FUJITSU

FUJITSU Software
Technical Computing Suite V4.0L20

Job Operation Software
API user's Guide
for Job Information Notification API

J2UL-2464-02ENZ0(01)
September 2020

Preface

Purpose of This Manual

This manual describes the job information notification API, which is a part of the job operation management function of the Job Operation Software included in Technical Computing Suite.

Intended Readers

This manual is intended for the administrator who operates and manages interconnects with the Job Operation Software.

The manual assumes that readers have the following knowledge:

- Basic Linux knowledge
- General knowledge of the Job Operation Software from the "Job Operation Software Overview"
- General knowledge of the job operation management function from the "Job Operation Software Administrator's Guide for Job Management"

Organization of This Manual

This manual is organized as follows.

Chapter 1 Overview of the Job Information Notification API

Describes an overview of the job information notification API and types of the job information notification API.

Chapter 2 Use of Job Information Notification API

Describes how to create a program that uses the job information notification API as well as a usage image of the API.

Appendix A Notes Relevant to Bulk Job Information Reported With the Job Information Notification API

Describes special notes on bulk job information reported with the job information notification API (how to select a report pattern of bulk job information and how to set a PJM code that is not subject to billing).

Appendix B Reference: APIs Relevant to Connection to Job Manager Function

Describes the functions used for connecting to or disconnecting from the job manager function.

Appendix C Reference: APIs Relevant to Setting of Monitoring Target Information

Describes the functions used to set a job to be monitored and set notifications (filter) of state transition information from the job manager function.

Appendix D Reference: APIs Relevant to Reading of Header Information

Describes the functions used for reading the header of notifications sent from the job manager function.

Appendix E Reference: APIs Relevant to Data Information

Describes the functions used for reading notification data sent from the job manager function.

Notation Used in This Manual

Notation of model names

In this manual, the computer that based on Fujitsu A64FX CPU is abbreviated as "FX server", and FUJITSU server PRIMERGY as "PRIMERGY server" (or simply "PRIMERGY").
Also, specifications of some of the functions described in the manual are different depending on the target model. In the description of such a function, the target model is represented by its abbreviation as follows:
[FX]: The description applies to FX servers.
[PG]: The description applies to PRIMERGY servers.

Administrators

The Job Operation Software has different types of administrators: system administrator, cluster administrator, and job operation administrator. However, they may all be represented as just "administrator" in this document. In such cases, an administrator who manages the system usually means the system administrator or cluster administrator. An administrator who manages job operations means the cluster administrator or job operation administrator.

Symbols in This Manual

This manual uses the following symbols.

Note

The Note symbol indicates an item requiring special care. Be sure to read these items.

See

The See symbol indicates the written reference source of detailed information.

Information

The Information symbol indicates a reference note related to Job Operation Software.

Export Controls

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Trademarks

- Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.
- All other trademarks are the property of their respective owners.

Date of Publication and Version

Version	Manual code
September 2020, Version 2.1	J2UL-2464-02ENZ0(01)
March 2020, Second version	J2UL-2464-02ENZ0(00)
January 2020, First version	J2UL-2464-01ENZ0(00)

Copyright

Update history

Changes	Location	Version
Fixed errata.	-	2.1
Added the value to be stored in the member pjsub_option_exflg of the job information notification structure Pjmapi_info_job_t.	E.1.1	2
Changed the look according to product upgrades.	-	
Fixed errata and modified descriptions.	-	

Contents

Chapter 1 Overview of the Job Information Notification API

The job manager function of the job operation management function provides an API (Application Program Interface) that notifies the programs of job-related information when a transition of job state occurs. Such programs are those that perform processes specific to job operation such as a billing processes and job tracing processes created by the job operation administrator. This API is called "job information notification API."

The job manager function provides the functions of the job information notification API as listed below. For details on using the functions, see "Chapter 2 Use of Job Information Notification API."

Table 1.1 Functions of job information notification API

Function	Description
PJM_connect()	Connecting to the job manager function
PJM_disconnect()	Disconnecting from the job manager function
PJM_set_target_jobinfo()	Setting monitoring target information (filter)
PJM_unset_target_jobinfo()	Cancelling monitoring target information (filter)
PJM_read_head()	Reading header information
PJM_read_data()	Reading data information

See

For details of each job information notification API, see from "Appendix B Reference: APIs Relevant to Connection to Job Manager Function" to "Appendix E Reference: APIs Relevant to Data Information."

