NSLS II MIDDLELAYER SERVICES*

Guobao Shen#, Yong Hu, Marty Kraimer, Shroff Kunal, BNL, Upton, NY 11973, USA
Dejan Dezman, Cosylab, Ljubljana, Slovenia

Abstract

A service-oriented architecture has been designed. It will be used for commissioning and daily operation of the NSLS II project. Middle layer services are being developed, and some have been deployed into NSLS II control network to support our beam commissioning. The services are primarily based on 2 technologies: web-service/RESTful and EPICS V4. The services provide functions to take machine status snapshot, convert magnet setting between different unit systems, and provide lattice information and simulation results. This paper presents the latest status of services, and our future development plan.

INTRODUCTION

Beginning with the early stage of NSLS II (National Synchrotron Light Source II) project [1], a middle layer service based architecture has been designed to support high level applications ranging from physics applications to control room beam operation. With this architecture, we are able to replace the more traditional monolithic high level application approach. This allows a narrow client API. With the narrow API, existing applications developed in different language under different architecture have been ported to our platform with minimum effort.

Some services already [2, 3] provide solid to support for high level physics application [4] development. Some are already deployed in our control network support beam commissioning and physics study.

This paper is arranged as below: 1) A description of the system architecture; 2) MASAR (MAchine Snapshot, Archiving, and Retrieve) service; 3) learning experience from NSLS II machine commissioning; 4) MUNICONV (Multiple UNIt CONVersion); 5) Lattice/Model service; and 6) summary and conclusion.

SYSTEM ARCHITECTURE

As described above, middle layer service has been developing to support high-level application from beam commissioning to beam operation. Like most modern control systems for large scale experimental facilities, the control system of NSLS II project is 3-tier based architecture as shown as Fig. 1.

The 3-tier consists of 1) distributed front-end layer, 2) middle service layer, and 3) application layer respectively. A description of these 3 tiers was discussed in [3], and the 2nd layer has been updated and revised during our development by introducing a HTTP/REST protocol.

*Work supported under auspices of the U.S. Department of Energy under Contract No. DE-AC02-98CH10886 with Brookhaven Science Associates, LLC, and in part by the DOE Contract DE-AC02-76SF00515
#shengb@bnl.gov

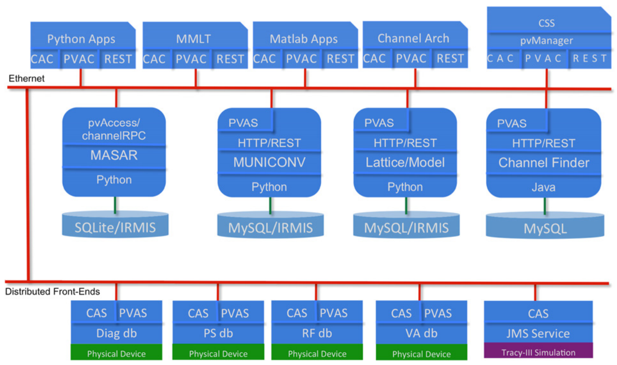


Figure 1: System architecture.

MASAR SERVICE

MASAR service takes a machine snapshot with pre-configuration, archives a snapshot, and retrieves data back for post analysis, and/or to restore machine to a particular state. The architecture is illustrated in Fig. 2.

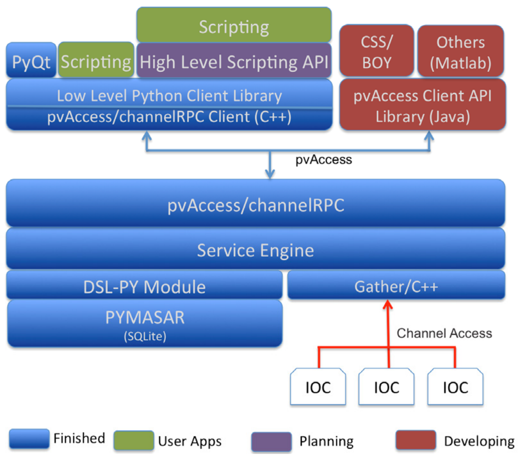


Figure 2: MASAR service.

Service Implementation

As shown in Fig.2, MASAR service uses SQLite [5] as the RDB (Relational Database) to store configuration and snapshot data. Access to SQLite is via a Python based API library called PYMASAY.

MASAR takes a machine snapshot using a pre-defined configuration. Each configuration is a collection of EPICS PV (Process Variable) names, Each configuration is saved into the RDB.

MASAR supports all scalar and waveform PV types e.g. float, double, string, and enum. Each configuration could be a mix of any of the types.

ISBN 978-3-95450-139-7

Commands from client are analysed by the service engine, which is implemented via EPICS V4 framework [6] in C++. A DSL-PY (Data Service Layer in Python) module is provided to manage data between C++ and Python Domain.

With a pre-defined configuration, a client can request MASAR to take a machine snapshot. Once the service engine gets a request, it retrieves the PV list for given configuration from SQLite, requests Gather library (a local Channel Access library implemented in C++ to gather PV data) to get data for each PV, and saves data into RDB via PYMASAR. In addition to the value, the connection status, time stamp, alarm severity, and alarm status for each PV is also saved.

MASAR provides a mechanism to allow user to preview a saved data set to make sure the data set is exact what is desired. By default, MASAR flags a data set as not approved. User has to explicitly approve one data set as good.

The data is transferred over network using V4 communication mechanism of pvAccess/channelRPC.

As shown in Fig. 2, MASAR only reads value from IOC, i.e, a restore function is left out of server side. This keeps the service simple and general because restoring procedures are usually complicated, and vary from system to system, facility to facility. A basic restore function is implemented inside MASR UI.

Client API

MASAR client library provides 7 APIs in Python as shown in Table 1.

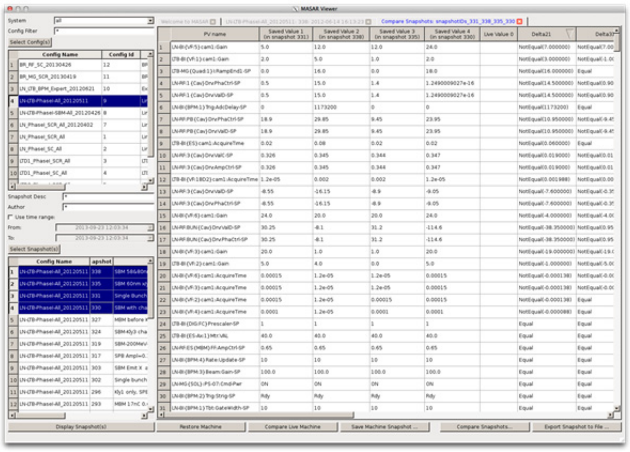
Table 1: MASAR Client Interface

API	Description
retrieveSystemList	Retrieve a list of system
retrieveServiceConfigs	Retrieve a list of configuration
retrieveServiceEvents	Retrieve snapshot list w/o data
retrieveSnapshot	Retrieve one snapshot with data
saveSnapshot	Save/take a snapshot
updateSnapshotEvent	Update/Approve snapshot
getLiveMachine	Get machine live data

Those 7 APIs provide interfaces for communication with MASAR. As shown in Fig. 2, user can use these APIs directly in their application, and a high level API is planned for next version for functions like comparing multiple data sets. Integration with CSS (Control System Studio) is under development.

User Interface

A PyQt4 based user interface has been developed on the top of client API, and is serving our beam commissioning. It provided a convenient way to the end user to take machine snapshots, browser data, and compare data with either live machine, or archived snapshot data set. An example is illustrated as Fig. 3.


MASAR Viewer
Welcome to MASAR
UN-LTB-PhaseI-All_20120511: 338: 2012-06-14 16:13:23
Compare Snapshots: snapshotIDs_331_338_335_330
Display Snapshot(s)
Restore Machine
Compare Live Machine
Save Machine Snapshot ...
Compare Snapshots...
Export Snapshot to File ...

System: all
Config Filter: *
Select Config(s)

	Config Name	Config Id	
1	BR_RF_SC_20130426	12	BP
2	BR_MG_SCR_20130419	11	BP
3	LN_LTB_BPM_Expert_20120621	10	Ex
4	LN-LTB-PhaseI-All_20120511	9	Lir
5	LN-LTB-PhaseI-SBM-All_20120426	8	Lir
6	LN_PhaseI_SCR_All_20120402	7	Lir
7	LN_PhaseI_SCR_All	1	Lir
8	LN_PhaseI_SC_All	2	Lir
9	LTD1_PhaseI_SCR_All	3	LTI
10	LTD1_PhaseI_SC_All	4	LTI

Snapshot Desc: *
Author: *
Use time range:
From: 2013-09-23 12:03:34
To: 2013-09-23 12:03:34
Select Snapshot(s)

	Config Name	apshot	
1	LN-LTB-PhaseI-All_20120511	338	SBM 586/80n
2	LN-LTB-PhaseI-All_20120511	335	SBM 60nm x/
3	LN-LTB-PhaseI-All_20120511	331	Single Bunch
4	LN-LTB-PhaseI-All_20120511	330	SBM with cha
5	LN-LTB-PhaseI-All_20120511	327	MBM before
6	LN-LTB-PhaseI-All_20120511	324	SBM-Kly3 cha
7	LN-LTB-PhaseI-All_20120511	319	SBM-200MeV-
8	LN-LTB-PhaseI-All_20120511	317	SPB Ampl=0.
9	LN-LTB-PhaseI-All_20120511	303	SBM Emit X a
10	LN-LTB-PhaseI-All_20120511	302	Single bunch
11	LN-LTB-PhaseI-All_20120511	296	Kly1 only. SPE
12	LN-LTB-PhaseI-All_20120511	293	MBM 17nC 0.

	PV name	Saved Value 1 (in snapshot 331)	Saved Value 2 (in snapshot 338)	Saved Value 3 (in snapshot 335)	Saved Value 4 (in snapshot 330)	Live Value 0	Delta21	Delta3
1	LN-BI{VF:5}-cam1:Gain	5.0	12.0	12.0	24.0		NotEqual(7.000000)	NotEqual(7.00
2	LTB-BI{VF:1}cam1:Gain	2.0	5.0	1.0	2.0		NotEqual(3.000000)	NotEqual(-1.0
3	LTB-MG{Quad:1}I:RampEnd1-SP	0.0	16.0	0.0	18.0		NotEqual(16.000000)	Equal
4	LN-RF:1{Cav}DrvPhaCtrl-SP	0.5	15.0	1.4	1.2490009027e-16		NotEqual(14.500000)	NotEqual(0.90
5	LN-RF:1{Cav}DrvValD-SP	0.5	15.0	1.4	1.2490009027e-16		NotEqual(14.500000)	NotEqual(0.90
6	LN-BI{BPM:1}Trig:AdcDelay-SP	0	1173200	0	0		NotEqual(1173200)	Equal
7	LN-RF:PB{Cav}DrvPhaCtrl-SP	18.9	29.85	9.45	23.95		NotEqual(10.950000)	NotEqual(-9.45
8	LN-RF:PB{Cav}DrvValD-SP	18.9	29.85	9.45	23.95		NotEqual(10.950000)	NotEqual(-9.45
9	LTB-BI{ES}cam1:AcquireTime	0.02	0.08	0.02	0.02		NotEqual(0.060000)	Equal
10	LN-RF:3{Cav}DrvValC-SP	0.326	0.345	0.344	0.347		NotEqual(0.019000)	NotEqual(0.01
11	LN-RF:3{Cav}DrvAmpCtrl-SP	0.326	0.345	0.344	0.347		NotEqual(0.019000)	NotEqual(0.01
12	LTB-BI{VF:1BD2}cam1:AcquireTime	1.2e-05	0.002	0.002	1.2e-05		NotEqual(0.001988)	NotEqual(0.00
13	LN-RF:3{Cav}DrvValD-SP	-8.55	-16.15	-8.9	-9.05		NotEqual(-7.600000)	NotEqual(-0.35
14	LN-RF:3{Cav}DrvPhaCtrl-SP	-8.55	-16.15	-8.9	-9.05		NotEqual(-7.600000)	NotEqual(-0.35
15	LN-BI{VF:6}-cam1:Gain	24.0	20.0	20.0	24.0		NotEqual(-4.000000)	NotEqual(-4.00
16	LN-RF:BUN{Cav}DrvValD-SP	30.25	-8.1	31.2	-114.6		NotEqual(-38.350000)	NotEqual(0.95
17	LN-RF:BUN{Cav}DrvPhaCtrl-SP	30.25	-8.1	31.2	-114.6		NotEqual(-38.350000)	NotEqual(0.95
18	LN-BI{VF:3}-cam1:Gain	20.0	1.0	1.0	20.0		NotEqual(-19.000000)	NotEqual(-19.0
19	LTB-BI{VF:2}cam1:Gain	5.0	4.0	0.0	5.0		NotEqual(-1.000000)	NotEqual(-5.00
20	LN-BI{VF:6}-cam1:AcquireTime	0.00015	1.2e-05	1.2e-05	0.00015		NotEqual(-0.000138)	NotEqual(-0.00
21	LN-BI{VF:3}-cam1:AcquireTime	0.00015	1.2e-05	1.2e-05	0.00015		NotEqual(-0.000138)	NotEqual(-0.00
22	LN-BI{VF:2}-cam1:AcquireTime	0.00015	1.2e-05	0.00015	0.00015		NotEqual(-0.000138)	Equal
23	LN-BI{VF:4}-cam1:AcquireTime	0.0001	1.2e-05	0.0001	0.0001		NotEqual(-0.000088)	Equal
24	LTB-BI{DIG:FC}Prescaler-SP	1	1	1	1		Equal	Equal
25	LTB-BI{ES-Ax:1}Mtr.VAL	40.0	40.0	40.0	40.0		Equal	Equal
26	LN-RF:ES{MBM}FF:AmpCtrl-SP	0.65	0.65	0.65	0.65		Equal	Equal
27	LN-BI{BPM:4}Rate:Update-SP	10	10	10	10		Equal	Equal
28	LN-BI{BPM:3}Beam:Gain-SP	100.0	100.0	100.0	100.0		Equal	Equal
29	LN-MG{SOL}:PS-07:Cmd-Pwr	ON	ON	ON	ON		Equal	Equal
30	LN-BI{BPM:2}Trig:Strig-SP	Rdy	Rdy	Rdy	Rdy		Equal	Equal
31	LN-BI{BPM:1}Tbt:GateWidth-SP	10	10	10	10		Equal	Equal

Figure 3: MASAR UI.

The table of left pane upper window of Fig. 3 shows all configurations belong to one or all systems. User can select one or multiple configurations. Snapshot header information belonging to selected configuration(s) is then listed as shown in the bottom table of Fig. 3.

The user can either browse a saved data set, or compare them by selecting one or multiple snapshots. Fig. 3 shows a result to compare 4 data sets.

Since the restore function is left out of MASAR server side, a basic restore function is implemented in this UI:

- Mark PVs not to restore for those were not connected when the snapshot was taken;
- Check all PVs connectivity, and mark them as problem PVs if they can not be connected;
- Restore all PVs in good conditions, and recheck all problem PVs;
- Restore PVs if any problem PV is back to normal;
- Report the not restored PVs.

One more useful function provided by this UI is to allow user to export a data set into a CSV file. This is useful for performing off-line analysis.

Experience at NSLS II

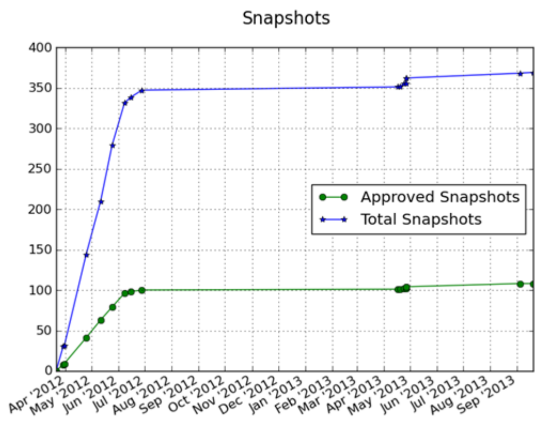


Figure 4: MASAR Snapshots analysis.

ISBN 978-3-95450-139-7

The MASAR service has been deployed into our control network, and has been used for beam commissioning since Apr 2012. Up to now, there were 369 snapshots taken, and 108 snapshots have been approved to be good as shown in Fig. 4.

Since all data is currently saved into SQLite database, a concern is the SQLite data file size growing with time. An analysis about the database size growing at NSLS II project has been monitored, and the result is shown as Fig. 5. The file size was taken during regular maintenance. Therefore, there might have more than one snapshot at each time point, or does not have anything.

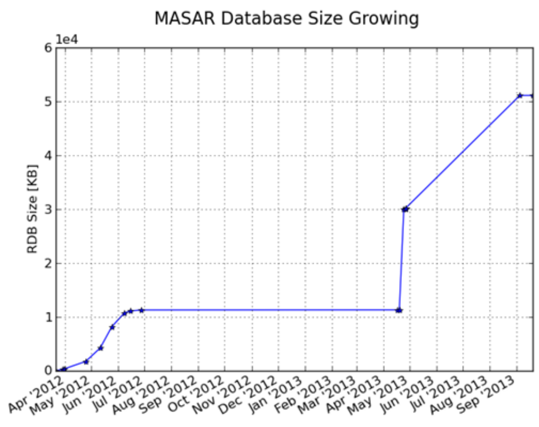


Figure 5: MASAR RDB size growing.

As shown in Fig. 5, there are 2 significant database size increases. Those increases were caused by 5 snapshots for power supply ramping table of our booster ring. That configuration contains over 60 PVs for booster ring power supplies, and each PV is a waveform with over 10,000 points.

Our experience with MASAR at NSLS II project shows that the current implementation can satisfy the requirements of beam commissioning even with many large waveform data. However, to solve this concern, we are planning to shift data set itself out of SQLite database. Since PYMASAR isolates underneath RDB and server, this change will be transparent to service engine, and therefore client.

MUNICONV SERVICE

Another recently released service is MUNICONV (multiple purpose unit conversion). This service is implemented as a RESTful web service using Django framework [7], and its architecture is illustrated as Fig. 6.

Service Implementation

The first implementation supports conversion of magnet values between 3 unit systems: 1) i value, which is engineering value, power supply current usually; 2) b value, which is magnetic field; and 3) k value, which is the value used by simulation code.

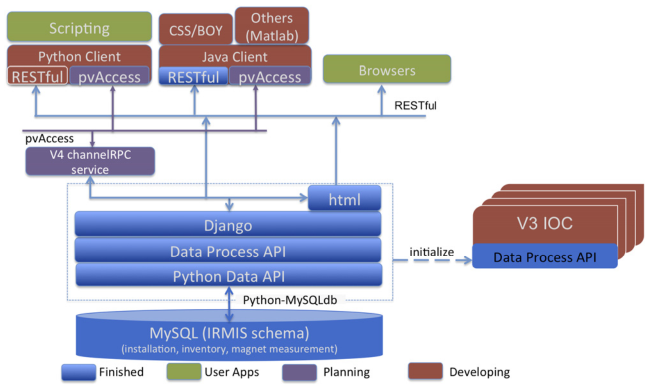


Figure 6: MUNICONV service.

As shown in Fig.6, MUNICONV service uses MySQL as the RDB (Relational Database). The following information is stored in RDB:

- Device information. A device can be in inventory or installed. If a device is in inventory, its serial number, and component type is required to find it. If a device has been installed already, it normally has a name and is assigned to a system. Thus a client can find it via the system name and device name.
- Magnet measurement data. This provides fundamental conversion information between engineering unit and magnetic field.
- Algorithm to perform a conversion. This information provides direct information for conversion value between (i, b, k) unit systems. According the requirement, the conversion is predefined between systems, and could use measurement data with a well-defined numerical algorithm, or an equation generated in advance.

A Python data API provides a full access to MySQL database. Like MASAR, this API layer isolates MySQL and the data process layer. It makes RDB schema changes transparent and provides flexibility to plug in another RDBMS.

The data processing layer responses commands from the upper layer, gets required data from database, and converts value from source unit to destination unit if needed information is available, and returns data back to the upper layer.

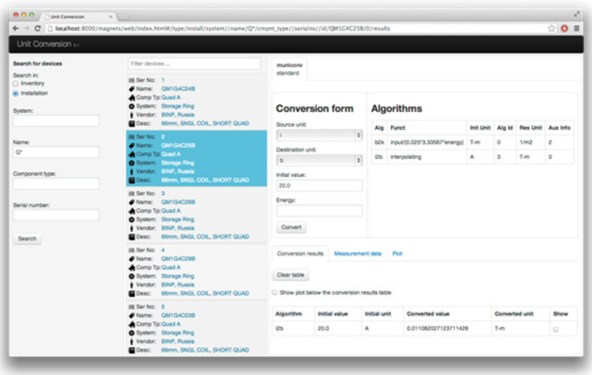
As shown in Fig. 6, the service under Django framework has been implemented, and a web interface has been developed. Meanwhile we are planning to embed the conversion function into a V3 IOC to conduct a real time conversion for our power supply.

The integration with EPICS V4 has been planned, and will be developed in its next release.

Client API and User Interface

A client can access MUNICONV service via RESTful interface, and 2 different implementations: 1) Python API, and 2) Web UI. Web UI access is demonstrated in Fig. 7.

ISBN 978-3-95450-139-7


Unit Conversion

localhost:8000/magnets/web/index.html#/type/install/system//name/Q*/cmpnt_type//serialno//id/QM1C4C258/0/results

Unit Conversion

Search for devices
Search in:
○ Inventory
◉ Installation
System:
Name: Q*
Component type:
Serial number:
Search

Filter devices ...

Ser No: 1
Name: QM1G4C24B
Comp Tp: Quad A
System: Storage Ring
Vendor: BINP, Russia
Desc: 66mm, SNGL COIL, SHORT QUAD

Ser No: 2
Name: QM1G4C25B
Comp Tp: Quad A
System: Storage Ring
Vendor: BINP, Russia
Desc: 66mm, SNGL COIL, SHORT QUAD

Ser No: 3
Name: QM1G4C26B
Comp Tp: Quad A
System: Storage Ring
Vendor: BINP, Russia
Desc: 66mm, SNGL COIL, SHORT QUAD

Ser No: 4
Name: QM1G4C29B
Comp Tp: Quad A
System: Storage Ring
Vendor: BINP, Russia
Desc: 66mm, SNGL COIL, SHORT QUAD

Ser No: 5
Name: QM1G4C03B
Comp Tp: Quad A
System: Storage Ring
Vendor: BINP, Russia
Desc: 66mm, SNGL COIL, SHORT QUAD

municonv standard

Conversion form
Source unit: i
Destination unit: b
Initial value: 20.0
Energy:
Convert

Algorithms

Alg	Funct	Init Unit	Alg Id	Res Unit	Aux Info
b2k	input/(0.025*3.33567*energy)	T-m	0	1/m2	2
i2b	interpolating	A	3	T-m	0

Conversion results | Measurement data | Plot

Clear table
☐ Show plot below the conversion results table

Algorithm	Initial value	Initial unit	Converted value	Converted unit	Show
i2b	20.0	A	0.011082027123711428	T-m	☐

Figure 7: MUNICONV Web UI.

A device can be searched via the inventory, or via installation as shown on left pane in Fig. 7. All devices found are listed in the middle pane. By selecting a particular device, its conversion information is displayed on the right pane. With the conversion information, user could select desired conversion between different unit systems, give an initial value, and request a conversion. The conversion result will be displayed in the result table.

Currently, all NSLS II magnets data (LINAC through storage ring) has been loaded into the database.

LATTICE/MODEL SERVICE

As its name indicated, this lattice/model service consists of 2 parts: lattice and model. The lattice part captures element geometric information (layout with misalignment), and magnetic strength of each element if it is provided. A lattice could be from any source such as design release, a study scenario, or a physical installation and real measurement.

The model part captures beam parameters such as twiss parameters, closed orbit, and transfer matrix for each element, and global parameters like tune, beam energy, chromaticity (up to 2nd order). It could be results from simulation or from measurement.

Service Implementation

Service is implemented under Django framework as illustrated in Fig. 8. The MySQL schema is a collaboration result [8]. In addition to capturing lattice/model results, this lattice/model service provides capability to conduct a quick simulation, and save the simulation result in model domain. Two simulation codes, Tracy-III and elegant, are supported.

Expanding to EPICS V4 is planned, and will be implemented in the near future.

Client API and User interface

The service provides a RESTful interface to client, and the development for Python API is on going, followed by Web UI, and Java API.

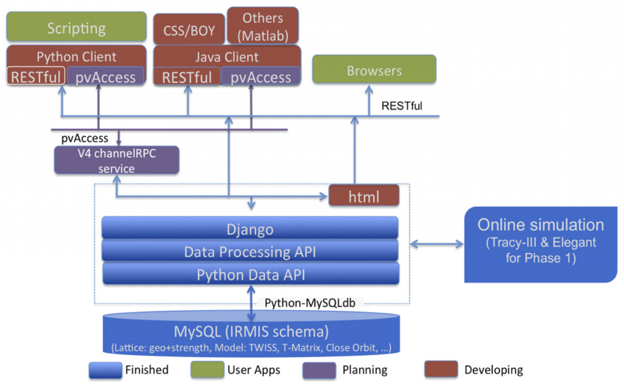


Figure 8: Lattice/model service.

SUMMARY

Middle layer services are being developed at NSLS II. Implementation details and status of the MASAR, MUNICONV, Lattice/Model, have been discussed. Experience using those services has been summarized. Future development plan has been discussed.

ACKNOWLEDGMENT

The authors would like to thank the EPICS V4 working group for their contributions, especially Matej Sekoranja at COSYLAY, and Michael Davidsaver at BNL. The authors also thank Don Dohan at BNL for his wonderful suggestion and kindness to share the IRMIS schema. Meanwhile, the authors want to thank the DISCS team for their helpful discussion on database. Finally, but not the last, the authors want to thank our colleagues at BNL for their suggestions, and feedback.

REFERENCES

[1] G. Shen, "A Software Architecture for High Level Applications", Proceedings of PAC09 (2009), May 2009, Vancouver Canada, FR5REP004
[2] G. Shen, et al., "NSLS-II High Level Application Infrastructure and Client API Design", Proceedings of PAC 2011, New York USA, 2011, MOP250, p. 582;
[3] G. Shen, et al., "Server Development for NSLS-II Physics Applications and Performance Analysis", Proceedings of PAC 2011, New York USA, 2011, MOP252, p. 585
[4] L. Yang, et al., "The Design of NSLS-II High Level Physics Applications", these proceedings, TUPPC130
[5] http://www.sqlite.org/
[6] T. Korhonen, et al., "EPICS V4 Progress Report", these proceedings, TUCOCB04
[7] https://www.djangoproject.com/
[8] P. Chu, et al., "Accelerator Lattice and Model Services", this proceeding, TUPPC152