**Smart Paint Shop Query Guide**

**기술연구소**

|  |  |  |
| --- | --- | --- |
| **Date** | **Writer** | **Description** |
| 2019.09.04 | 제갈 윤 | 초본 |
| 2019.09.23 | 제갈 윤 | 알람 API 추가 |
| 2019.10.04 | 제갈 윤 | 진단 토크데이터 API 추가 |
| 2019.10.08 | 제갈 윤 | 진단 토크온도 API 추가 |
| 2019.11.11 | 제갈 윤 | 진단 알람통계 API 변경 |
|  |  |  |

1. 정보

|  |
| --- |
| GET /info/factorys |
| 없음 |
| 사이트의 factory id 를 리턴한다. |

|  |
| --- |
| GET /info/robots |
| SELECT zone\_id AS zone, robot\_name AS name, robot\_id AS id, robot\_type AS robottype FROM def\_robot\_config WHERE factory\_id= (팩토리 아이디) ORDER BY show |
| 로봇 설정 정의 테이블에서 존 아이디, 로봇 이름, 로봇 아이디 그리고 로봇 종류에 대해서 가져온다. |

|  |
| --- |
| POST /info/robots |
| SELECT zone\_id AS zone, robot\_name AS name, robot\_id AS id, robot\_type AS robottype FROM def\_robot\_config WHERE factory\_id= (팩토리 아이디) ORDER BY show |
| 로봇 설정 정의 테이블에서 존 아이디, 로봇 이름, 로봇 아이디 그리고 로봇 종류에 대해서 가져온다. |

|  |
| --- |
| GET /info/alive |
|  |
| KEEP ALIVE 기능 |

1. 진단  
   2.1. 토크 데이터

|  |
| --- |
| POST /torquedata/data/gridtable |
| with tmp as (SELECT time\_stamp s\_time, e\_time FROM ( select \*,(CASE robot\_mode\_run WHEN 1 THEN LEAD(time\_stamp) OVER ( order by time\_stamp asc ) END ) e\_time from (SELECT time\_stamp, robot\_mode\_run, lag( robot\_mode\_run ) over ( order by time\_stamp asc ) tt FROM his\_plc\_data  WHERE factory\_id = $1 and booth\_id = $2 and zone\_id =  $3 and robot\_id = $4 and time\_stamp between to\_timestamp\_imu($5) - interval '1 hour' AND to\_timestamp\_imu($5) + interval '25 hour' ORDER BY time\_stamp asc)tmp WHERE (robot\_mode\_run = 1 and tt  = 0) or (robot\_mode\_run = 0 and tt = 1 )) a WHERE e\_time is not null) SELECT distinct tmp.s\_time, tmp.e\_time,rt.job\_name, round( EXTRACT ( epoch FROM (tmp.e\_time - tmp.s\_time) ) :: NUMERIC ) AS cycle\_time  FROM tmp inner join his\_robot\_torque rt ON ( rt.time\_stamp BETWEEN tmp.s\_time and tmp.e\_time )  WHERE rt.factory\_id = $1 and rt.booth\_id = $2 and rt.zone\_id = $3 and rt.robot\_id = $4 and rt.time\_stamp  between to\_timestamp\_imu($5) - interval '1 hour'  AND to\_timestamp\_imu($5) + interval '25 hour'  and job\_name not in ('HOME','MASTER') order by s\_time  desc; |
| 그리드 테이블의 데이터 목록을 가져온다.  파라미터 목록 : 팩토리 아이디, 부스아이디, 존아이디, 로봇아이디, 선택날짜 |