Chapter 2 Use of Job Information Notification API

This chapter describes how to create a program that uses the job information notification API as well as how to use it.

2.1 How to Create a Program

The job information notification API is provided by the header file and library shown below. The header file and library are stored in the compute cluster management node.

- Header file

  /usr/include/FJSVtcs/pjm/pjmapi.h

- Library

  /usr/lib64/libpjmapi.so

The description below explains how to create a program.

Create a program in the compute cluster management node.

1. Include the header pjmapi.h in a source file that uses the job information notification API.

```
#include <FJSVtcs/pjm/pjmapi.h>

int main(void)
{
    // Processing using the job information notification API
    return 0;
}
```

2. Compile the created source file.
   When creating an executable file, link lpjmapi, the library of the job information notification API.

```
# gcc -lpjmapi -o module name source file
```

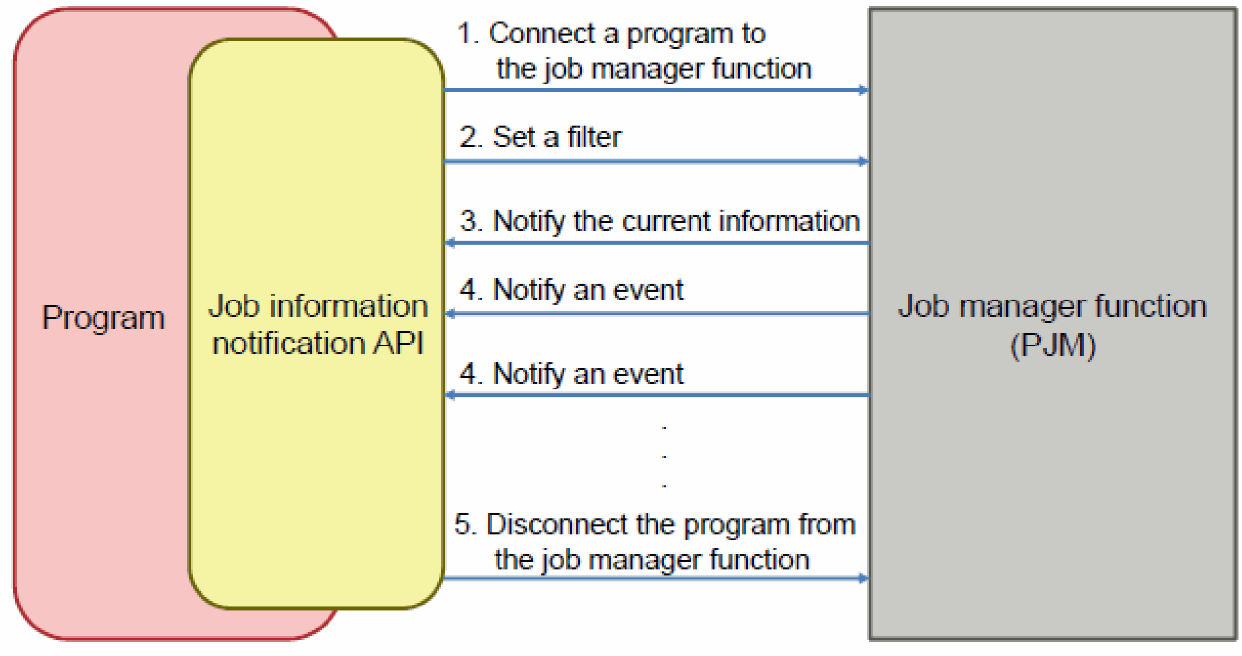
Note

As to a compiler, use the OS-standard gcc. All other compilers are not supported.

2.2 Processing Flow

The following is the processing flow when a program created by the job operation administrator uses the job information notification API.

