

Manual_v2

업데이트 사항

1. file_time에서 시,분,초도 parsing하도록 변경

- 시,분,초가 있는 경우

	file_id [PK] bigint	file_name text	file_time timestamp with time zone	
1	169859	bbso_halph_fl_20230617_225551	2023-06-17 22:55:51+00	
2	169858	bbso_halph_fr_20230617_225551	2023-06-17 22:55:51+00	
3	169857	bbso_halph_fr_20230617_224443	2023-06-17 22:44:43+00	
4	169856	bbso_halph_fl_20230617_224443	2023-06-17 22:44:43+00	

- 시,분,초가 없는 경우

	file_id [PK] bigint	file_name text	file_time timestamp with time zone	data_id text
1	36225	20230618_Gp_mag_1m	2023-06-18 00:00:00+00	noaa_swpc
2	36224	20230618_Gp_part_5m	2023-06-18 00:00:00+00	noaa_swpc
3	36223	20230618_Gp_xr_1m	2023-06-18 00:00:00+00	noaa_swpc
4	36222	20230617_Gp_mag_1m	2023-06-17 00:00:00+00	noaa_swpc

2. 리눅스 명령어인 `nohup` 대신 파이썬의 `subprocess` 모듈을 사용하여 백그라운드 프로세스를 실행하도록 변경

```
(db11) [wyslab@swds script]$ python scan_multi.py --cfg ../config/multi_cfg.json
2023-06-18 19:37:01,540:scan_multi:INFO:Process 'cfg_noaa_goes' started with PID: 4000438
2023-06-18 19:37:01,540:scan_multi:INFO:Process 'cfg_ghn' started with PID: 4000439
(db11) [wyslab@swds script]$
```

실행시 pid를 출력하고 pid_list.txt 라는 파일 생성

3. top에서 프로세스 이름 확인 가능

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
4000439	wyslab	20	0	378472	56336	18036	R	80.8	0.0	0:19.94	cfg_ghn
4000438	wyslab	20	0	378608	56492	18256	R	80.5	0.0	0:20.06	cfg_noaa_goes

4. pid_list.txt 파일을 활용하여 기본적인 모니터링 프로그램 제작

```
python monitor_proc.py
```

```
(db11) [wyslab@swds script]$ python monitor_proc.py
Process with PID 4000438 exists. Status: running. Name:cfg_noaa_goes.
Process with PID 4000439 exists. Status: running. Name:cfg_ghn.
(db11) [wyslab@swds script]$
```

- 모든 프로세스를 죽일때도 사용가능

```
> python monitor_proc.py --kill True
```

실행법

모든 로그는 기본적으로 파이썬에서 제공되는 로거 라이브러리를 이용하여 `log.log`에 작성됨

가상환경 생성 & 활성화

1. 가상환경 생성

```
> conda env create -f db11.yaml
```

2. 가상환경 활성화

```
> conda activate db11
```

여러 설정파일에 대해 실행

```
> python scan_multi.py --cfg <멀티실행 설정파일 경로>
```

모두 background process 로 실행함

예시

```
> python scan_multi.py --cfg multi_cfg.json
```

멀티실행 설정파일 (`multi_cfg.json`)

```
{
  "cfg_db": "cfg_db.json",
  "cfg_storage": "cfg_storage.json",
  "cfg_metadata": [
    "cfg_noaa_goes.json",
    "cfg_ghn.json"
  ]
}
```

`cfg_db`

DB 설정파일 경로

`cfg_storage.json`

Storage 설정파일 경로

`cfg_metadata`

메타데이터 설정파일 경로 리스트

config폴더를 실행파일과 다른 폴더에 놓았을 경우에, multi_cfg.json 파일에서 각 설정파일의 경로를 절대경로로 입력해야 한다.

위의 경우 상대경로이고, 아래의 경우 절대경로이다.

```
{
  "cfg_db":
"/home/wyslab/Source/database_mingyu/database/config/cfg_db.json",
  "cfg_storage":
"/home/wyslab/Source/database_mingyu/database/config/cfg_storage.json",
  "cfg_metadata": [
"/home/wyslab/Source/database_mingyu/database/config/cfg_noaa_goes.json",
  "/home/wyslab/Source/database_mingyu/database/config/cfg_ghn.json"
  ]
}
```

하나의 설정파일에 대해 실행

일반 실행

```
> python scan_single.py --db_cfg <DB 설정파일 경로> --sto_cfg <Storage 설정파일 경
로> --tbl_cfg <메타데이터 설정파일 경로>
```