|  |
| --- |
| POST /torquedata/data/table |
| SELECT robot\_id, robot\_name ,atom\_model\_id ,rc\_model\_id,  def\_model\_config.model\_name, booth\_name, zone\_name , Ip\_addr,  def\_model\_config.model\_name as robot\_model , c.model\_name as atom\_model,  install\_date FROM def\_robot\_config  INNER JOIN def\_model\_config on def\_robot\_config.robot\_model\_id = def\_model\_config.model\_id  INNER JOIN def\_booth\_config on def\_robot\_config.booth\_id = def\_booth\_config.booth\_id  AND def\_robot\_config.factory\_id = def\_booth\_config.factory\_id  INNER JOIN def\_zone\_config on def\_robot\_config.booth\_id = def\_zone\_config.booth\_id  AND def\_robot\_config.factory\_id = def\_zone\_config.factory\_id  AND def\_robot\_config.zone\_id = def\_zone\_config.zone\_id  INNER JOIN def\_model\_config as c on def\_robot\_config.atom\_model\_id = c.model\_id  WHERE def\_robot\_config.factory\_id = $1 and def\_robot\_config.booth\_id = $2 and def\_robot\_config.zone\_id = $3  and def\_robot\_config.robot\_id = $4 |
| 로봇 컨트롤러의 정보를 불러 온다. |

|  |
| --- |
| POST /torquedata/data/trend |
| SELECT round(EXTRACT ( epoch FROM time\_stamp - to\_timestamp\_imu($1, 'YYYY-MM-DD HH24:MI:SS.MS' ) )::numeric, 1) AS ms, \  step\_no, T.motor\_torque [$2] AS torque \  FROM his\_robot\_torque AS T WHERE factory\_id = $3 AND booth\_id = $4 AND zone\_id = $5 AND robot\_id = $6 \  AND time\_stamp BETWEEN $7 AND $8 AND job\_name = $9 ORDER BY time\_stamp ASC, ms ASC |
| 선택 된 날짜의 작업시간 목록 별로 토크데이터 트렌드 값을 가져온다. |

2.2. 토크 범위

|  |
| --- |
| POST /diagnostics/torquerange/data/table/warning/axis/from |
| SELECT axis FROM def\_torquelimit\_config  WHERE factory\_id = 106 AND booth\_id = 3 AND zone\_id =  9 AND robot\_id = 26 AND job\_name = 'PR1-CUV' |
| 선택한 로봇 및 JOB 에서 경고값이 있는 축을 리턴 |

* 1. 토크 온도

|  |
| --- |
| POST /diagnostics/torquetemperature/data/trend |
| SELECT date\_part($1,time\_stamp), round(avg(motor\_encoder[$2])), max(motor\_encoder[$2]) FROM his\_robot\_temperature WHERE booth\_id = $3 and zone\_id = $4 and robot\_id = $5 and time\_stamp between $6 and $7 group by date\_part($1,time\_stamp); |
| 선택한 날짜, 공정의 평균 온도 최고 온도 데이터 가져옴 |

|  |
| --- |
| POST /diagnostics/torquetemperature/data/limit |
| INSERT INTO his\_temperaturelimit\_config values($1, $2, $3) |
| 온도 설정 값 저장 |

|  |
| --- |
| POST /diagnostics/torquetemperature/data/limit/renew |
| UPDATE his\_temperaturelimit\_config SET temperature\_limit = $1 WHERE update\_time = $2 |
| 온도 설정 값 저장실패 시 해당 날짜로 업데이트(같은 날짜는 PK로 묶여 있어서 중복을 할 수가 없다.) |

|  |
| --- |
| POST /diagnostics/torquetemperature/trend/week |
| SELECT date\_part('month',time\_stamp) AS month,(extract('day' from date\_trunc('week', time\_stamp) - date\_trunc('week', date\_trunc('month', time\_stamp))) / 7 + 1) as week, max(motor\_encoder[$1]), round(avg(motor\_encoder[$1]))::integer from his\_robot\_temperature where booth\_id = $2 and zone\_id = $3 and robot\_id = $4 and time\_stamp between $5 and $6 group by date\_part('month',time\_stamp),(extract('day' from date\_trunc('week', time\_stamp) - date\_trunc('week', date\_trunc('month', time\_stamp))) / 7 + 1) order by date\_part('month',time\_stamp),(extract('day' from date\_trunc('week', time\_stamp) - date\_trunc('week', date\_trunc('month', time\_stamp))) / 7 + 1) asc; |
| 주 단위의 온도 값 가져오기 |

|  |
| --- |
| GET /diagnostics/torquetemperature/data/limit |
| SELECT temperature\_limit AS limit, update\_time FROM his\_temperaturelimit\_config ORDER BY update\_time DESC LIMIT 1 |
| 온도 설정 값 가져오기(최근 저장 된 값) |

* 1. 알람 통계

|  |
| --- |
| POST /diagnostics/alarmstatistics/data/gridtable/statistics/alarm/detail/chart |
| WITH t AS (SELECT booth\_id AS process, count(booth\_id) AS count FROM his\_alarm\_list WHERE alarm\_code = $1 AND factory\_id = $2  AND time\_stamp BETWEEN to\_timestamp\_imu($3, 'YYYY-MM-DD HH24:MI:SS') AND to\_timestamp\_imu($4, 'YYYY-MM-DD HH24:MI:SS')  GROUP BY process ORDER by count(booth\_id) desc) SELECT t.count,c.booth\_name AS name  FROM t join def\_booth\_config AS c on (t.process = c.booth\_id); |
| 알람 통계에서 최다 발생 알람명, 최장 정지 알람명 의 “공정 간 경보 발생 비율” 원형 차트의 데이터를 가져온다. |

1. 유지보수

|  |
| --- |
| POST /maintenance/data/insp/list |
| SELECT  dm.maint\_point,  dm.maint\_code,  dm.maint\_name\_" + commonModule.task.getGlobalLanguage() + " AS maint\_name,  ( SELECT sub\_name FROM def\_spare\_sub\_group WHERE sub\_id = dm.unit\_id ) AS unit,  dm.maint\_cycle,  dm.maint\_description\_" + commonModule.task.getGlobalLanguage() + " AS maint\_description ,  dm.etc,  dm.file\_name,  round(  (  CAST (  SUM (  CASE  WHEN ( now\_timestamp ( ) - cm.last\_check\_date ) > CAST ( dm.maint\_cycle || ' ' || ' month' AS INTERVAL ) THEN  0 ELSE 1  END  ) AS DOUBLE PRECISION  ) / COUNT ( \* )  ) \* 100  ) AS progress  FROM  maintenance.def\_maint\_list dm  LEFT JOIN maintenance.cur\_robot\_maint cm ON ( dm.maint\_code = cm.maint\_code AND  cm.factory\_id = "  + req.body.factoryid;  if (!commonModule.common.isEmpty(req.body.boothid)) {  query +=  " AND cm.booth\_id = "  + req.body.boothid;  }  if (!commonModule.common.isEmpty(req.body.zoneid)) {  query +=  " AND cm.zone\_id = "  + req.body.zoneid;  }  if (!commonModule.common.isEmpty(req.body.robotid)) {  query +=  " AND cm.robot\_id = "  + req.body.robotid;  }  query += ")  GROUP BY  dm.maint\_code,  dm.maint\_name\_" + commonModule.task.getGlobalLanguage() + ",  dm.unit\_id,  dm.maint\_cycle,  dm.maint\_description\_" + commonModule.task.getGlobalLanguage() + ",  dm.etc,  dm.file\_name ORDER BY maint\_code ASC; |
| 유지보수의 점검 항목에 대한 쿼리.  2019.09.05 – 메인트코드 정렬 |