Figure 2.1 Processing Flow when a Program Uses the Job Information Notification API


Program
Job information notification API
1. Connect a program to the job manager function
2. Set a filter
3. Notify the current information
4. Notify an event
4. Notify an event
5. Disconnect the program from the job manager function
Job manager function (PJM)

1. A program uses the function PJM_connect() to connect to the job manager function.
2. The program uses the function PJM_set_target_jobinfo() to set a filter for information that the program obtains.
3. The job manager function notifies of job information according to the filter, and the program obtains the information. After reading header information by using the function PJM_read_head(), the program uses the function PJM_read_data() to read job information.
4. The job manager function notifies of an event (differential information) according to the filter, and the program obtains the information. After reading header information by using the function PJM_read_head(), the program uses the function PJM_read_data() to read differential information that is contained in the header information and is specific to the event.
5. The program uses the function PJM_disconnect() to disconnect it from the job manager function.

2.3 Job-Related Information Obtained by a Program

The job-related information that a program obtains by using the job information notification API includes "job information" and "differential information."

Job information

Job-related information such as information of amounts of resources required by a job, limit values, scheduling result information, and statistical information. Job information is reported only on either of the following conditions.

- The job is filtered as a notification target ("3. Notification of current information" in Figure 2.1).
- The submitted job enters the QUEUED state for the first time ("4. Event notification" in Figure 2.1).

Differential information

Job information that is updated on job state transition. Differential information specific to each event is reported when a job state transition occurs ("4. Event notification" in Figure 2.1).

The order in which job information and differential information are reported is, for example, as follows for the period from job submission to job end.

1. Notification of change to the ACCEPT state
2. Notification of job information
3. Notification of change to the QUEUED state
4. Notification of scheduling result

5. Notification of change to the QUEUED state
6. Notification of change to the RUNNING-A state
7. Notification of change to the RUNNING state
8. Notification of change to the RUNOUT state
9. Notification of change to the EXIT state

Note

Notification of the transition to the ACCEPT state occurs prior to notification of job information.

See

When data of a notification sent by the function PJM_read_data() is read, in the case of job information, the data is stored in the job information notification structure PJM_INFO_JOB. In the case of differential information, the data is stored in the job state change notification structure PJM_CHANGE_*state name after transition* of the notification type corresponding to the state transition of the job. For details of each structure, see "Appendix E Reference: APIs Relevant to Data Information."

2.4 Example of the Job Information Notification API

This section provides an example of using the job information notification API.

The following is a program example.

- Main processing

```
main () {
     PJM_connect()                          /* 1. Connect a program to the job manager function */
     PJM_set_target_jobinfo()               /* 2. Set a filter */
     pthread_create()                       /* Creating a thread for accepting an event *./
     while(1){
                   /* Setting change acceptance process */.
               PJM_set_target_jobinfo()     /* 2. Set a filter */
               PJM_unset_target_jobinfo()   /* 2. Remove a filter */
     }
}
```

- Thread processing

```
thread_main() {
     while(1){
               PJM_read_head()              /* 3. Reading the header (waiting to be read) */
               PJM_read_data()              /* 3. Reading the data */
               switch(){                    /* Processing according to the read event */
                                            /* Describing processing details */
               }
     }
}
```

When the function PJM_read_head(), which reads the header of a notification sent from the job manager function, is called, the function waits for reading until the header to be reported is read. Therefore, if main processing such as accepting of change is performed even during the reading, use the function PJM_connect() to connect to the job manager function first, and use the function PJM_set_target_jobinfo() to set a filter, like the above program. Then, create a thread for reading for processing.

Appendix A Notes Relevant to Bulk Job Information Reported With the Job Information Notification API

This appendix describes special notes relevant to bulk job information reported with the job information notification API.

A.1 Bulk Job Information Reported

For a bulk job, you can use the function PJM_set_target_jobinfo(), which is used for setting filters, to select a notification pattern of bulk job information.

