Network Computing and System Lab.

eddiesangwonkim@gmail.com

ksy940822@naver.com

summary

This document informs that how API and the Database are implemented

cnc m/c

*Appendix*

index

API4

Common4

index4

establish4

join5

Account5

signup5

cert6

validate6

reset6

Admin7

approve7

instantQuery7

users9

Data10

insert10

renewedData10

FCM11

register11

unregister11

compare12

Video13

list13

upload13

remove14

getInterval14

setInterval15

streaming15

MySQL17

Table17

data17

user18

View19

renewed\_data19

registered\_fcm19

# api

서버에 정의된 API의 기능과 필요로 하는 parameter를 명세함.

## Common

API의 기능을 수행하기에 앞서 필요한 기본적인 Connexion 요청과 그에 대한 응답이 정의되었다.

### index

웹브라우저로 부터 시작된 기본 URL요청에 대한 응답을 정의.

URL

/index

Methoad

GET

Parameters

N/A

Processing sequence

WhoAreYou: func 을 수행해 Application으로 부터의 request를 모두 무시한다.  
따라서, 웹 브라우저의 요청만 수행하며 index.pug를 render한다.

### Establish

애플리케이션으로 부터 시작된 기본 URL 요청에 대한 응답을 정의.

URL

/establish

Methoad

GET

Parameters

N/A

Processing sequence

config: js 에 정의된 organization의 값을 응답한다.  
이 값은 추후 애플리케이션에서 지속적으로 이용된다.

### join

웹브라우저로 부터 시작된 join페이지 render 요청에 대한 응답을 정의.

URL

/join

Methoad

GET

Parameters

N/A

Processing sequence

웹브라우저에 join: pug 를 render한다.

## Account

계정 생성과 초기화, 그 외 user token과 관련된 기능이 정의되었다.

### signup

user의 회원가입에 대한 응답을 처리하는 기능을 정의.

URL

/account/signup

Methoad

POST

Parameters

Header

N/A

Body

email: string  
name: string  
organization: string  
password: string

Processing sequence

HTTP Body에 기록된 필드 값을 DB에 기록한다.  
token, fcm 필드는 unknown value로 처리되어 추가적인 처리를 필요로 한다.

### cert

/admin/approve에 의해 승인된 사용자가 LogIn하기 위해 필요한 toke발급 기능을 정의.

URL

/account/cert

Methoad

POST

Parameters

Header

N/A

Body

email: string  
password: string

Processing sequence

compPassword: func 수행한 결과 email과 password가 일치할 경우 jwt-token을 이용해 토큰을 발급하고 이를 사용자에게 전달한다. 전달받은 사용자는 validate하는 과정이 필요하다.

### validate

서버로 부터 발급받은 token을 검증하는 기능을 정의.

URL

/account/validate

Methoad

GET

Parameters

Header

authorization: token

Body

N/A

Processing sequence

사용자가 전달한 token을 검증하고 이를 이용해 DB에서 해당 사용자를 찾는다.  
이와 함께 timestamp 를 갱신해 마지막 접속시간을 기록하고 전달한다.

### reset

사용자 암호를 초기화 하는 기능을 정의.

URL

/account/reset

Methoad

POST

Parameters

Header

N/A

Body

email: string  
name: string  
organization: string  
password: string

Processing sequence

사용자가 가입할때 입력했던 모든 정보를 입력받아 일치할 경우 해당 email의 암호를 재설정 한다..

## admin

### approve

가입을 요청한 사용자를 승인하는 기능을 정의.

URL

/admin/approve

Methoad

PUT

Parameters

Header

N/A

Body

email: string

Processing sequence

DB에서 email 필드값과 일치하는 사용자를 찾아 임시 token을 발급한다..

### instantQuery

사용자가 DB로 직접 Query를 수행할 수 있도록 기능을 정의.

URL

/admin/instant\_query

Methoad

POST

Parameters

Header

authorization: token

Body

query: string

Processing sequence

checkPermission: func을 수행해 유요한 토큰을 가진 사용자인지 확인한다.  
유효한 사용자인 경우 query를 수행한 결과를 반환한다.

### users

등록된 사용자 정보를 인출하는 기능을 정의.

URL

/admin/users

Methoad

GET

Parameters

Header

authorization: token

Body

N/A

Processing sequence

checkPermission: func을 수행해 token을 검증한 후 수행 결과를 반환한다.

