ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-MOBPL04

MADOCA II DATA COLLECTION FRAMEWORK FOR SPring-8

T. Matsumoto†, Y. Furukawa, Y. Hamada,
Japan Synchrotron Radiation Research Institute, Hyogo, 679-5198, Japan

Abstract

MADOCA II (Message and Database Oriented Architecture II) is the next generation of the MADOCA control framework and has been implemented in accelerator and beamline control for the SPring-8 since 2014. In this paper, we report on the recent evolution in MADOCA II for data collection, which was missing in the past reports at ICALEPCS [1, 2]. To improve the management flexibility, we developed a data collection framework to manage various data collection types in SPring-8 with a unified method. All of the data collection methods (polling, event triggered type), data formats (such as point and waveform data) and platforms (UNIX, Embedded, and Windows including LabVIEW [3]) can be managed within the same framework. We also developed a signal registration procedure to facilitate the preparation for the data collection. In MADOCA, we managed all the parameters used in the data collections with an RDBMS, and requested the equipment manager to fulfil a Signal Registration Table (SRT) to update the data collection. However, this required an extensive work owing to inconsistencies in the SRT and the iteration of communications with the DB manager for the registration of the SRT into the RDBMS. In MADOCA II, we facilitated the signal registration procedure with a prior test of the data collection and a validity check in SRT with a web-based user interface. We started to implement the MADOCA II data collection into SPring-8 with 220 hosts and have confirmed stable operation since April 2016.

INTRODUCTION

MADOCA is a distributed control framework developed to control the SPring-8 accelerator and beamline. It has been adopted for this operation since 1997 [4]. Though we have experienced stable operation with MADOCA for more than 15 years, we have developed the next generation of MADOCA, called MADOCA II, which has been implemented to include new functions to cope with current requirements in the controls. MADOCA II has been adopted into the SPring-8 and SACLA DAQ system since 2014 [5, 6]. The schematic view of messaging control in MADOCA II is shown in Figure 1. MADOCA II performs text-based messaging controls with SVOC sentence structure for distributed controls, similar to MADOCA. However, much of the functionality has been implemented by introducing ZeroMQ [7] and MessagePack [8] as core messaging architectures. Flexibility in the messaging communication was improved to handle the capability to attach variable length data such as image data to the messages, and to add support for the Windows platform. Fast data logging was also archived by using NoSQL databases such as Redis [9] and Cassandra [10].

† matumot@spring8.or.jp

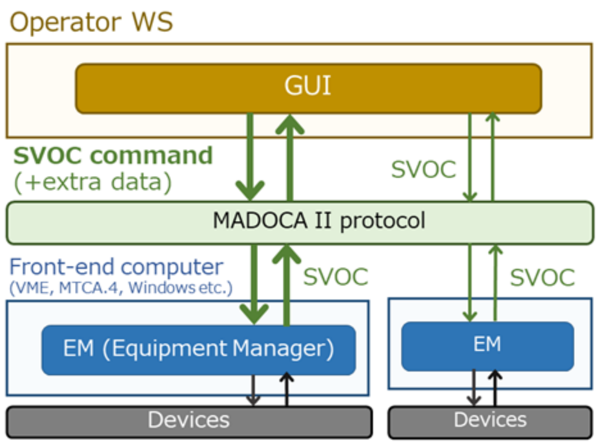


Figure 1: Schematic view of the software framework for messaging controls with MADOCA II.

At SPring-8, we operate data logging with about 500 hosts and 30 k signals. On average, about 9 k signals per second are collected, where intervals of data collections for signals vary from one second to 10 minutes. For the data logging, we archive the data for each signal with the signal name from the O/C in the SVOC command of MADOCA. For example, the signal name is set to “sr_mag_ps_a/voltage” for the signal data in the voltage of a power supply for a magnet in a storage ring.

Most of the data collections are operated with Poller/Collector applications [11]. The Poller/Collector periodically collects point data in several formats such as integers, or statuses with bit information and floats, which are applied to the data logging of vacuum pressure, temperature and voltage in magnet power supplies and so on. However, we have also encountered other data types in the data collection during the 20 years of operation at SPring-8. In a Linac accelerator, we encountered event triggered data collections synchronized in time with the injection beams of a different framework [11]. We also have other types of data collections such as bunch current measurements and closed orbit distortion (COD) in electron beams, which deal with the structure of the data format.

The schematic view of the data collection at SPring-8 is shown in Figure 2. We still use MADOCA for the data collection. However, data logging is upgraded to MADOCA II for high flexibility. Therefore, data collected with MADOCA is sent to the streamer in MADOCA II. After passing the data into the streamer, the data is archived into the NoSQL database through writers. These applications in

MADOCA II data logging, which has already been reported in the last ICALEPCS [2].

For the Poller/Collector, the poller application is used to perform periodic data collection and a collector client application is used to manage the operation of the data collection with the messaging commands of MADOCA. However, other data collections in Linac synchronized data and COD data are performed separately. Therefore, the management of these data collections is complicated and lacks operational flexibility.