백그라운드 실행 (nohup)

```
> nohup [명령어] 1> /dev/null 2>&1 &
```

예시

```
> python scan_single.py --db_cfg cfg_db.json --sto_cfg cfg_storage.json --
tbl_cfg cfg_noaa_goes.json
```

```
> python scan_single.py --db_cfg cfg_db.json --sto_cfg cfg_storage.json --
tbl_cfg cfg_ghn.json
```

```
> nohup python scan_single.py --db_cfg cfg_db.json --sto_cfg
cfg_storage.json --tbl_cfg cfg_noaa_goes.json 1> /dev/null 2>&1 &
```

```
> nohup python scan_single.py --db_cfg cfg_db.json --sto_cfg
cfg_storage.json --tbl_cfg cfg_ghn.json 1> /dev/null 2>&1 &
```

설정파일

DB 설정파일 (cfg_db.json)

```
{
  "database": {
    "url": {
      "drivername": "postgresql+psycopg2",
```

```

        "username": "postgres",
        "password": "postgres",
        "host": "localhost",
        "port": "54325",
        "database": "postgres"
    },
    "engine": {
        "echo": false
    }
}
}

```

DB 접속 psql 명령어

`psql -h {호스트명} -p {포트 번호} -U {사용자명} -d {데이터베이스명}` 에 들어가는 호스트, 포트번호, 사용자명, 데이터베이스명 + 비밀번호를 각각 `host`, `port`, `username`, `database`, `password`에 입력하면 된다.

`echo`의 경우 log에 sqlalchemy의 상세한 DB 관련 로그를 보고 싶을때 True로 하면 되나, 관련 로그양이 굉장히 많기 때문에 log파일 용량이 매우 빠르게 늘어나기 때문에 주의가 필요하다.

Storage 설정파일 (`cfg_storage.json`)

```

{
    "storage_info": [
        {
            "storage_id": "ioGuard",
            "storage_name": "Etrojan (service) (IOGuard1) | Etrojan-NAS",
            "volume_name": "/vol4, /vol5, /vol6, /vol7",
            "ip": "203.253.237.162",
            "hostname": "210.219.33.246/vol4, vol5, vol6, vol7",
            "comment": ""
        },
        {
            "storage_id": "3par_gateway",
            "storage_name": "3par (gateway) | 3PAR GATEWAY",
            "volume_name": "/hp01, /hp02, /hp03",
            "ip": "203.253.237.156",
            "hostname": "swc4.kasi.re.kr",
            "comment": ""
        }
    ]
}

```

hardware.hardware_storage							Table
Name	Data type	PK	FK	UQ	Not null	Default value	Description
storage_id	varchar(50)	✓			✓		스토리지 아이디 ex) 1. storage_name + "_" + volume_name 2. system number code
storage_name	varchar(20)						스토리지 명
volume_name	varchar(20)						볼륨명 ex) vol7
ip	varchar(20)						스토리지 IP
hostname	varchar(30)						스토리지 hostname
comment	varchar(100)						설명

해당 컬럼에 해당하는 값을 작성하면 된다.

메타데이터 설정 파일

cfg_ghn.json

```
{
  "table_info": {
    "scan_flag": false,
    "observer_flag": true,
    "table_name": "metadata_ghn",
    "data_info": [
      {
        "data_path": "/NAS/ioGuard/vol7/swc/data/bbso/Global Halpha
Network",
        "data_group": "bbso_ghn",
        "member_info": [
          {
            "data_id": "ghn_gong",
            "storage_id": "ioGuard",
            "file_format": [
              "gong_06768_fd_%Y%m%d_%H%M%S.jpg",
              "gong_magxx_fd_%Y%m%d_%H%M%S.jpg"
            ],
            "institute": "",
            "observatory": "",
            "satellite": "",
            "model": "",
            "telescope": "",
            "wavelength": "",
            "channel": "",
            "instrument": "",
            "file_server": 0
          },
          {
            "data_id": "ghn_kanz",
```

```

"storage_id": "ioGuard",
"file_format": [
    "kanz_halph_fl_%Y%m%d_%H%M%S.jpg",
    "kanz_halph_fr_%Y%m%d_%H%M%S.fts.gz",
    "kanz_halph_fr_%Y%m%d_%H%M%S.jpg"
],
"institute": "",
"observatory": "",
"satellite": "",
"model": "",
"telescope": "",
"wavelength": "",
"channel": "",
"instrument": "",
"file_server": 0
},
{
"data_id": "ghn_bbso",
"storage_id": "ioGuard",
"file_format": [
    "bbso_logs_%Y%m%d.txt",
    "bbso_logs_%Y%m%d.html",
    "bbso_halph_fl_%Y%m%d_%H%M%S.fts",
    "bbso_halph_fl_%Y%m%d_%H%M%S.fts.gz",
    "bbso_halph_fl_%Y%m%d_%H%M%S.jpg",
    "bbso_halph_fr_%Y%m%d_%H%M%S.fts",
    "bbso_halph_fr_%Y%m%d_%H%M%S.fts.gz",
    "bbso_halph_fr_%Y%m%d_%H%M%S.jpg"
],
"institute": "",
"observatory": "",
"satellite": "",
"model": "",
"telescope": "",
"wavelength": "",
"channel": "",
"instrument": "",
"file_server": 0
},
{
"data_id": "ghn_oact",
"storage_id": "ioGuard",
"file_format": [

