

# Key parts of LocalDB

---

Yuma Uematsu (KEK)

[yuma.uematsu@kek.jp](mailto:yuma.uematsu@kek.jp)

Database for ITk Pixel Module

localhost:5000/localdb/component?id=6635ca0094479d...

LocalDB Top | Cell-Loaded Modules | Modules | PCBs | Bare Modules | Sensors | QC Tests | YARR Scans | America/Los\_Angeles | admin

Top Page > Component List > 20 U PG M2 3603039 > QC Result

## 20 U PG M2 3603039

Outer-Quad-Module/ITkPix\_v2

ITkPD Component Page

### Component Information

Item	Value
Serial Number	20 U PG M2 3603039
Production DB Component ID	7dbe8c47d51ef338a9537eddc141e3d4
LocalDB Component ID	6635ca0094479d00359a578d
Component Type	module
Super-Component	No match.
Sub-Components	<ul style="list-style-type: none"> <li>Bare Module: 20 U PG B4 3309001</li> <li>FE Chip: 20 U PG FC 0135110</li> <li>FE Chip: 20 U PG FC 0135094</li> <li>PCB: 20 U PG PQ 2603039</li> <li>FE Chip: 20 U PG FC 0135081</li> <li>FE Chip: 20 U PG FC 0135100</li> </ul>
Flags	

### Properties

Item	Data Type	Value
FE chip version	codeTable	ITkpix_v2
PCB-Bare Orientation isNormal	boolean	Yes
Wirebond protection roof presence	boolean	N/A
IREF Trim Bit FE1	integer	
IREF Trim Bit FE2	integer	
IREF Trim Bit FE3	integer	
IREF Trim Bit FE4	integer	

Update Properties

### Comments

Date	Commented by	on which Component	Comment
	admin		Fill your comment here...

Save

### Current Stage: Initial Warm

Stage Sign-off | Switch to alternative stage | Switch to previous stage

Test Type	Local Result	Action
Electrical Test (e-test) Module Summary (E_SUMMARY)	No TestRun!	Create Summary
IV measurement (IV_MEASURE)	No TestRun!	Submit RAW Result

Candidate TestRuns of this Stage

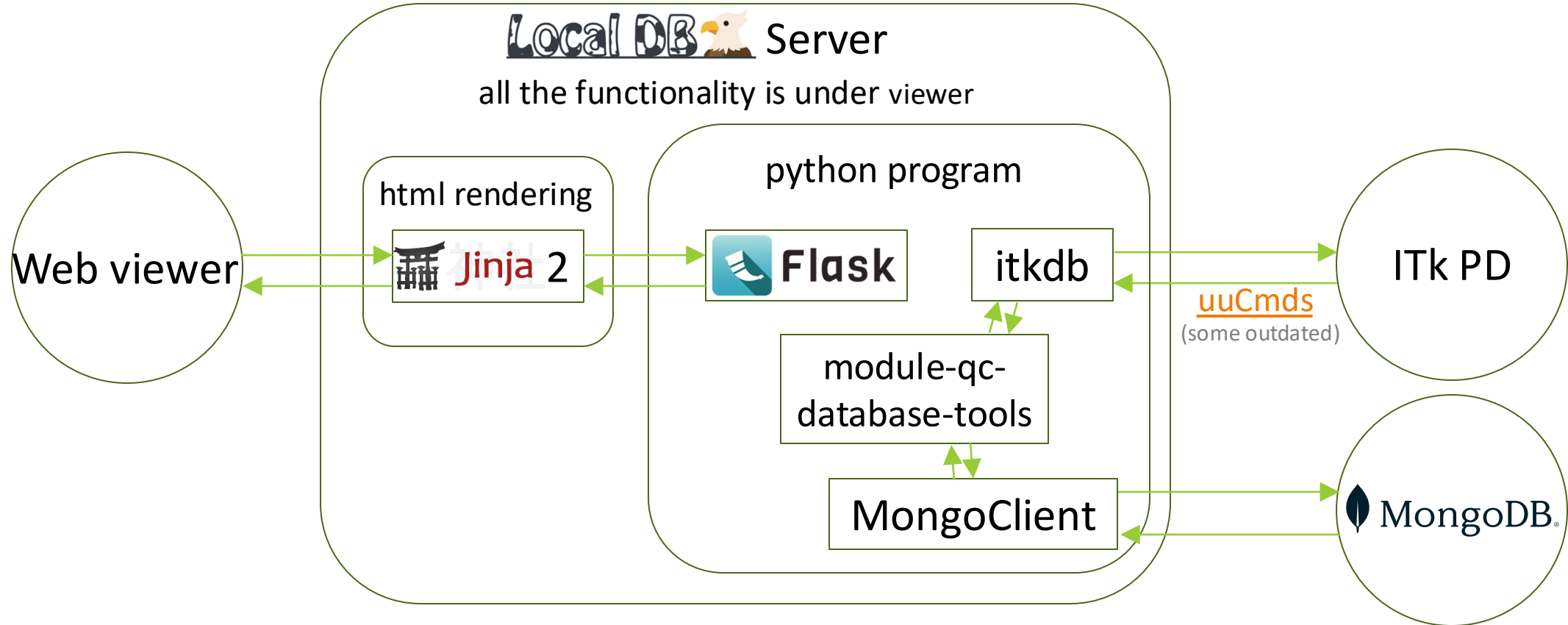
Past QC Stages and Results » Submit to ITkPD