## data

### insert

PLC로 부터 데이터 입력을 수행하는 기능을 정의.

URL

/data/insert

Methoad

POST

Parameters

Header

N/A

Body

data: json

Processing sequence

JSON으로 입력된 데이터를 parsing해 DB에 기록.

### renewedData

미리 정의된 DB view 를 이용해 장비별 최신 데이터를 반환하는 기능을 정의.

URL

/data/renewed\_data

Methoad

GET

Parameters

Header

authorization: token

Body

N/A

Processing sequence

checkPermission: func을 수행해 token을 검증한다.

주기적으로 요청되는 API로 Client에서 실시간 데이터 refresh에 이용된다.

## fcm

### register

Client가 FCM서버로 부터 발급받은 token을 DB에 등록하는 기능을 정의.

URL

/fcm/register

Methoad

POST

Parameters

Header

authorization: token

Body

email: string  
fcm: fcm token

Processing sequence

checkPermission: func을 수행해 token을 검증한다.

사용자가 발급받은 FCM token을 DB에 refresh해 주기적으로 push메세지를 받을 수 있도록 한다..

### unregister

DB에서 client의 FCM token을 삭제하는 기능을 정의.

URL

/fcm/unregister

Methoad

POST

Parameters

Header

authorization: token

Body

email: string

Processing sequence

checkPermission: func을 수행해 token을 검증한다.

DB에서 해당 사용자의 FCM token을 제거해 push를 받지 않도록 할 수 있다..

### compare

client가 가지고 있는 token이 DB에 기록된 내용과 일치하는지 점검하는 기능을 정의.

URL

/fcm/compare

Methoad

POST

Parameters

Header

authorization: token

Body

fcm: FCM token  
email: string

Processing sequence

checkPermission: func을 수행해 token을 검증한다.  
token이 일치하지 않는 경우 client는 token을 재발급 해야 한다.

## video

### list

서버에 저장된 장비별 영상 리스트를 인출하는 기능을 정의

URL

/video/list

Methoad

POST

Parameters

Header

authorization: token

Body

id: number

Processing sequence

checkPermission: func을 수행해 token을 검증한다.  
장비 ID마다 기록되는 디렉토리의 내용을 반환한다.

### upload

녹화장비로 부터 영상을 업로드 받는 기능을 정의.

URL

/video/upload

Methoad

POST

Parameters

Header

id: number

Body

file: binary file format

Processing sequence

장비ID의 디렉터리에 현재 서버시간으로 파일이름을 변경해 저장한다.

### remove

특정 영상을 삭제하는 기능을 정의.

URL

/video/remove

Methoad

PUT

Parameters

Header

authorization: token

Body

filename: string

Processing sequence

checkPermission: func을 수행해 token을 검증한다.

특정 장비의 특정 영상을 삭제하는 기능을 수행.

### getInterval

녹화장비와 Client에서 사전에 지정된 녹화주기를 인출하는 기능을 정의.

URL

/video/get\_interval

Methoad

GET

Parameters

Header

id: nullable number

Body

N/A

Processing sequence

id필드가 정의된 경우 해당 장비의 녹화주기만 반환하며, 정의되지 않고 request할 경우 모든 장비들의 녹화 주기를 반환한다.  
녹화장비에서 프로그램 수행 시 사용자가 미리 녹화주기를 설정하지 않는 경우 기본값 70초로 설정된다.

### setInterval

녹화장비의 녹화 주기를 설정하는 기능을 정의

URL

/video/set\_interval

Methoad

POST

Parameters

Header

N/A

Body

id: number  
interval: number

Processing sequence

checkPermission: func을 수행해 token을 검증한다.

특정 장비의 녹화 주기를 설정한다..

### streaming

요청된 영상을 스트리밍하는 기능을 정의.

URL

/video/streaming/장비ID/영상이름.mp4

Methoad

GET

Parameters

Header

N/A

Body

N/A

Processing sequence

/video/list를 수행한 결과 반환된 영상 제목을 이용해 스트리밍을 요청한다.