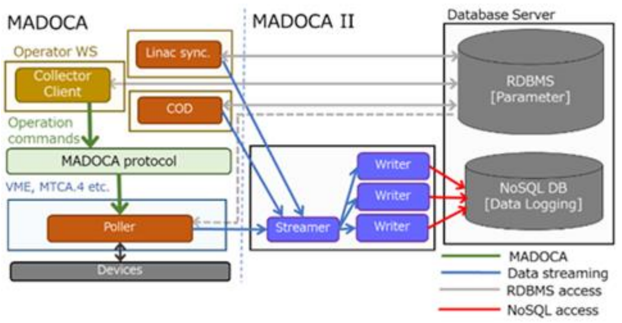


Figure 2: Data logging system at SPring-8.

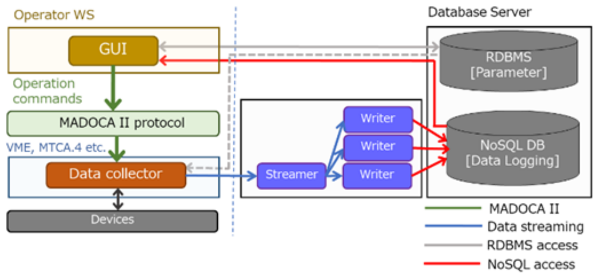


Figure 3: Data logging system with MADOCA II.

To solve the problem in these data collections and to have all systems controlled with MADOCA II, we developed a new MADOCA II data collection as shown in Figure 3. In the MADOCA II data collection, various data collection methods in SPring-8 can be managed with the same framework. Additionally, management of these data collections can be unified through the messaging commands of MADOCA II and the parameters in the RDBMS.

In the MADOCA II data collection, we also aimed to solve the elaborate problem of the registration of signal configurations in the RDBMS. At SPring-8, we managed all the parameters in the data collection with an RDBMS. The parameters were signal name, equipment group, data collection cycle, flag for off and decimation settings for signals, alarm settings and so on. We setup the Signal Registration Table (SRT) with an excel file to manage these parameters and asked the equipment manager to update the information. However, the required time and cost was high due to the difficulties in managing several parameters in SRT and the iterative communication process with the DB manager for the registration, due to inconsistencies in SRT.

We also encountered other issues in the data collection for MADOCA as shown below.

- Data collections were not supported for LabVIEW [12] and the Windows OS.
- The data collection operated with the group unit only in the Poller/Collector.

In the following sections, we report on the framework developed in the MADOCA II data collection to solve these problems. After that, we describe the implementation of the data collections in SPring-8.

MADOCA II DATA COLLECTION FRAMEWORK

In this section, we report on the MADOCA II data collection framework to resolve the problems described in the previous section. For the management of the data collection operations, we report the issue in the next section.

Signal Registration Procedure

The flow of the signal registration procedure is shown in Figure 4. The procedure is roughly the same as MADOCA. To facilitate this procedure, we added a web interface for the validity check of the SRT as well as to test the data collections in MADOCA II. The web interface for SRT is shown in Figure 5. We describe each step in the signal registration below.

- Generation of a template for SRT

To retain consistency for parameters in the RDBMS, we first generate a template for the SRT from the RDBMS. The equipment manager requests that the target hosts for updating the data collection, then the web interface of SRT generates the template for the corresponding hosts. SRT is composed of several sheets of configurations of signals, status bit information, alarm settings and so on. The web interface was built with ruby on rails [13]. Since we used Python to manage the database access with the RDBMS, we developed Python scripts for the management of the RDBMS and these scripts were also utilized inside the web interface.

- Update of SRT

After the generation of the template, the equipment manager edits SRT using the web interface. The update can be performed in a similar way as in an excel file. To add new signals in the data collection, we need to update SRT. To achieve consistency among new signals in the configuration table (config.tbl), used in the Equipment Manager (EM), we can input the config.tbl into the web interface and reflect the new signals in config.tbl to update SRT. After this procedure, the equipment manager updates the contents of SRT, if necessary. Since SRT contains a lot of information, we often find inconsistencies in the values in SRT. With the web interface, these inconsistencies can be clearly visualized. Therefore, the equipment manager can easily obtain consistency in SRT through corrections made in the web interface.

- Test of data collection

Even if SRT is prepared to have format consistency, the data collection itself may not work as expected because some parameters may be not appropriate for the operation. To confirm the validity of SRT, we ask the equipment manager to test the data collection in the test system. For the test, parameters in SRT are not registered in the RDBMS as during operation. However, we can perform the test by using a datacol file generated from SRT, which contains a signal list and related settings for the data collection. We also setup a NoSQL database for the test. Additionally, we setup tools for the test to view and browse the data in a web server similar to the capabilities available during operation. Therefore, we could easily test the data collection independently with the operation and it was very useful to have a valid data collection. For example, we could perform a test data collection in a machine study without updating the data collection for the operation in production.