```

```

        "oact_halph_fl_%Y%m%d_%H%M%S.fts.gz",
        "oact_halph_fl_%Y%m%d_%H%M%S.jpg",
        "oact_halph_fr_%Y%m%d_%H%M%S.fts.gz",
        "oact_halph_fr_%Y%m%d_%H%M%S.jpg"
    ],
    "institute": "",
    "observatory": "",
    "satellite": "",
    "model": "",
    "telescope": "",
    "wavelength": "",
    "channel": "",
    "instrument": "",
    "file_server": 0
}
    ]
}
    ]
}
}

```

cfg_noaa_goes.json

```

{
    "table_info": {
        "scan_flag": false,
        "observer_flag": true,
        "table_name": "metadata_noaa_goes",
        "data_info": [
            {
                "data_path":
"/NAS/ioGuard/vol7/swc/data/noaa/swpc/goes/geomag",
                "data_group": "noaa_goes_geomag",
                "member_info": [
                    {
                        "data_id": "noaa_swpc_g12_geomag_1m",
                        "storage_id": "ioGuard",
                        "file_format": [
                            "%Y%m%d_G12mag_1m.txt"
                        ],
                        "institute": "B",
                        "observatory": "A",
                        "satellite": "A",
                        "model": "A",

```

```

        "telescope": "A",
        "wavelength": "A",
        "channel": "A",
        "instrument": "A",
        "file_server": 999
    },
    {
        "data_id": "noaa_swpc_gp_geomag_1m",
        "storage_id": "ioGuard",
        "file_format": [
            "%Y%m%d_Gp_mag_1m.txt"
        ],
        "institute": "",
        "observatory": "",
        "satellite": "",
        "model": "",
        "telescope": "",
        "wavelength": "",
        "channel": "",
        "instrument": "",
        "file_server": 0
    },
    {
        "data_id": "noaa_swpc_gs_geomag_1m",
        "storage_id": "ioGuard",
        "file_format": [
            "%Y%m%d_Gs_mag_1m.txt"
        ],
        "institute": "",
        "observatory": "",
        "satellite": "",
        "model": "",
        "telescope": "",
        "wavelength": "",
        "channel": "",
        "instrument": "",
        "file_server": 0
    }
]

},
{
    "data_path":
"/NAS/ioGuard/vol7/swc/data/noaa/swpc/goes/particle",

```



```
"data_group": "noaa_goes_particle",
"member_info": [
  {
    "data_id": "noaa_swpc_g9_particle_5m",
    "storage_id": "ioGuard",
    "file_format": [
      "%Y%m%d_G9part_5m.txt"
    ],
    "institute": "",
    "observatory": "",
    "satellite": "",
    "model": "",
    "telescope": "",
    "wavelength": "",
    "channel": "",
    "instrument": "",
    "file_server": 0
  },
  {
    "data_id": "noaa_swpc_g8_particle_5m",
    "storage_id": "ioGuard",
    "file_format": [
      "%Y%m%d_G8part_5m.txt"
    ],
    "institute": "",
    "observatory": "",
    "satellite": "",
    "model": "",
    "telescope": "",
    "wavelength": "",
    "channel": "",
    "instrument": "",
    "file_server": 0
  },
  {
    "data_id": "noaa_swpc_g10_particle_5m",
    "storage_id": "ioGuard",
    "file_format": [
      "%Y%m%d_G10part_5m.txt"
    ],
    "institute": "",
    "observatory": "",
    "satellite": "",
```

```

        "model": "",
        "telescope": "",
        "wavelength": "",
        "channel": "",
        "instrument": "",
        "file_server": 0
    },
    {
        "data_id": "noaa_swpc_g11_particle_5m",
        "storage_id": "ioGuard",
        "file_format": [
            "%Y%m%d_G11part_5m.txt"
        ],
        "institute": "",
        "observatory": "",
        "satellite": "",
        "model": "",
        "telescope": "",
        "wavelength": "",
        "channel": "",
        "instrument": "",
        "file_server": 0
    },
    {
        "data_id": "noaa_swpc_g12_particle_5m",
        "storage_id": "ioGuard",
        "file_format": [
            "%Y%m%d_G12part_5m.txt"
        ],
        "institute": "",
        "observatory": "",
        "satellite": "",
        "model": "",
        "telescope": "",
        "wavelength": "",
        "channel": "",
        "instrument": "",
        "file_server": 0
    },
    {
        "data_id": "noaa_swpc_gp_particle_5m",
        "storage_id": "ioGuard",
        "file_format": [