See

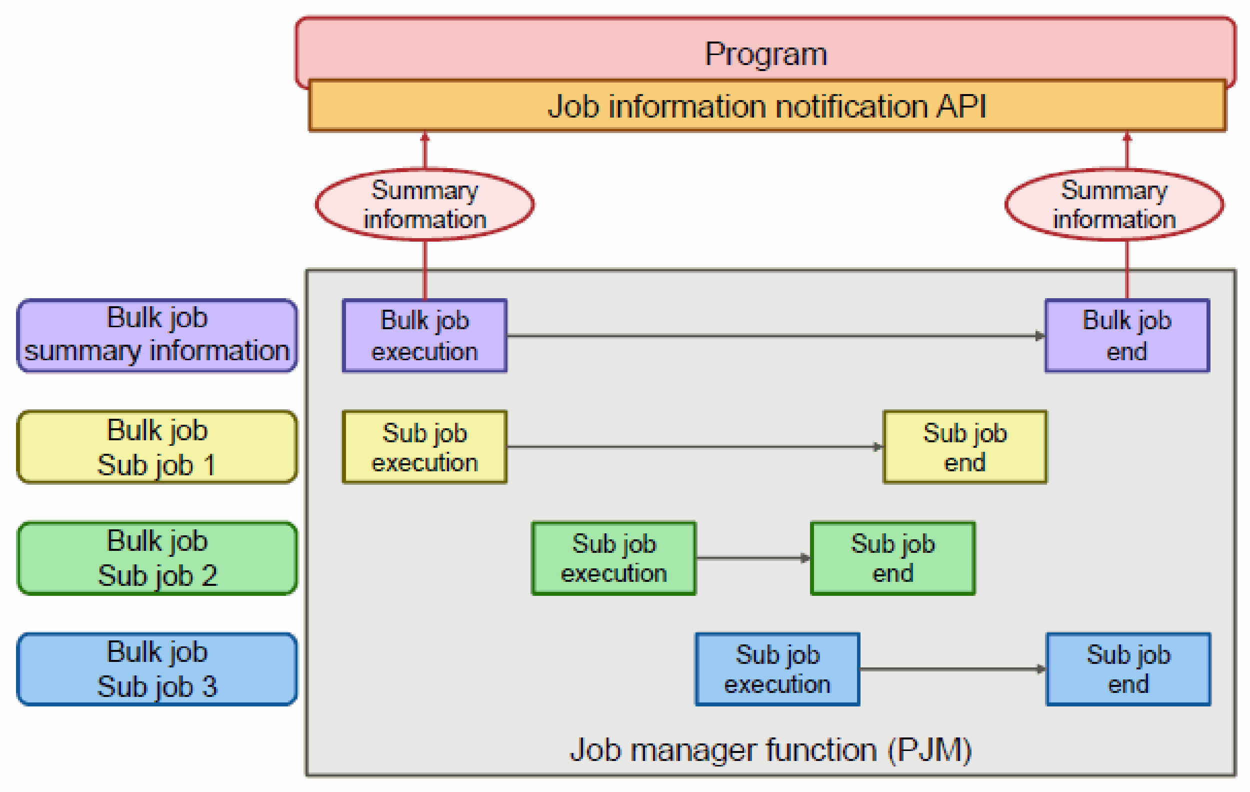
For details of the function PJM_set_target_jobinfo(), see "C.1 PJM_set_target_jobinfo()."

The following explains how to select a notification pattern of bulk job information.

- For notification of only summary information of a bulk job
  Specify "0" in the bulk_subjob_notify_trigger member of the target_info structure to be handed over to the function PJM_set_target_jobinfo().

  The figure below indicates the notification timings of summary information from the job manager function when a bulk job is executed. Summary information is reported at the timing when the first sub job is started, and summary information of the bulk end is reported at the timing when the last sub job is ended.