- Registration of SRT into the RDBMS

After confirming the contents in SRT with the procedures above, the equipment manager notifies the DB manager that the SRT is prepared and ready by using a request button in the web interface. To reduce manual intervention, it may be possible to perform the signal registration with an automated procedure. However, the requested SRT may contain unexpected patterns such as new 10 k signals. To guarantee the stable operation of SPring-8, a DB manager checks the consistency in the SRT before the registration, then registers the SRT into the RDBMS if there are no problems with the contents.

- Start of the data collection

After the signal registration into the RDBMS, the equipment manager generates a datacol file for the operation from the RDBMS, then starts the data collection.

Previously, we sometimes required an iterative process to communicate between the equipment manager and the DB manager to fix inconsistencies in SRT. With the developed procedure in MADOCA II, we can facilitate the preparation of valid SRT data thanks to the web interface for modifying SRT and the test system for the data collection. We can then proceed to the signal registration procedure smoothly, as well as reduce the time and cost of the signal registration.

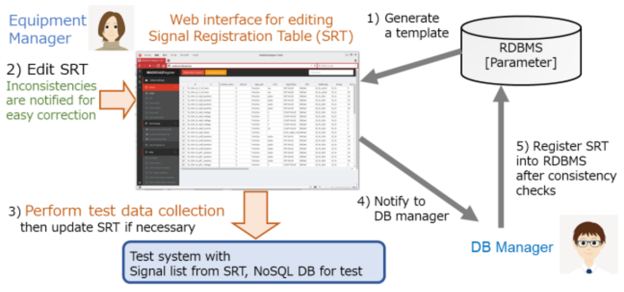
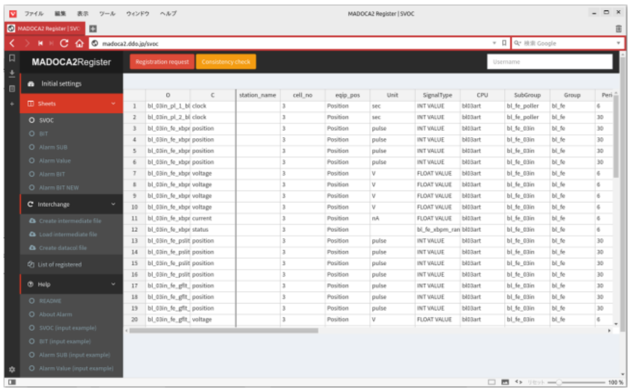

Equipment Manager
Web interface for editing Signal Registration Table (SRT)
1) Generate a template
RDBMS [Parameter]
2) Edit SRT
Inconsistencies are notified for easy correction
5) Register SRT into RDBMS after consistency checks
3) Perform test data collection then update SRT if necessary
4) Notify to DB manager
DB Manager
Test system with Signal list from SRT, NoSQL DB for test

Figure 4: Signal registration procedure in MADOCA II data collection.



	D	C	station_name	cell_no	eqip_pos	Unit	SignalType	CPU	SubGroup	Group	Peri
1	bl_03in_pl_1_bl	clock		3	Position	sec	INT VALUE	bl03art	bl_fe_poller	bl_fe	6
2	bl_03in_pl_2_bl	clock		3	Position	sec	INT VALUE	bl03art	bl_fe_poller	bl_fe	30
3	bl_03in_fe_xbpr	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
4	bl_03in_fe_xbpr	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
5	bl_03in_fe_xbpr	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
6	bl_03in_fe_xbpr	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
7	bl_03in_fe_xbpr	voltage		3	Position	V	FLOAT VALUE	bl03art	bl_fe_03in	bl_fe	6
8	bl_03in_fe_xbpr	voltage		3	Position	V	FLOAT VALUE	bl03art	bl_fe_03in	bl_fe	6
9	bl_03in_fe_xbpr	voltage		3	Position	V	FLOAT VALUE	bl03art	bl_fe_03in	bl_fe	6
10	bl_03in_fe_xbpr	voltage		3	Position	V	FLOAT VALUE	bl03art	bl_fe_03in	bl_fe	6
11	bl_03in_fe_xbpr	current		3	Position	nA	FLOAT VALUE	bl03art	bl_fe_03in	bl_fe	6
12	bl_03in_fe_xbpr	status		3	Position		bl_fe_xbpm_ran	bl03art	bl_fe_03in	bl_fe	6
13	bl_03in_fe_pslit	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
14	bl_03in_fe_pslit	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
15	bl_03in_fe_pslit	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
16	bl_03in_fe_pslit	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
17	bl_03in_fe_gflt_	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
18	bl_03in_fe_gflt_	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
19	bl_03in_fe_gflt_	position		3	Position	pulse	INT VALUE	bl03art	bl_fe_03in	bl_fe	30
20	bl_03in_fe_gflt_	voltage		3	Position	V	FLOAT VALUE	bl03art	bl_fe_03in	bl_fe	6

Figure 5: Web interface for editing SRT.