|  |
| --- |
| POST /data/insp/list/valve |
| with t as (  select  unnest(valve\_type) valve\_type  , unnest(valve\_name) valve\_name  , unnest( ARRAY[valve1,valve2,valve3,valve4,valve5,valve6,valve7,valve8,valve9,valve10,valve11,valve12,valve13,valve14,valve15,valve16,valve17,valve18,valve19,valve20,valve21,valve22,valve23,valve24,valve25,valve26,valve27,valve28,valve29,valve30,valve31,valve32,valve33,valve34,valve35,valve36,valve37,valve38,valve39,valve40] ) as current\_valvecount  from def\_robot\_config dr  left join defines.def\_atomizer\_config on atom\_id = atom\_model\_id  left join cur\_valve\_data cv on ( cv.booth\_id = 3 and cv.zone\_id = 14 and cv.robot\_id = 49)  where dr.booth\_id = 3 and dr.zone\_id = 14 and dr.robot\_id = 49  )  SELECT t.valve\_name,t.current\_valvecount  , ( SELECT valve\_maint\_count FROM maintenance.def\_valvemaint\_list b WHERE b.valve\_type = t.valve\_type)  , ( SELECT valve\_name FROM maintenance.def\_valvemaint\_list b WHERE b.valve\_type = t.valve\_type) as valve\_type  FROM t WHERE t.valve\_name is not null; |
| 밸브 카운트의 목록을 가져온다. |

|  |
| --- |
| POST /data/insp/list/valve/execute |
| INSERT INTO maintenance.his\_maint\_list(  factory\_id,  booth\_id,  zone\_id,  robot\_id,  time\_stamp,  maint\_code,  comment,  user\_id,  action\_type)  VALUES("  팩토리 아이디  ", "  부스 아이디  ", "  존 아이디  ", "  로봇 아이디  ", now\_timestamp(),'"  메인트코드  "','"  코멘트  "', '"  유저아이디  "', "  액션타입  ");"  commonModule.mainDB.execute(query, req.session.spsid, function(result) {  query =  "UPDATE cur\_valve\_data  SET "  밸브번호  " = 0, time\_stamp = now\_timestamp() WHERE factory\_id = "  팩토리아이디  " AND booth\_id = "  부스아이디  " AND zone\_id = "  존 아이디  " AND robot\_id = "  로봇 아이디 |
| 밸브 카운트 점검 결과를 저장한다. |

|  |
| --- |
| GET /maintenance/data/insp/:filename |
| SELECT display\_name\_ko\_kr AS display\_name, file\_name, file\_format, uid FROM def\_filemanual WHERE manual\_categories = 2 ORDER BY uid |
| 유지보수 매뉴얼을 가져옴 |

1. 알람 뷰

|  |
| --- |
| POST /alarmview/data/gridtable |
| SELECT his\_alarm\_list.time\_stamp, his\_alarm\_list.factory\_id, his\_alarm\_list.booth\_id, his\_alarm\_list.zone\_id, his\_alarm\_list.robot\_id, his\_alarm\_list.alarm\_code, his\_alarm\_list.alarm\_sub\_code, his\_alarm\_list.sub\_code\_info, his\_alarm\_list.alarm\_name, his\_alarm\_list.update\_time, his\_alarm\_list.alarm\_type, his\_alarm\_list.alarm\_level, his\_alarm\_list.alarm\_status, his\_alarm\_list.alarm\_content, his\_alarm\_list.job\_name, his\_alarm\_list.line\_no, his\_alarm\_list.step\_no, his\_alarm\_list.schedule\_id, his\_alarm\_list.alarm\_id, def\_booth\_config.booth\_name, def\_zone\_config.zone\_name, def\_robot\_config.robot\_name FROM his\_alarm\_list left outer JOIN def\_robot\_config on def\_robot\_config.factory\_id = his\_alarm\_list.factory\_id and  def\_robot\_config.booth\_id = his\_alarm\_list.booth\_id AND def\_robot\_config.zone\_id = his\_alarm\_list.zone\_id AND def\_robot\_config.robot\_id = his\_alarm\_list.robot\_id INNER JOIN def\_booth\_config ON his\_alarm\_list.factory\_id = def\_booth\_config.factory\_id AND his\_alarm\_list.booth\_id = def\_booth\_config.booth\_id inner join def\_zone\_config on his\_alarm\_list.factory\_id = def\_zone\_config.factory\_id AND his\_alarm\_list.booth\_id =  def\_zone\_config.booth\_id and his\_alarm\_list.zone\_id = def\_zone\_config.zone\_id WHERE his\_alarm\_list.factory\_id = 106AND to\_timestamp\_imu('2019-09-02','YYYY-MM-DD') <= his\_alarm\_list.time\_stamp AND ((to\_timestamp\_imu('2019-09-09','YYYY-MM-DD')) + interval '1 day') > his\_alarm\_list.time\_stamp and his\_alarm\_list.alarm\_code ilike '%4321%' and his\_alarm\_list.booth\_id = 3 and his\_alarm\_list.zone\_id = 9 and his\_alarm\_list.robot\_id = 25 and his\_alarm\_list.alarm\_type = 0 ORDER BY time\_stamp desc |
| 알람뷰 화면의 테이블 정보를 가져온다. |