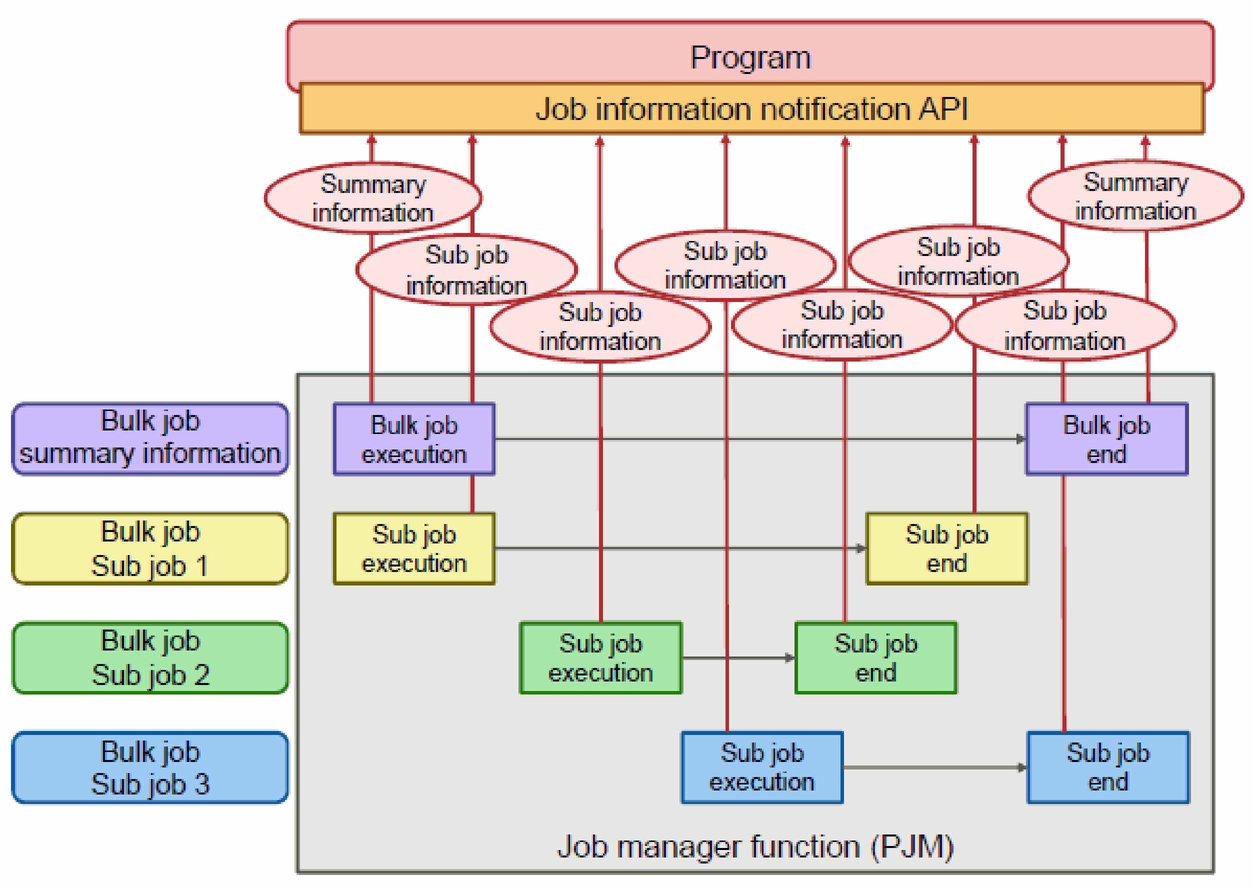
Figure A.1 Notification of Only Summary Information of a Bulk Job



- For notification triggered by every sub job besides notification of summary information of a bulk job
  Set the flag "PJM_STATUS_*state*" corresponding to the state for which notification is necessary in the bulk_subjob_notify_trigger member of the target_info structure to be handed over to the function PJM_set_target_jobinfo().

  The figure below indicates that summary information and sub job information are reported at the start and end of every sub job.

  Figure A.2 Notification Triggered by Every Sub Job Besides Notification of Summary Information of a Bulk Job



A.2 How to Set the End Code of a Job That is Not Subject to Billing

The job execution elapse time (elapse) of the summary information of a bulk job is the total value of the execution elapse times of all the sub jobs. However, this value includes the execution elapse times of the sub jobs that are not subject to billing. If some sub jobs are not executed correctly due to a node down or another cause (in the case of the job end code (PJM code) of a value other than 0), these sub jobs may be example cases of such sub jobs. In preparation of such a case, to exclude the elapse values of the sub jobs that do not end correctly from the billing targets, specific PIM codes can be set. Through this setting, the execution elapse times of the sub jobs that end with the specific PJM codes can be summarized as the job execution elapse time that is not subject to billing (elapse_off_acc).

To specify the PJM codes relevant to which execution elapse times are not subject to billing, add the setting below to papjmapi.conf, a configuration file relevant to the job information notification API. This file is provided in /var/opt/FJSVtcs/shared_disk/pjm/.private/ in the compute cluster management node.

```
ELAPSE_OFF_ACC= PJM code
```

[Notes on setting]

- Describe PJM codes in numerical characters. Description using other characters is invalid.
- To describe multiple PJM codes, delimit them with a comma "," or a space character.
- PJM codes are not checked. Therefore, if you describe an invalid PJM code, it is set.
- The lines in which ELAPSE_OFF_ACC= is missing at the beginning are skipped.