```

```

        "%Y%m%d_Gp_part_5m.txt"
    ],
    "institute": "",
    "observatory": "",
    "satellite": "",
    "model": "",
    "telescope": "",
    "wavelength": "",
    "channel": "",
    "instrument": "",
    "file_server": 0
},
{
    "data_id": "noaa_swpc_gs_particle_5m",
    "storage_id": "ioGuard",
    "file_format": [
        "%Y%m%d_Gs_part_5m.txt"
    ],
    "institute": "",
    "observatory": "",
    "satellite": "",
    "model": "",
    "telescope": "",
    "wavelength": "",
    "channel": "",
    "instrument": "",
    "file_server": 0
}
]
},
{
    "data_path":
"/NAS/ioGuard/vol7/swc/data/noaa/swpc/goes/xray",
    "data_group": "noaa_goes_xray",
    "storage_id": "ioGuard",
    "member_info": [
        {
            "data_id": "noaa_swpc_g8_xray_1m",
            "storage_id": "ioGuard",
            "file_format": [
                "%Y%m%d_G8xr_1m.txt"
            ],
            "institute": "",

```

```

        "observatory": "",
        "satellite": "",
        "model": "",
        "telescope": "",
        "wavelength": "",
        "channel": "",
        "instrument": "",
        "file_server": 0
    },
    {
        "data_id": "noaa_swpc_g10_xray_1m",
        "storage_id": "ioGuard",
        "file_format": [
            "%Y%m%d_G10xr_1m.txt"
        ],
        "institute": "",
        "observatory": "",
        "satellite": "",
        "model": "",
        "telescope": "",
        "wavelength": "",
        "channel": "",
        "instrument": "",
        "file_server": 0
    },
    {
        "data_id": "noaa_swpc_g11_xray_1m",
        "storage_id": "ioGuard",
        "file_format": [
            "%Y%m%d_G11xr_1m.txt"
        ],
        "institute": "",
        "observatory": "",
        "satellite": "",
        "model": "",
        "telescope": "",
        "wavelength": "",
        "channel": "",
        "instrument": "",
        "file_server": 0
    },
    {
        "data_id": "noaa_swpc_g12_xray_1m",

```

```
"storage_id": "ioGuard",
"file_format": [
    "%Y%m%d_Gl2xr_1m.txt"
],
"institute": "",
"observatory": "",
"satellite": "",
"model": "",
"telescope": "",
"wavelength": "",
"channel": "",
"instrument": "",
"file_server": 0
},
{
    "data_id": "noaa_swpc_gp_xray_1m",
    "storage_id": "ioGuard",
    "file_format": [
        "%Y%m%d_Gp_xr_1m.txt"
    ],
    "institute": "",
    "observatory": "",
    "satellite": "",
    "model": "",
    "telescope": "",
    "wavelength": "",
    "channel": "",
    "instrument": "",
    "file_server": 0
},
{
    "data_id": "noaa_swpc_gp_xray_5m",
    "storage_id": "ioGuard",
    "file_format": [
        "%Y%m%d_Gp_xr_5m.txt"
    ],
    "institute": "",
    "observatory": "",
    "satellite": "",
    "model": "",
    "telescope": "",
    "wavelength": "",
    "channel": "",
```

```

        "instrument": "",
        "file_server": 0
    }
}
]
}
]
}
}

```

table_name

DB에 만들어질 테이블 이름. 예를들어 `metadata_noaa_goes`라고 작성할 경우 DB의 metadata schema안에 해당 이름의 테이블이 만들어지게 됨 (`metadata.metadata_noaa_goes`)

scan_flag

최초 DB 생성시 `true`로 설정, `data_info` 리스트에 지정된 모든 `data_path` 아래에 있는 모든 파일을 스캔하면서 지정된 `file_format`과 맞는 파일정보를 DB에 업데이트함. `false`로 설정할 경우 해당과정을 하지 않고, DB에 있는 파일 정보를 이용하여

1. 저장된 file_path에 파일이 없다면 DB에 관련 행 삭제
2. DB에 저장되어 있는 값이 file_format과 다르다면 삭제
3. DB에 저장되어 있는 가장 나중의 modification time보다 더 나중에 만들어진/수정된 파일이 있으면 DB에 반영 되는 작업만 함.