|  |
| --- |
| POST /alarmview/manual |
| SELECT (SELECT display\_name\_kr FROM public.def\_filemanual AS file WHERE file.file\_name = robotalarm.alarm\_code) AS filename, sub\_code AS subcode, \  alarm\_name AS alarmname, contents, meaning, cause\_index AS causeindex, cause, remedy FROM defines.def\_robotalarm\_list\_kr AS robotalarm WHERE alarm\_code = $1 |
| 매뉴얼 파일 유무와 매뉴얼 컨텐츠 조회 |

|  |
| --- |
| POST /alarmview/manual/type |
| select (case when alarm\_code = $1 then 0 when alarm\_code != $1 then  (select (case when alarm\_code = $1 then 4 when alarm\_code != $1 then null end) AS alarmtype from defines.def\_plcalarm\_list\_kr order by alarmtype asc limit 1)  end) AS alarmtype from defines.def\_robotalarm\_list\_kr order by alarmtype asc limit 1 |
| 알람 코드로 알람 타입 조회 |

|  |
| --- |
| POST /alarmview/manual/file |
|  |
| 트러블슈팅 폴더에 매뉴얼 추가 |

|  |
| --- |
| DELETE /alarmview/manual/file/:name |
| delete from public.def\_filemanual where file\_name = $1 |
| url 파라미터로 이름을 받아 해당 테이블의 행을 삭제 |

1. 매뉴얼

|  |
| --- |
| POST /manual/type |
| select display\_name\_kr, array\_agg(file\_name), file\_format from def\_filemanual where manual\_categories = 4 GROUP BY display\_name\_kr, file\_format; |
| 트러블 슈팅 관련한 매뉴얼 리스트 업 |

|  |
| --- |
| GET /manual/type/robot/:filename |
| SELECT display\_name\_ko\_kr AS display\_name, file\_name, file\_format, uid FROM def\_filemanual WHERE manual\_categories = 1 ORDER BY uid |
| 해당 filename 파라미터 에 해당하는 매뉴얼을 찾아옴 |

|  |
| --- |
| GET /manual/type/maintenance/:filename |
| SELECT display\_name\_ko\_kr AS display\_name, file\_name, file\_format, uid FROM def\_filemanual WHERE manual\_categories = 2 ORDER BY uid |
| 해당 filename 파라미터 에 해당하는 매뉴얼을 찾아옴 |

|  |
| --- |
| GET /manual/type/trouble/:filename |
| SELECT display\_name\_ko\_kr AS display\_name, file\_name, file\_format, uid FROM def\_filemanual WHERE manual\_categories = 4 ORDER BY uid |
| 해당 filename 파라미터에 해당하는 매뉴얼을 찾아옴 |

|  |
| --- |
| GET /manual/type/alarm/:code |
| SELECT display\_name\_kr AS display\_name,  file\_name, file\_format, uid FROM def\_filemanual WHERE  manual\_categories = 4 ORDER BY uid |
| 알람 뷰 기능에서 매뉴얼 조회를 위한 API. 2019.09.10 에 모든 알람 뷰 매뉴얼을 트러블슈팅 폴더에서 관리. |