MADOCA II Data Collection

The main purpose of the MADOCA II data collection is to handle various data collections within the same framework to improve management flexibility. To achieve this goal, we developed the framework to implement the following features.

- Unification of management tables in RDBMS

In MADOCA, management tables in the RDBMS were different among the periodic type, event triggered type and the data collections with various data formats. Therefore, the management of these became complicated. To monitor the collections of signal data for different data types, users also require elaborate data analysis. To solve these problems, management tables were unified in MADOCA II to DATACOL_INF and SIG_DATACOL_INF. DATACOL_INF manages the information for each data collection, such as data collection type and managed hosts. SIG_DATACOL_INF manages the information for each signal, such as flags for off and decimation. Based on these unified tables, the DB manager can facilitate the management of the data collection. Users also deserve an easy analysis of the correlations among different types of signal data with a unified method.

- Support of various data collection methods

The unification of various data collections was also adopted for the MADOCA II data collection processes.

Figure 6 shows the case handling the data collections of the periodic type and event triggered type with the same process. In an EM base data collector (EMDC), data collection process collects the signal data from devices using functions prepared for an EM. Then, the data collected are sent to the streamer and written to a NoSQL logging database. The operation of the data collection can be managed with the message commands of MADOCA II from a remote Graphical User Interface (GUI). In the data collection process, threads are created that generate timing signals which interrupt and trigger the collection of signal data. The timing of the threads is generated with an internal timer for the data collection within a periodic cycle. In the case of an event-triggered data collection, the timing of the threads is generated with an interrupted function triggered with the event.

In the case of the data collection with EMDC, the application needs to be written in the C programming language. In fact, we have found it difficult to implement data collections in other languages. To apply data collections for various program interfaces, we developed data collections with a messaging based data collector (MSDC) as shown in Figure 7. In the case of MSDC, the data collection process collects signals data from other EMs through messaging. Since an EM can be flexibly developed with multiple languages such as Python and LabVIEW for MADOCA II [14], we can manage the data collections with various types of applications. MSDC can perform data collection within a periodical cycle. It is also possible to drive the data collection for MSDC via an external request with the message commands of MADOCA II. To perform the data collection with multiple signals requires time if we send messages for each piece of signal data. To perform fast data collection for multiple signals, we attached and packed data for multiple signals in a message which is then used for the data collection.

In the case of manual COD data collection, signal data was prepared in a GUI and we wanted to reflect the signal data in the data logging. To cope with such demands, we developed MSDC as shown in Figure 8. In this case, MSDC performs data collection by receiving messaging requests from remote client applications. Here, signal data is attached to the requested message.

- Various data formats in data collections

To treat data collections in various data formats such as waveform data in the same framework, we utilized MessagePack to serialize the data and store the serialized data as is in a NoSQL database. Since MessagePack has a self-described data format, we can treat the data in various formats with a unified method. For each data collection, we defined the data format used with MessagePack and applied it to the data collection.

- Support of a multi-platform environment

At SPring-8, we have required data collections to be supported on multiple platforms such as Solaris x86, Linux, ARM/Linux and Windows. In the MADOCA II data collection, the applications do not rely on the database library directly because the data collected is sent to a messaging-based streamer based on ZeroMQ and MessagePack. We also managed the settings of the data collection with a datacol file generated from the RDBMS. With these procedures, we were able to flexibly manage data collections under multiple platforms.

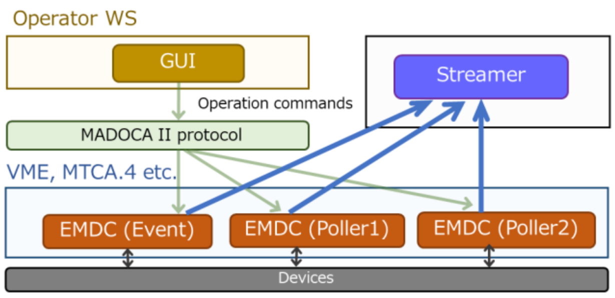


Figure 6: Configuration of the MADOCA II data collection processes with EMDC. EMDC can serve as an event triggered type (EMDC (Event)) or a periodic type (EMDC (Poller1, Poller2)). Here, EMDC (Poller1) and EMDC (Poller2)) have different data collection cycles.

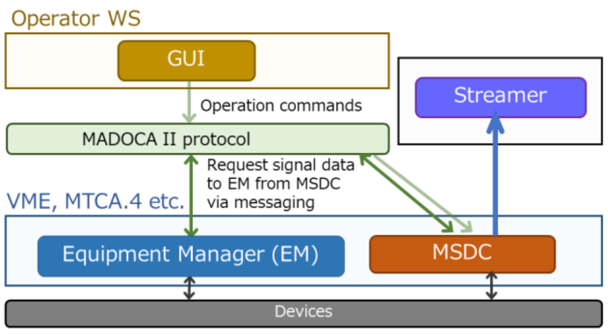


Figure 7: Configuration of MADOCA II data collection processes with MSDC. MSDC fetches signal data from an EM with the messaging commands of MADOCA II.

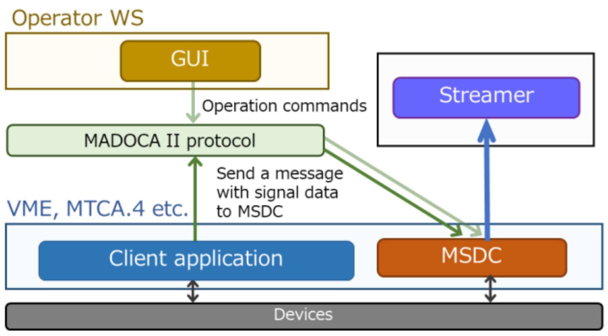


Figure 8: Configuration of MADOCA II data collection processes with MSDC. In this case, the client application prepares signals data and sends a message with signal data to MSDC.

IMPLEMENTATION OF MADOCA II DATA COLLECTION INTO SPring-8

The developed MADOCA II data collection framework described in the previous section has been partially implemented into the control system for the SPring-8 accelerator and beamline since April 2016. Currently MADOCA II data collection is applied for 11 equipment groups, 220 hosts and 277 data collection processes and used for Solaris x86, Linux, ARM/Linux, and Windows OS computers. For the signal registration procedure, we introduced test data collection which proved to be very useful in the preparation of the data collections. The web interface for editing SRT is still in the test phase, and an excel file was used instead.

Applications

We implemented MADOCA II data collections for applications with periodic and event triggered types.

Periodic data collection was applied to temperature and humidity measurements at a storage ring and status at a beamline PLC with Armadillo, for example. MADOCA II data collection was also applied to NewSUBARU in the same campus as SPring-8. Here, we applied data collection to monitor equipment with LabVIEW on the Windows OS. To achieve data collection with MADOCA II, we used the MADOCA II-LabVIEW interface to build an EM and MSDC shown in Figure 7. To perform 169 signals per second, multiple signal data was attached in one message for the first data collection.

In the case of data logging of DCCT integrated charge and DCCT lifetime, we needed to use DCCT current for the calculation of this data. In order to implement the data logging, we utilized MSDC as shown in Figure 8. Here, client applications fetch DCCT current data from the NoSQL database and calculated signal data is sent to MSDC via messaging by attaching the signal data.

Examples of the application of the MADOCA II data collection are also reported in other proceedings in this ICALEPCS [15].

Data Collection Operations Management

The management of operations for MADOCA II data collection is designed to have operational flexibility and easy to perform troubleshooting.

Figure 9 shows the schematic view of the management of the MADOCA II data collections. To manage the data collections operated in multiple hosts, we setup a data collection manager (DC manager) in an operator workstation. The DC manager was built with PyQt [16]. With the DC manager, we were able to perform administrative operations, such as the start and stop of the data collections in an equipment group unit, host and application. The DC manager also monitors the statuses of the data collection. These statuses are flexibly obtained from the message commands of MADOCA II and the parameters in the RDBMS. The statuses obtained can be tracked in the DC manager and also in the web portal in order to support external monitoring.

Figure 10 shows the DC manager managed by each equipment group. Here, we can watch the operation number of the data collection, as well as the host and failure counts for several cases, such as disconnection in the messaging, and stopped data collections. Since these counts in the failure are categorized for each case, it is useful for immediately understanding the cause of the issue. If counts with a red cell are displayed, the user can fix the issue by pushing a button on the right side. If counts with a brown cell are displayed, the user needs to ask the control experts to directly fix these issues. The detailed operations for each equipment group can be performed with a subwindow which appears when pushing a button in the purple cell on the left side. We can also operate a host and an application, and an example for the DC manager is shown in Figure 11.

Although the data collection Poller/Collector is still implemented in MADOCA, we updated the Poller/Collector to manage the operation with the message commands of MADOCA II and the management was also included in the DC manager. Thus, most of the data collections at SPring-8 could be flexibly managed with the DC manager and consequently, we were able to facilitate the operation and the troubleshooting of data collections at SPring-8.

Stability

After having started the operation of the MADOCA II data collection in April 2016, we have not encountered any issues with the operational stability of the data collection application itself. However, major issues occurred a few times with the data logging in Cassandra. In these situations, Cassandra nodes became unstable when the COD data was sent to the Cassandra nodes while its load was high due to the compaction process. The data size of the COD was 28 KB and collected with 10 Hz. Then, the data size became 24 GB per day. In the operation of Cassandra, we assigned a row key for signal data for one day and as a result, the managed data size exceeded the preferred data size for a row key of 10 to 20 MB. Therefore, we updated the row key to one per minute for COD data and reduced the collecting cycle to 1 Hz. After the fix, we encountered consistent stable operation in the MADOCA II data logging system.

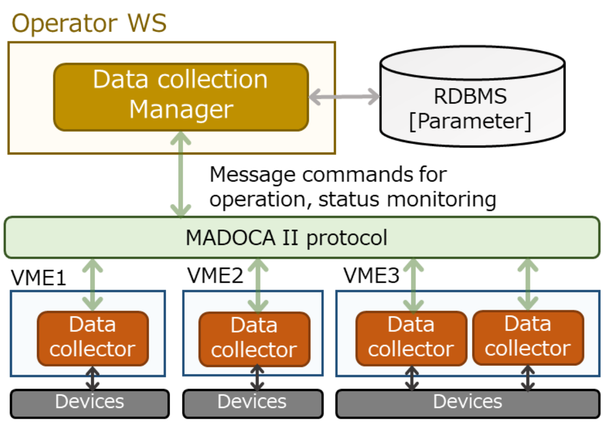
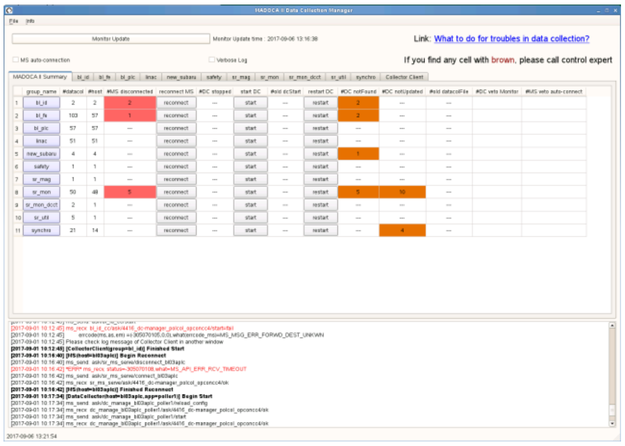


Figure 9: Management of the operation of the MADOCA II data collection.


MADOCA II Data Collection Manager

File Info

Monitor Update Monitor Update time : 2017-09-06 13:16:38 Link: What to do for troubles in data collection?

MS auto-connection Verbose Log If you find any cell with brown, please call control expert

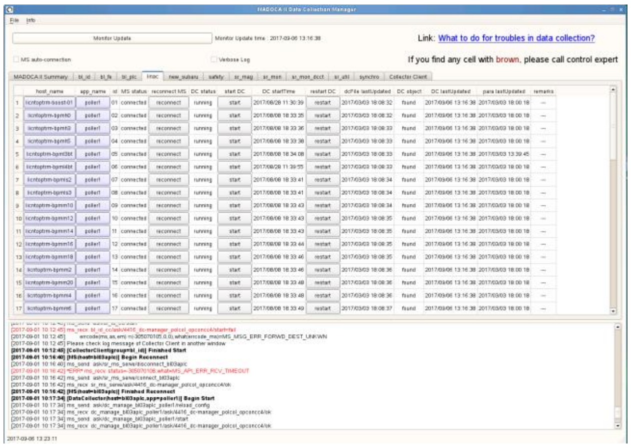
MADOCA II Summary | bl_id | bl_fe | bl_plc | linac | new_subaru | safety | sr_mag | sr_mon | sr_mon_dcct | sr_util | synchro | Collector Client

	group_name	#datacol	#host	#MS disconnected	reconnect MS	#DC stopped	start DC	#old dcStart	restart DC	#DC notFound	#DC notUpdated	#old datacolFile	#DC veto Monitor	#MS veto auto-connect
1	bl_id	2	2	2	reconnect	---	start	---	restart	2	---	---		
2	bl_fe	103	57	1	reconnect	---	start	---	restart	2	---	---		
3	bl_plc	57	57	---	reconnect	---	start	---	restart	---	---	---		
4	linac	51	51	---	reconnect	---	start	---	restart	---	---	---		
5	new_subaru	4	4	---	reconnect	---	start	---	restart	1	---	---		
6	safety	1	1	---	reconnect	---	start	---	restart	---	---	---		
7	sr_mag	1	1	---	reconnect	---	start	---	restart	---	---	---		
8	sr_mon	50	48	5	reconnect	---	start	---	restart	5	10	---		
9	sr_mon_dcct	2	1	---	reconnect	---	start	---	restart	---	---	---		
10	sr_util	5	1	---	reconnect	---	start	---	restart	---	---	---		
11	synchro	21	14	---	reconnect	---	start	---	restart	---	4	---		

```
[2017-09-01 10:12:45] ms_recv: bl_id_cc/ask/4416_dc-manager_polcol_opconcc4/start=fail
[2017-09-01 10:12:45]        errcode(ms,as,em) =(-305070105,0,0),what(errcode_ms)=MS_MSG_ERR_FORWD_DEST_UNKWN
[2017-09-01 10:12:45] Please check log message of Collector Client in another window
[2017-09-01 10:12:45] [CollectorClient(group=bl_id)] Finished Start
[2017-09-01 10:16:40] [MS(host=bl03aplc)] Begin Reconnect
[2017-09-01 10:16:40] ms_send: ask/sr_ms_serve/disconnect_bl03aplc
[2017-09-01 10:16:42] *ERR* ms_recv: status=-305070108,what=MS_API_ERR_RCV_TIMEOUT
[2017-09-01 10:16:42] ms_send: ask/sr_ms_serve/connect_bl03aplc
[2017-09-01 10:16:42] ms_recv: sr_ms_serve/ask/4416_dc-manager_polcol_opconcc4/ok
[2017-09-01 10:16:42] [MS(host=bl03aplc)] Finished Reconnect
[2017-09-01 10:17:34] [DataCollector(host=bl03aplc,app=poller1)] Begin Start
[2017-09-01 10:17:34] ms_send: ask/dc_manage_bl03aplc_poller1/reload_config
[2017-09-01 10:17:34] ms_recv: dc_manage_bl03aplc_poller1/ask/4416_dc-manager_polcol_opconcc4/ok
[2017-09-01 10:17:34] ms_send: ask/dc_manage_bl03aplc_poller1/start
[2017-09-01 10:17:34] ms_recv: dc_manage_bl03aplc_poller1/ask/4416_dc-manager_polcol_opconcc4/ok
```