17만개 파일 기준으로 `true`면 대략 8분, `false`면 대략 30초의 초기 실행시간(프로세스가 running에서 sleeping으로 상태가 바뀔때까지의 시간)이 걸림.

observer_flag

지정된 `data_path`를 계속 지켜보면서 수정사항이 발생했을 시 DB에 반영하고 싶으면 `true`로 설정. 이럴경우 프로세스가 `sleeping`상태로 background에 존재하게 됨. 해당 `data_path`에 저장된 파일들이 전혀 업데이트 되고 있지 않아 background process가 필요없다면 `false`로 설정

`scan_flag`와 `observer_flag`가 모두 false일 경우 파일 스캔 작업은 하지 않고 DB에 저장된 테이블의 값을 주어진 설정파일을 이용해서 update만 함.

data_info

아래와 같은 형식의 dictionary들의 리스트.

```

{
    "data_path": "/NAS/ioGuard/vol7/swc/data/noaa/swpc/goes/geomag",
    "data_group": "noaa_goes_geomag",
    "member_info": [<dictionary들의 리스트>]
}

```

data_path

데이터가 저장되어 있는 폴더 경로

`data_group`

해당 폴더에 대응되는 data_group 명을 입력

data.data_master							
데이터 목록							
Name	Data type	PK	FK	UQ	Not null	Default value	Description
data_id	varchar(30)	✓			✓		데이터 아이디
data_group	varchar(30)						데이터 그룹

`data.data_master` 테이블에 들어가는 값

`member_info`

아래와 같은 형식의 dictionary들의 리스트.

```
{
  "data_id": "noaa_swpc_g9_particle_5m",
  "storage_id": "ioGuard",
  "file_format": [
    "%Y%m%d_G9part_5m.txt"
  ],
  "institute": "",
  "observatory": "",
  "satellite": "",
  "model": "",
  "telescope": "",
  "wavelength": "",
  "channel": "",
  "instrument": "",
  "file_server": 0
}
```

데이터 정보

Name	Data type	PK	FK	UQ	Not null	Default value	Description
data_id	varchar (30)	✓	✓		✓		데이터 아이디
institute	varchar (30)						기관명
observatory	varchar (50)						관측기명
satellite	varchar (50)						위성명
model	varchar (50)						모델명
telescope	varchar (50)						관측기명
wavelength	varchar (50)						wavelength
channel	varchar (50)						channel
instrument	char (15)						기기명

storage_id

앞선 설정파일에서 hardware.hardware_storage 테이블 관련 설정 파일 (cfg_storage.json)에 작성한 storage_id 중 해당 데이터가 저장되어 있는 스토리지에 해당되는 storage_id를 입력해주면 된다.

file_format

DB에 정보를 업로드할 파일이름형식들을 리스트로 지정하면 된다.

```
{
  "file_format": [
    "kanz_halph_fl_%Y%m%d_%H%M%S.jpg",
    "kanz_halph_fr_%Y%m%d_%H%M%S.fts.gz",
    "kanz_halph_fr_%Y%m%d_%H%M%S.jpg"
  ]
}
```

날짜와 시간관련 형식은 "Python strftime() and strptime() Format Codes" 를 따라 작성하면 된다.

YYYYMMDD_hhmmss = %Y%m%d_%H%M%S

Directive	Meaning	Example
%Y	Year with century as a decimal number.	0001, 0002, ..., 2013, 2014, ..., 9998, 9999
%m	Month as a zero-padded decimal number.	01, 02, ..., 12
%d	Day of the month as a zero-padded decimal number.	01, 02, ..., 31

Directive	Meaning	Example
<code>%H</code>	Hour (24-hour clock) as a zero-padded decimal number.	00, 01, ..., 23
<code>%M</code>	Minute as a zero-padded decimal number.	00, 01, ..., 59
<code>%S</code>	Second as a zero-padded decimal number.	00, 01, ..., 59

더 자세한 사항은 아래 링크 참조.

<https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes>