 및 데이터에 액세스 할 수 있으므로 보안에 유의하십시오. Token 이 유출된 경우  버튼을 클릭하여 Token 을 갱신할 수 있습니다.

Error Code

모든 API 는 errorCode 및 errorStr 을 반환합니다. errorCode 가 '0'이면 성공을 의미하고 다른 errorCode 는 다음과 같은 의미를 가집니다.

errorCode	errorStr	Explanation
-1	SERVER_ERR	서버 점검 혹은 장애
-2	NOT_FOUND_TOKEN_PARAM	Token 파라미터 누락
-3	INVALID_TOKEN	Token의 만료 혹은 잘못된 token
-10	NOT_FOUND_GATEWAY_ID_PARAM	gateway_id 파라미터 누락
-11	INVALID_GATEWAY_ID	등록되지 않은 gateway_id
-20	NOT_FOUND_NODE_ID_PARAM	node_id 파라미터 누락
-21	INVALID_NODE_ID	등록되지 않은 node_id
-31	INVALID_PARAM	파라미터 오류. 시간 정보가 UTC 포맷이 아닐 경우.
-40	NOT_FOUND_URL_PARAM	URL 파라미터 누락

API Refetence

GATEWAY/GET-LIST

URL : <https://town.coxlab.kr/api/gateway/get-list>

Parameter

- token : RESTful API 토큰

Response

- list[] : 게이트웨이 객체 배열
- gateway_id : 게이트웨이 아이디
- gateway_type : IoT.own 등록시 유저가 임의 입력하는 정보
- gateway_desc : IoT.own 등록시 유저가 임의 입력하는 정보
- created_at : IoT.own 에 등록된 시간(UTC)
- bootup_time (optional) : 마지막으로 부팅된 시간(UTC). 부팅 메시지가 누락된 경우,

필드가 없을 수 있음.

Example

- Request

```
https://town.coxlab.kr/api/gateway/get-  
list?token=dfc4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6cf64fd
```

- Response

```
{ "list": [  
  { "created_at": "2016-05-30T05:54:59.995Z",  
    "gateway_desc": "GW002 설명",  
    "gateway_type": "GW002 타입",  
    "gateway_id": "GW002" },  
  { "created_at": "2016-05-30T05:27:14.808Z",  
    "gateway_desc": "GW001 설명",  
    "gateway_type": "GW001 타입",  
    "gateway_id": "GW001" }  
], "errorCode": 0, "errorStr": "" }
```

GATEWAY/GET-INFO

```
URL : https://town.coxlab.kr/api/gateway/get-info
```

Parameter

- token : RESTful API 토큰
- gateway_id : 게이트웨이 아이디

Response

- gateway : 게이트웨이 객체
- gateway_id : 게이트웨이 아이디
- gateway_type : IoT.own 등록시 사용자가 임의 입력하는 정보

- gateway_desc : IoT.own 등록시 사용자가 임의 입력하는 정보
- created_at : IoT.own 에 등록된 시간(UTC)
- bootup_time (optional) : 마지막으로 부팅된 시간(UTC). 부팅 메시지가 누락된 경우, 필드가 없을 수 있음.

Example

- Request

```
https://town.coxlab.kr/api/gateway/get-
info?token=dfc4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6cf64f
d&gateway_id=GW001
```

- Response

```
{ "gateway":
  { "created_at":"2016-05-30T05:27:14.808Z",
    "gateway_desc":"GW001 설명",
    "gateway_type":"GW001 종류",
    "gateway_id":"GW001"},
  "errorCode":0,"errorStr":""}
```

NODE/GET-LIST

```
URL : https://town.coxlab.kr/api/node/get-list
```

Parameter

- token : RESTful API 토큰

Response

- list[] : 노드 객체 배열
- node_id : 노드 아이디
- node_type : IoT.own 등록시 사용자가 임의 입력하는 정보

- node_desc : IoT.own 등록시 유저가 임의 입력하는 정보
- created_at : IoT.own 에 등록된 시간(UTC)
- connected_gw_id (optional) : 노드가 연결된 노드 아이디. 부팅 메시지가 누락된 경우 제외
- bootup_time (optional) : 마지막으로 부팅된 시간(UTC). 부팅 메시지가 누락된 경우 제외
- last_data (optional) : 마지막에 기록된 데이터 객체. 데이터가 한번도 기록되지 않은 경우 제외
- last_timestamp (optional) : 마지막 데이터가 기록된 시간(UTC). 데이터가 한번도 기록되지 않은 경우 제외

Example

- Request

```
https://town.coxlab.kr/api/node/get-list?token=dfc4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6cf64fd
```

- Response

```
{ "list" : [
  { "node_id": "N003",
    "node_type": "N003 종류",
    "node_desc": "N003 설명",
    "created_at": "2016-05-27T08:13:31.633Z",
  },
  { "node_id": "N002",
    "node_type": "N002 종류",
    "node_desc": "N002 설명",
    "bootup_time": "2016-05-26T08:07:05.893Z",
    "connected_gw_id": "GW001",
    "created_at": "2016-05-27T08:13:31.633Z",
    "last_data": { "CC": "33", "BB": "11", "AA": "00" },
    "last_timestamp": "2016-05-30T07:50:10.848Z",
  },
  { "node_id": "N001",
```

```
"node_type": "N001 종류",
"node_desc": "N001 설명",
"created_at": "2016-05-27T08:13:31.633Z",
"last_data": {"A": "0", "B": "1"},
"last_timestamp": "2016-05-30T07:50:52.033Z"},
"errorCode": 0, "errorStr": ""}
```

NODE/GET-INFO

URL : <https://town.coxlab.kr/api/node/get-info>

Parameter

- token : RESTful API 토큰
- node_id : 노드 아이디

Response

- node : 노드 객체
- node_id : 노드 아이디
- node_type : IoT.own 등록시 유저가 임의 입력하는 정보
- node_desc : IoT.own 등록시 유저가 임의 입력하는 정보
- created_at : IoT.own 에 등록된 시간(UTC)
- connected_gw_id (optional) : 노드가 연결된 노드 아이디. 부팅 메시지가 누락된 경우 제외
- bootup_time (optional) : 마지막으로 부팅된 시간(UTC). 부팅 메시지가 누락된 경우 제외
- last_data (optional) : 마지막에 기록된 데이터 객체. 데이터가 한번도 기록되지 않은 경우 제외
- last_timestamp (optional) : 마지막 데이터가 기록된 시간(UTC). 데이터가 한번도 기록되지 않은 경우 제외

Example

- Request

```
https://town.coxlab.kr/api/node/get-  
list?token=dfc4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6cf64fd  
&node_id=N001
```

- Response

```
{ "node":  
  { "node_id": "N001",  
    "node_type": "N001 종류",  
    "node_desc": "N001 설명",  
    "created_at": "2016-05-27T08:13:31.633Z",  
    "last_data": {"A": "0", "B": "1"},  
    "last_timestamp": "2016-05-30T07:50:52.033Z"},  
  "errorCode": 0, "errorStr": ""}
```

STORAGE/GET-DATA

URL : <https://town.coxlab.kr/api/storage/get-data>

Parameter

- token : RESTful API 토큰
- node_id (optional) : 노드 아이디. 콤마(,)로 구분.
 - 예 : node_id=N001,N002,N003
 - node_id가 없을 경우 모든 노드를 대상으로 검색
- from (optional) : 검색할 시간 범위의 시작 값 UTC 형태만 가능. (예: from=2016-06-01T13:45+09:00)
- to (optional) : 검색할 시간 범위의 끝 값 UTC 형태만 가능. (예: from=2016-06-01T13:45+09:00)
- lastKey (optional) : 검색 결과가 5000개를 초과할 경우, 5000개와 함께 lastey가 리터됨.
 - 동일 검색 조건에 lastKey를 추가하여 다시 호출하면 다음 5000개를

리턴함.(반복적으로 5000개씩 로딩 가능)

Response

- Data : 검색된 데이터 객체 배열
- node_id : 노드 아이디
- name : 데이터 이름
- value : 데이터 값
- timestamp : 데이터가 기록된 시간(UTC)
- lastKey (optional) : 검색 결과가 5000개를 초과할 경우, 5000개와 함께 lastKey가 리턴됨.
 - 동일 검색 조건에 lastKey를 추가하여 다시 호출하면 다음 5000개를 리턴함.(반복적으로 5000개씩 로딩 가능)

Example

- Request

```
https://town.coxlab.kr/api/storage/get-  
data?token=dfe4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6cf64  
fd&node_id=N001,N002&from=2016-02-30T01:25%2B09:00&to=2016-12-  
01T01:01:20%2B09:00
```

- 주의사항 : UTC 사용시(from, to) '+' 기호 대신에 '%2B' 를 사용해야 함.

예 : 2016-02-30T01:25+09:00 -> 2106-02-30T01:25%2B09:00

- Response

```
{ "data":[  
  {"node_id":"N001","name":"B","value":"1","timestamp":"2016-02-  
30T07:50:52.033Z"},  
  {"node_id":"N001","name":"A","value":"0","timestamp":"2016-02-  
30T07:50:52.033Z"},  
  .... 생략 ....  
  {"node_id":"N001","name":"B","value":"1","timestamp":"2016-05-  
30T09:50:21.748Z"},
```

```
{ "node_id": "N001", "name": "A", "value": "0", "timestamp": "2016-05-30T09:50:21.748Z" },
{ "node_id": "N001", "name": "B", "value": "1", "timestamp": "2016-05-30T09:50:21.169Z" } ],
"lastKey": "574bf0dbabe178ec1f776758", "errorCode": 0, "errorStr": "" }
```

- 위의 예제처럼 lastKey 가 리턴된 경우, 아래와 같이 다시 호출하면 다음 5000 개를 로딩함.

```
https://town.coxlab.kr/api/storage/get-
data?token=dfe4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6cf64
fd&node_id=N001,N002&from=2016-05-30T16:25%2B09:00&to=2016-12-
01T01:01:20%2B09:00&lastKey=574bf0dbabe178ec1f776758
```

SET-CALLBACK

URL : <https://town.coxlab.kr/api/set-callback>

Parameter

- token : RESTful API 토큰
- url : callback URL

Response

- errorCode : 에러 코드
- errorStr : 에러 코드 설명

Example

- Request

```
https://town.coxlab.kr/api/set-
callback?token=dfe4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6c
f64fd&url=http://your.server.url
```

- Response

```
{"errorCode":0,"errorStr":""}
```

GET-CALLBACK

```
URL : https://town.coxlab.kr/api/get-callback
```

Parameter

- token : RESTful API 토큰

Response

- callbackURL : 등록된 URL
- callbackStatus : URL의 상태 코드
 - 0 : 등록된 URL이 없음
 - 1 : 등록된 URL이 있음
 - -1 : URL 접속이 실패하여 콜백호출이 정지됨 (예:connection timeout error)
 - -2 : URL 리턴값이 비정상인 이유로 콜백호출이 정지됨 (예:404 에러)
 - ✓ 한번 정지된 콜백호출은 set-callback을 통해 재동작이 가능함
- errorCode : 에러 코드
- errorStr : 에러 코드 설명

Example

- Request

```
https://town.coxlab.kr/api/get-callback?token=dfc4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6cf64fd
```

- Response

```
{ callbackURL : "http://your.server.url", callbackStatus : "1", "errorCode":0,"errorStr":""}
```

CLEAR-CALLBACK

URL : <https://town.coxlab.kr/api/clear-callback>

Parameter

- token : RESTful API 토큰

Response

- errorCode : 에러 코드
- errorStr : 에러 코드 설명

Example

- Request

```
https://town.coxlab.kr/api/clear-  
callback?token=dfe4767a829349cabe36e3db0fc558774a2638f5d56dbb94d0f5ddb4c6c  
f64fd
```

- Response

```
{"errorCode":0,"errorStr":""}
```

IoT.own-BIZ

서비스 시작 및 종료

- 서비스 시작

```
$ systemctl start iotown
```

- 서비스 종료

```
$ systemctl stop iotown
```

환경설정

- 설정 파일 열기

```
$ systemctl start iotown
```

- 서버 IP & Port 설정

설정 파일의 Http_DNS, Http_Port 변수를 설정하고자 하는 IP 와 Port 로 변경합니다.
변경 후 반드시 서비스를 재시작해야 합니다.

```
var Http_DNS = "https://192.168.0.5";  
var Http_Port = 443;
```

- 언어 설정

영어(us)와 한글(kr)을 지원합니다. 설정파일의 locale 값을 원하는 로케일 값으로 변경
후 서비스를 재시작하면 적용됩니다.

```
exports.locale = "us";
```

게이트웨이 인증서 관리

게이트웨이들은 서버와의 VPN 접속을 위해 서버가 발행한 인증서를 내장하고 있어야 합니다.
게이트웨이 인증서들은 /opt/coxlab/gateways 위치에 1000 개(GW1.tar.gz ~GW1000.tar.gz)가
미리 발행되어 있습니다. 이를 각 게이트웨이 내부의 지정된 위치에 압축을 풀어 저장하면
됩니다.