2017-09-06 13:21:54

Figure 10: GUI for the MADOCA II data collection for the management of the operation of an equipment group unit.


MADOCA II Data Collection Manager

File Info

Monitor Update Monitor Update time : 2017-09-06 13:16:38 Link: What to do for troubles in data collection?

MS auto-connection Verbose Log If you find any cell with brown, please call control expert

MADOCA II Summary | bl_id | bl_fe | bl_plc | linac | new_subaru | safety | sr_mag | sr_mon | sr_mon_dcct | sr_util | synchro | Collector Client

	host_name	app_name	id	MS status	reconnect MS	DC status	start DC	DC startTime	restart DC	dcFile lastUpdated	DC object	DC lastUpdated	para lastUpdated	remarks
1	licntoptrm-boost01	poller1	01	connected	reconnect	running	start	2017/08/28 11:30:39	restart	2017/03/03 18:08:32	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
2	licntoptrm-bpmt0	poller1	02	connected	reconnect	running	start	2017/08/08 18:33:35	restart	2017/03/03 18:08:32	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
3	licntoptrm-bpmt3	poller1	03	connected	reconnect	running	start	2017/08/08 18:33:36	restart	2017/03/03 18:08:33	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
4	licntoptrm-bpmt5	poller1	04	connected	reconnect	running	start	2017/08/08 18:33:38	restart	2017/03/03 18:08:33	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
5	licntoptrm-bpmt3bt	poller1	05	connected	reconnect	running	start	2017/08/08 18:34:08	restart	2017/03/03 18:08:33	found	2017/03/06 13:16:38	2017/03/03 13:39:45	---
6	licntoptrm-bpmt4bt	poller1	06	connected	reconnect	running	start	2017/08/28 11:39:55	restart	2017/03/03 18:08:33	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
7	licntoptrm-bpms2	poller1	07	connected	reconnect	running	start	2017/08/08 18:33:41	restart	2017/03/03 18:08:34	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
8	licntoptrm-bpms3	poller1	08	connected	reconnect	running	start	2017/08/08 18:33:41	restart	2017/03/03 18:08:34	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
9	licntoptrm-bpmm10	poller1	09	connected	reconnect	running	start	2017/08/08 18:33:43	restart	2017/03/03 18:08:34	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
10	licntoptrm-bpmm12	poller1	10	connected	reconnect	running	start	2017/08/08 18:33:43	restart	2017/03/03 18:08:35	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
11	licntoptrm-bpmm14	poller1	11	connected	reconnect	running	start	2017/08/08 18:33:43	restart	2017/03/03 18:08:35	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
12	licntoptrm-bpmm16	poller1	12	connected	reconnect	running	start	2017/08/08 18:33:44	restart	2017/03/03 18:08:35	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
13	licntoptrm-bpmm18	poller1	13	connected	reconnect	running	start	2017/08/08 18:33:46	restart	2017/03/03 18:08:35	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
14	licntoptrm-bpmm2	poller1	14	connected	reconnect	running	start	2017/08/08 18:33:46	restart	2017/03/03 18:08:36	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
15	licntoptrm-bpmm20	poller1	15	connected	reconnect	running	start	2017/08/08 18:33:48	restart	2017/03/03 18:08:36	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
16	licntoptrm-bpmm4	poller1	16	connected	reconnect	running	start	2017/08/08 18:33:48	restart	2017/03/03 18:08:36	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---
17	licntoptrm-bpmm6	poller1	17	connected	reconnect	running	start	2017/08/08 18:33:49	restart	2017/03/03 18:08:37	found	2017/03/06 13:16:38	2017/03/03 18:00:18	---

```
[2017-09-01 10:12:45] ms_recv: bl_id_cc/ask/4416_dc-manager_polcol_opconcc4/start=fail
[2017-09-01 10:12:45]        errcode(ms,as,em) =(-305070105,0,0),what(errcode_ms)=MS_MSG_ERR_FORWD_DEST_UNKWN
[2017-09-01 10:12:45] Please check log message of Collector Client in another window
[2017-09-01 10:12:45] [CollectorClient(group=bl_id)] Finished Start
[2017-09-01 10:16:40] [MS(host=bl03aplc)] Begin Reconnect
[2017-09-01 10:16:40] ms_send: ask/sr_ms_serve/disconnect_bl03aplc
[2017-09-01 10:16:42] *ERR* ms_recv: status=-305070108,what=MS_API_ERR_RCV_TIMEOUT
[2017-09-01 10:16:42] ms_send: ask/sr_ms_serve/connect_bl03aplc
[2017-09-01 10:16:42] ms_recv: sr_ms_serve/ask/4416_dc-manager_polcol_opconcc4/ok
[2017-09-01 10:16:42] [MS(host=bl03aplc)] Finished Reconnect
[2017-09-01 10:17:34] [DataCollector(host=bl03aplc,app=poller1)] Begin Start
[2017-09-01 10:17:34] ms_send: ask/dc_manage_bl03aplc_poller1/reload_config
[2017-09-01 10:17:34] ms_recv: dc_manage_bl03aplc_poller1/ask/4416_dc-manager_polcol_opconcc4/ok
[2017-09-01 10:17:34] ms_send: ask/dc_manage_bl03aplc_poller1/start
[2017-09-01 10:17:34] ms_recv: dc_manage_bl03aplc_poller1/ask/4416_dc-manager_polcol_opconcc4/ok
```

2017-09-06 13:23:11

Figure 11: GUI for the MADOCA II data collection for the management of the operation with a host unit and the data collection process.

SUMMARY

In this paper, we reported on the developed MADOCA II data collection framework. The main purpose of MADOCA II data collection is to manage various data collections for SPring-8 within a unified framework. We achieved the goal by developing various data collection methods. All of the data collection methods (polling, event triggered type), various data formats, and multiple platforms can be flexibly managed within the same framework.

We also developed a web interface to facilitate the signal registration procedure, as well as a test system for the data collection. Furthermore, the management of the data collections was facilitated by leveraging the DC manager. With these developments, we obtained flexible management of the data collection and the maintenance costs have been reduced.

We implemented MADOCA II data collections into SPring-8 with 220 hosts, which has been operating with stability since April 2016.

ACKNOWLEDGEMENT

We would like to thank our colleagues in the Control Group in the Light Source Division, and the Information-technology Promotion Group in the Information-technology Promotion Division for useful discussions and suggestions on MADOCA II.

REFERENCES

[1] T. Matsumoto *et al.*, "Next-generation MADOCA for SPring-8 control framework", in *Proc. ICALEPCS'13*, San Francisco, USA, 2013, p.944.

[2] A. Yamashita *et al.*, "MADOCA II data logging system using NoSQL database for SPring-8", in *Proc. ICALEPCS'15*, Melbourne, Australia, 2015, p.648.

[3] National Instruments, `http://www.ni.com//labview`

[4] R. Tanaka *et al.*, "The first operation of control system at the SPring-8 storage ring", in *Proc. ICALEPCS'97*, Beijing, China, 1997, p.1.

[5] K. Okada *et al.*, "Upgrade of SACLA DAQ system adapts to multi-beamline operation", in *Proc. PCaPAC'14*, Karlsruhe, Germany, 2014, p. 22.

[6] T. Matsumoto *et al.*," Multi-host message routing in MADOCA II", in *Proc. ICALEPCS'15*, Melbourne, Australia, 2015, p. 954.

[7] ZeroMQ, `http://zeromq.org/`.

[8] MessagePack, `http://msgpack.org/`.

[9] Redis, `https://redis.io/`.

[10] Cassandra, `http://cassandra.apache.org/`.

[11] A. Taketani *et al.*, "Data Acquisition System with Database at the Storage Ring", in *Proc. ICALEPCS'97*, Beijing, China (1997), p.585

[12] T. Masuda *et al.*, "Event-synchronized data acquisition system for beam position monitors of SPring-8 linac", Nucl. Instrum. Methods A 545 (2005), p.415.

[13] Ruby On Rails, `http://rubyonrails.org/`

[14] T. Matsumoto *et al.*, "LabVIEW interface for MADOCA II with key-value stores in messages", in *Proc.* ICALEPCS'15, Melbourne, Australia, 2015, p.669.

[15] A. Kiyomichi *et al.*," MADOCA to EPICS gateway", ICALEPCS'17, Barcelona, Spain, this conference.

[16] River Bank Computing, `https://riverbankcomputing.com/software/pyqt/`.