Focusing on Parts.

LocalDB provides

- Web viewer
- interfacing “local database”
- interacting with ITk PD

# Illustrate a bit more...



# interaction with ITk PD

---

- ◆ functionalities in functions/itkpd\_interface based on itkdb
  - recursiveComponentSynchronizer for pulling components from PDB
  - recursiveUploader to update PDB with Local status

## functions/common.py

```
def process_request(code1, code2):
    global pd_client

    success = 0
    try:
        user = itkdb.core.User(access_code1=code1, access_code2=code2)
        pd_client = itkdb.Client(user=user, use_eos=True)
        mount_retry_adapter(pd_client)
        :
```

## functions/itkpd\_interface/PDInterface.py

```
class PDInterface:

    def getCompFromProdDB(self, component_id):
        return self.pd_client.get("getComponent", json={"component": component_id})
```

## functions/itkpd\_interface/recursiveUploader.py

```
class RecursiveUploader(PDInterface.PDInterface):
    def __submitTest(self, serialNumber, stage, test):
        :

        # submit a new TestRun
        tr_new = self.pd_client.post("uploadTestRunResults", json=out)
```

# MongoDB

---

- ◆ “local database” is a MongoDB
  - data (=document) is saved in JSON-like binary, BSON
  - you can do whatever you like
    - need to respect the data format defined by ourselves to keep integrity
- ◆ Document Type
  - component
  - parent-child relation
  - QC status
  - test
  - message
  - ...

## Example: QC status

```
{
  'QC_results': {...,
    'MODULE/INITIAL_WARM': {
      'E_SUMMARY': '65bccdc72d1ca36262b8bb53',
      'IV_MEASURE': '-1'
    }, ...
  },
  'QC_results_pdb': {...,
    'MODULE/INITIAL_WARM': {
      'E_SUMMARY': '664d04fdad15db00355cb2dc',
      'IV_MEASURE': '-1'
    }, ...
  },
  '_id': ObjectId('6643d220f60bf95d7acae221'),
  'component': '65a133c1bc46d00042ed96fd',
  'componentType': 'MODULE',
  ...,
  'stage': 'MODULE/FINAL_WARM',
  ...,
  'upload_status': {..., 'MODULE/INITIAL_WARM': '1', ...}
}
```

## On the Web viewer



Stage: Initial Warm (MODULE/INITIAL\_WARM) Complete

Expand Results

Test	Local Result	ProdDB Record
Electrical Test (e-test) Module Summary (E_SUMMARY)	<span>Registered</span>	<span>Registered</span>
IV measurement (IV_MEASURE)	<span>No result</span>	<span>N/A</span>

# MongoClient

---

## ◆ MongoDB can be handled with MongoClient

### □ find existing document (find, find\_one)

- e.g. by ID

### □ update existing document

### □ delete existing document

### □ insert new document

```
pages/qc.py
def save_raw_and_analyze(test, component):
    :
    rawHash = hashlib.md5(json.dumps(test).encode()).hexdigest()
    :
    rawRecord = localdb.QC.testRAW.find_one({"rawHash": rawHash, "stage": stage})

    if rawRecord is None:
        raw_id = localdb.QC.testRAW.insert_one(
            {"raw": raw, "rawHash": rawHash, "stage": stage}
        ).inserted_id
    :
```