- If you describe multiple lines, they are all valid.
- If a line includes an invalid character string, the line is valid up to that character.
- The maximum length of a line is 4096 characters. A line that is 4097 characters long or longer causes a read error.
- The number of PJM codes that can be set is 1024. If 1025 or more PJM codes are set, the first 1024 codes are set and the rest are ignored.

The following is an example.

```
# cat /var/opt/FJSVtcs/shared_disk/pjm/.private/papjmapi.conf
ELAPSE_OFF_ACC= 1,2,3
ELAPSE_OFF_ACC= 4 5 6
ELAPSE_OFF_ACC= 5 6 7
```

In this example, for sub jobs with PJM codes 1 to 7, the elapse values are added to elapse_off_acc since they are treated as sub jobs that are not subject to billing.

Appendix B Reference: APIs Relevant to Connection to Job Manager Function

This appendix describes the APIs relevant to connection to the job manager function.

Information

The description from "Appendix B: Reference: APIs Relevant to Connection to Job Manager Function" to "Appendix E Reference: APIs Relevant to Data Information" shows input-output of the arguments of functions (in and out). The meanings of in and out are as follows.

- in
  A program that uses the job information notification API sets this information, and the job manager function refers to this information.
- out
  The job manager function sets this information, and a program that uses the job information notification API refers to this information.

B.1 PJM_connect()

This is a function used for connecting to the job manager function.

```
#include <FJSVtcs/pjm/pjmapi.h>
int PJM_connect(int *errcd);
```

Table B.1 Argument of PJM_connect()

Argument	Type	Input-Output	Description
errcd	int *	out	If connection to the job manager function fails, the value corresponding to the error is set.

Table B.2 Return Values of PJM_connect()

Result	Return Value
In the case of success	The connected file descriptor is returned.
In the case of an error	-1 is returned, and any one of the following values corresponding to errors is set in errcd.

Table B.3 Values Set in errcd in the Case of Errors

Value Set in errcd	Meaning
PJM_ERR_BOOTYET	The job manager function has not been started.
PJM_ERR_CANTOPEN	Opening of the file descriptor failed.
PJM_ERR_INTERNAL	An internal error occurred.

B.2 PJM_disconnect()

This is a function used for disconnecting from the job manager function.

```
#include <FJSVtcs/pjm/pjmapi.h>
int PJM_disconnect(int fd, int *errcd);
```

Table B.4 Arguments of PJM_disconnect()

Argument	Type	Input-Output	Description
fd	int	in	Set the file descriptor returned by PJM_connect.
errcd	int *	out	If disconnection fails, the value corresponding to the error is set.

Table B.5 Return Values of PJM_disconnect()

Result	Return Value
In the case of success	0 is returned.
In the case of an error	-1 is returned, and any one of the following values corresponding to errors is set in errcd.

Table B.6 Values Set in errcd in the Case of Errors

Value Set in errcd	Meaning
PJM_ERR_CANTCLOSE	A close error occurred.
PJM_ERR_INTERNAL	An internal error occurred.

Appendix C Reference: APIs Relevant to Setting of Monitoring Target Information

This appendix describes APIs relevant to setting of monitoring target information.

C.1 PJM_set_target_jobinfo()

This function is used to set a job to be monitored and to make the setting of notification of state transition information from the job manager function (filter). It reports job information or differential information of the job specified by target_job as a monitoring target at the timing specified by notify_trigger.

```
#include <FJSVtcs/pjm/pjmapi.h>
int PJM_set_target_jobinfo(int fd, const struct target_info *targetinfo);
```

Table C.1 Arguments of PJM_set_target_jobinfo()

Argument	Type	Input-Output	Description
fd	int	in	The file descriptor returned from PJM_connect is set.
targetinfo	const struct target_info *	out	The relevant information can be added as a monitoring target by setting a condition in each member variable.

The structure target_info is as follows.

```
typedef struct target_info {
    struct target_job targetjob;               /* (in)target_job */
    int     kind_info                          /* (in)kind of targetjob */
    int     notify_trigger;                    /* (in)job info notify trigger */
    int     bulk_subjob_notify_trigger ;       /* (in)bulk subjob notify trigger */
    int     simple_data_model;                 /* (in)simple data of model */
} target_info_t;

typedef struct target_job {
    int     kind_job                           /* (in)kind of information */
    int     job_id;                            /* (in)job ID */
    uid_t   uid;                               /* (in)user id */
    gid_t   gid;                               /* (in)user group id*/
    char    rscunit_name[PJM_RSCUNAME_MAX];    /* (in)rsc unit name */
    char    rscgrp_name[PJM_RSCGROUP_MAX];     /* (in)rsc group name*/
} target_job_t;
```

Each member has the following values.