# Mysql

## table

### data

장비의 실시간 데이터를 기록하고 유지

Columns

seq\_Num: INT(11) AUTO\_INC PK  
id: INT(11)  
lubricant\_machin: TINYINT(1)  
lubricant\_saw: TINYINT(1)  
pressure\_air\_main: TINYINT(1)  
pressure\_oil\_hydraulic: TINYINT(1)  
servo\_cut: TINYINT(1)  
servo\_transfer: TINYINT(1)  
spindle: TINYINT(1)  
safety\_door: TINYINT(1)  
depletion: TINYINT(1)  
workload: INT(10) unsigned  
timestamp: INT(11)

DDL

create table data  
(  
 seq\_Num int auto\_increment  
 primary key,  
 id int not null,  
 lubricant\_machine tinyint(1) not null,  
 lubricant\_saw tinyint(1) not null,  
 pressure\_air\_main tinyint(1) not null,  
 pressure\_oil\_hydraulic tinyint(1) not null,  
 servo\_cut tinyint(1) not null,  
 servo\_transfer tinyint(1) not null,  
 spindle tinyint(1) not null,  
 safety\_door tinyint(1) not null,  
 depletion tinyint(1) not null,  
 workload int(10) unsigned not null,  
 timestamp int not null  
)

### user

가입된 사용자들의 정보를 기록하고 유지

Columns

email: VARCHAR(255) PK  
password: TEXT  
name: VARCHAR(20)  
organization: TEXT  
token: LONGTEXT  
timestamp: INT(11)  
fcm: LONGTEXT

DDL

create table users  
(  
 email varchar(255) not null  
 primary key,  
 password text null,  
 name varchar(20) null,  
 organization text null,  
 token longtext null,  
 timestamp int null,  
 fcm longtext not null,  
 constraint users\_email\_uindex  
 unique (email)  
)

**create unique index** users\_email\_uindex  
 **on** users (**email**)

## view

### renewed\_data

Columns

fcm: LONGTEXT

DDL

**create view** registered\_fcm **as   
SELECT** `cnc\_mc`.`users`.**`fcm` AS** `fcm`  
 **FROM** `cnc\_mc`.`users`  
 **WHERE** (`cnc\_mc`.`users`.**`fcm`** <> **'-'**)

### registered\_fcm

Columns

seq\_Num: INT(11) AUTO\_INC PK  
id: INT(11)  
lubricant\_machin: TINYINT(1)  
lubricant\_saw: TINYINT(1)  
pressure\_air\_main: TINYINT(1)  
pressure\_oil\_hydraulic: TINYINT(1)  
servo\_cut: TINYINT(1)  
servo\_transfer: TINYINT(1)  
spindle: TINYINT(1)  
safety\_door: TINYINT(1)  
depletion: TINYINT(1)  
workload: INT(10) unsigned  
timestamp: INT(11)

DDL

**create view** renewed\_data **as**(**SELECT** `cnc\_mc`.`data`.**`seq\_Num` AS** `seq\_Num`,  
 `cnc\_mc`.`data`.**`id` AS** `id`,  
 `cnc\_mc`.`data`.**`lubricant\_machine` AS** `lubricant\_machine`,  
 `cnc\_mc`.`data`.**`lubricant\_saw` AS** `lubricant\_saw`,  
 `cnc\_mc`.`data`.**`pressure\_air\_main` AS** `pressure\_air\_main`,  
 `cnc\_mc`.`data`.**`pressure\_oil\_hydraulic` AS** `pressure\_oil\_hydraulic`,  
 `cnc\_mc`.`data`.**`servo\_cut` AS** `servo\_cut`,  
 `cnc\_mc`.`data`.**`servo\_transfer` AS** `servo\_transfer`,  
 `cnc\_mc`.`data`.**`spindle` AS** `spindle`,  
 `cnc\_mc`.`data`.**`safety\_door` AS** `safety\_door`,  
 `cnc\_mc`.`data`.**`depletion` AS** `depletion`,  
 `cnc\_mc`.`data`.**`workload` AS** `workload`,  
 `cnc\_mc`.`data`.**`timestamp` AS** `timestamp`  
 **FROM** `cnc\_mc`.`data`  
 **WHERE** `cnc\_mc`.`data`.**`timestamp` IN** (**SELECT** *max*(`cnc\_mc`.`data`.**`timestamp`**)  
 **FROM** `cnc\_mc`.`data`  
 **GROUP BY** `cnc\_mc`.`data`.**`id`**)  
 **GROUP BY** `cnc\_mc`.`data`.**`id`**)