# mongosh

---

## ◆ mongosh allows interactive access to MongoDB

```
% MONGO
MongoDB shell version v5.0.21
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("42c14f1c-a138-4e08-b99e-c79e053df40b") }
MongoDB server version: 7.0.2
WARNING: shell and server versions do not match
=====
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility. The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
=====
---
The server generated these startup warnings when booting:
  2025-01-15T03:22:49.138-08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
> show dbs
admin      0.000GB
config     0.000GB
local      0.000GB
localdb    3.099GB
localdbtools 0.002GB
> use localdb
switched to db localdb
```



# mongosh

## ◆ mongosh allows interactive access to MongoDB

- note: `find_one` (MongoClient) vs. `findOne` (mongosh)

```
> db.getCollectionNames()
[
  "QC.module.status",
  "QC.result",
  "QC.testRAW",
  "QC.testRuns_pdb_ldb_map",
  "childParentRelation",
  "component",
  "componentTestRun",
  "config",
  "fe_config_revision",
  "fe_configs",
  "fs.chunks",
  "fs.files",
  "institution",
  "pd.institution",
  "testRun",
  "user"
]
```

```
## obtain component ID from serial number
> db.component.find({"serialNumber": "20UPGM22601095"})
{ "_id" : ObjectId("6572c6066316ce00414314bb"), "name" : "20UPGM22601095", "chipType" : "2", "serialNumber" :
"20UPGM22601095", ...

## obtain QC status from component ID
> db.QC.module.status.find({"component": "6572c6066316ce00414314bb"})
{ "_id" : ObjectId("671d390214877824fc765aae"), "sys" : { "mts" : ISODate("2025-01-16T01:55:32.018Z"), "cts" : ISODate("2025-01-
16T01:55:32.018Z"), "rev" : 0 }, "dbVersion" : 1.01, "proddbVersion" : 1.02, "componentType" : "MODULE", "component" :
"6572c6066316ce00414314bb", "stage" : "MODULE/QC_STATUS", "status" : "created", "rework_stage" : [], "QC_results" : {
"MODULE/INIT" : { }, "MODULE/ASSEMBLY" : { "MASS_MEASUREMENT" : "-1", "GLUE_MODULE_FLEX_ATTACH" :
"6576611bd24c620783d80c61", "TRIPLT_METROLOGY" : "-1", "VISUAL_INSPECTION" : "65765dc1d24c620783d80c56",
"FLATNESS" : "-1", "QUAD_MODULE_METROLOGY" : "673ffd065f5366e933e0f2a9" }, "MODULE/WIREBONDING" : {
"WIREBONDING" : "65766848d24c620783d80c7b", "WIREBOND_PULL_TEST" : "673d7fd10ba3326c369ba3cb",
"VISUAL_INSPECTION" : "657666a7d24c620783d80c77" }, "MODULE/INITIAL_WARM" : { "E_SUMMARY" :
"66785a1146198bd10c3aa535", "IV_MEASURE" : "66a5b84b08600a2577c37ec1" }, ...

## list IV_MEASURE related to the component
> db.QC.result.find({"component": "6572c6066316ce00414314bb", "testType": "IV_MEASURE"})
```

# Web viewer: html

## ◆ page rendered based on Jinja2 html template (templates)

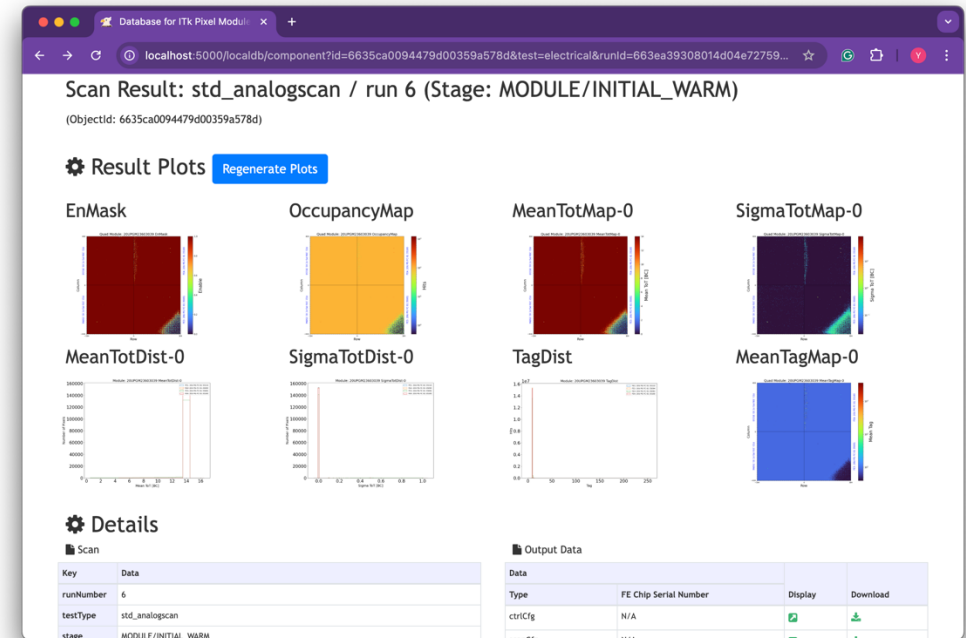
templates/yarr\_scans/plot.html

```
<form ...>
<input type="hidden" name="regeneratePlots" value="true" placeholder="View" />
<h3>
  <i class="fa fa-cog"></i> Result Plots
  <button class="btn btn-primary" name="regeneratePlots" title="Click this button when e.g. cache is broken.">Regenerate Plots</button>
</h3>
</form>
```

```
<div class="row align-items-center justify-content-flex-start">
  <div class="col">
```

⋮

```
{% if 'plots_light' in component['electrical'] %}
  <div class="row align-items-top justify-content-center">
    {% for plot_key in component['electrical']['plots_light'] %}
      <div class="col" align="left">
        <h4>{{ plot_key }}</h4>
        <a href="{{ component['electrical']['plots_light'][plot_key] }}"
          ></a>
      </div>
    {% endfor %}
  </div>
{% endif %}
</div>
</div>
```



# Web viewer: python

## ◆ python script gives what is rendered in template using Flask (pages)

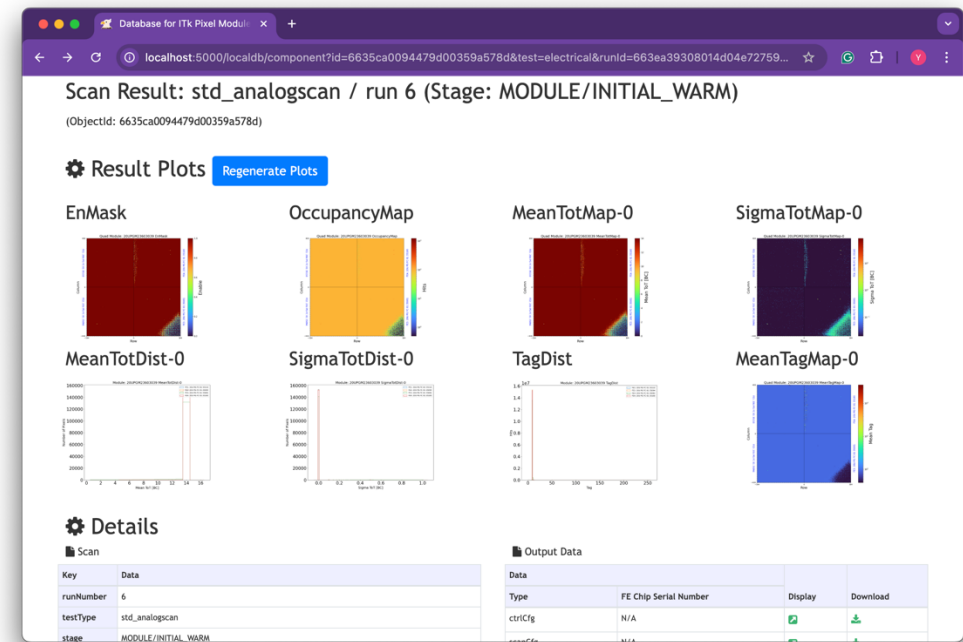
```
pages/component.py
# display component page
@component_api.route("/component", methods=["GET", "POST"])
def show_component():
    global plotting_mutex_lock
    ⋮

    try:
        plots_light = set_light_plots(
            session["this"],
            component_type,
            session["runId"],
            session["regeneratePlots"],
        )
        ⋮

    electrical = {
        ⋮
        "plots_light": plots_light,
    }

    component = {
        ⋮
        "electrical": electrical,
        ⋮
    }
```

```
⋮
return render_template(
    "component.html",
    component=component,
    table_docs=table_docs,
)
```



# Summary

---

## ◆ Key parts for LocalDB

- itkdb to communicate with ITk PD
- MongoClient to manipulate “local” MongoDB
- Flask + Jinja2 to render web page

# Backup

---