Table C.2 Members of the target_info Structure

Member	Type	Input-Output	Description
target_job	struct target_job	in	The relevant jobs can be added as monitoring targets by setting a condition in each member. If kind_job is PJM_KINDJOB_ALL and the PJM_KINDINFO_JOB flag is set in kind_info, all the jobs are targets.
kind_info	int	in	Specifies what type of information is obtained. The following value is specified. PJM_KINDINFO_JOB: Job information
notify_trigger	int	in	Specifies what conditions are notified. For reported information, see "E.1 PJM_read_data()." Specification is valid only when the PJM_KINDINFO_JOB flag is set

Member	Type	Input-Output	Description
			in kind_info. Logical addition of the following values are specified. PJM_STATUS_INFO_JOB: Job information is reported on either of the following conditions. - The filter setting specifies the job as a notification target. - The submitted job enters the QUEUED state for the first time. PJM_STATUS_ACCEPT: ACCEPT state PJM_STATUS_QUEUED: QUEUED state PJM_STATUS_RUNNING_A: RUNNING-A state PJM_STATUS_RUNNING: RUNNING state PJM_STATUS_RUNOUT: RUNOUT state PJM_STATUS_EXIT: EXIT state PJM_STATUS_CANCEL: CANCEL state PJM_STATUS_HOLD: HOLD state PJM_STATUS_ERROR: ERROR state PJM_STATUS_SCHED: Scheduling notification PJM_STATUS_REJECT: REJECT state PJM_STATUS_RUNNING_P: RUNNING-P state PJM_STATUS_RUNNING_E: RUNNING-E state PJM_STATUS_ALTER: Job attribute change notification PJM_STATUS_ALL: All the states
bulk_subjob_notify_trigger	int	in	Specifies in what conditions a sub job of a bulk job is notified. For reported information, see "E.1 PJM_read_data()." Specification is valid only when the PJM_KINDINFO_JOB flag is set in kind_info. Logical addition of the following values are specified. The setting of a status for which there is no notification event of a sub job (PJM_STATUS_REJECT, PJM_STATUS_ACCEPT, etc.) is ignored. PJM_STATUS_INFO_JOB: Job information is reported on either of the following conditions. - The filter setting specifies the job as a notification target. - The submitted job enters the QUEUED state for the first time. PJM_STATUS_ACCEPT: ACCEPT state PJM_STATUS_QUEUED: QUEUED state PJM_STATUS_RUNNING_A: RUNNING-A state PJM_STATUS_RUNNING: RUNNING state PJM_STATUS_RUNOUT: RUNOUT state PJM_STATUS_EXIT: EXIT state PJM_STATUS_CANCEL: CANCEL state PJM_STATUS_HOLD: HOLD state PJM_STATUS_ERROR: ERROR state PJM_STATUS_SCHED: Scheduling notification PJM_STATUS_REJECT: REJECT state PJM_STATUS_RUNNING_P: RUNNING-P state PJM_STATUS_RUNNING_E: RUNNING-E state PJM_STATUS_ALTER: Job attribute change notification PJM_STATUS_ALL: All the states (All bits are set)
simple_data_model	int	in	Specifies whether to report a normal job, a step job, summary information of a bulk job, or sub job notification of a bulk job as simple data.

Member	Type	Input-Output	Description
			Logical addition of the following values are specified. For a job model for which any of the following values are not set, data corresponding to the conventional notification type is reported. PJM_SIMPLE_DATA_NORMAL: Normal job notification is reported as simple data. PJM_SIMPLE_DATA_STEP: Step job notification is reported as simple data. PJM_SIMPLE_DATA_BULK_SUMMERY: The summary information of a bulk job is reported as simple data. PJM_SIMPLE_DATA_BULK_SUBJOB: Sub job notification of a bulk job is reported as simple data. PJM_SIMPLE_DATA_ALL: All notification is reported as simple data (all the bits are set).
kind_job	int	in	Specifies jobs of notification targets. Logical addition of the following values are specified. If multiple bits are set, jobs that satisfy all the conditions are targets. PJM_KINDJOB_ALL: All the jobs PJM_KINDJOB_ID: Specification by a job ID PJM_KINDJOB_USER: Specification by a user ID PJM_KINDJOB_GROUP: Specification by a group ID PJM_KINDJOB_RSCUNIT: Specification by a resource unit ID PJM_KINDJOB_RSCGRP: Specification by a resource group ID
job_id	uint	in	Specifies the job ID of a target job. Specification is valid only when PJM_KINDJOB_ID is spedified for kind_job.
uid	uid_t	in	Specifies the owner user ID of a target job. Specification is valid only when PJM_KINDJOB_USER is spedified for kind_job.
gid	gid_t	in	Specifies the owner group ID of a target job. Specification is valid only when PJM_KINDJOB_GROUP is spedified for kind_job.
rscunit_name [PJM_RSCUNAME_MAX]	char[]	in	Specifies the owner resource unit ID of a target job. PJM_RSCUNAME_MAX is 64. Specification is valid only when PJM_KINDJOB_RSCUNIT is spedified for kind_job.
rscgrp_name [PJM_RSCGROUP_MAX]	char[]	in	Specifies the owner resource group ID of a target job. PJM_RSCUNAME_MAX is 64. Specification is valid only when PJM_KINDJOB_RSCGRP is spedified for kind_job.

Table C.3 Return Values of PJM_set_target_jobinfo()

Result	Return Value
In the case of success	0 is returned.
In the case of an error	The value corresponding to any one of the following errors is returned. PJM_ERR_SEND: A send error occurred. PJM_ERR_INTERNAL: An internal error occurred.

Note

If PJM_set_target_job() is called multiple times with different arguments, new information is added while the previously registered job information transition notification is maintained as it is.

C.2 PJM_unset_target_jobinfo()

This function is used to deselect jobs from the monitoring targets. This function is also used to deselect the state transition notification information (filter) of the job manager function from the monitoring targets.
The specification method of jobs is the same as that for PJM_set_target_jobinfo(). For details, see "C.1 PJM_set_target_jobinfo()."

```
#include <FJSVtcs/pjm/pjmapi.h>
int PJM_unset_target_jobinfo(int fd, const struct target_info *targetinfo);
```

Table C.4 Argument of PJM_unset_target_jobinfo()

Argument	Type	Input-Output	Description
fd	int	in	The file descriptor returned from PJM_connect is set.
targetinfo	const struct target_info *	in	The same as struct target_info **targetinfo* of PJM_set_target_jobinfo() For details, see "Table C.1 Arguments of PJM_set_target_jobinfo()."

Table C.5 Return Values of PJM_unset_target_jobinfo()

Result	Return Value
In the case of success	0 is returned.
In the case of an error	The value corresponding to any one of the following errors is returned. PJM_ERR_SEND: A send error occurred. PJM_ERR_INTERNAL: An internal error occurred.

Appendix D Reference: APIs Relevant to Reading of Header Information

This appendix describes the APIs relevant to reading of header information.

D.1 PJM_read_head()

This function is used to read the header of a notification sent from the job manager function. A program using this API waits for read data until the header to be reported is read. By calling PJM_read_data() according to the reported event type, the program can obtain the contents of the notification.

```
#include <FJSVtcs/pjm/pjmapi.h>
int PJM_read_head(int fd, int *notice_kind, int *data_flags, struct timespec *sendtime, int
*datasize);
```

Table D.1 Arguments of PJM_read_head()

Argument	Type	Input-Output	Description
fd	int	in	The file descriptor returned from PJM_connect is set.
notice_kind	int *	out	The following notification types reported by the job manager function are stored. PJM_INFO_JOB: Job information notification PJM_CHANGE_ACCEPT: Notification of change to the ACCEPT state PJM_CHANGE_QUEUED: Notification of change to the QUEUED state PJM_CHANGE_RUNNING_A: Notification of change to the RUNNING-A state PJM_CHANGE_RUNNING: Notification of change to the RUNNING state PJM_CHANGE_RUNOUT: Notification of change to the RUNOUT state PJM_CHANGE_EXIT: Notification of change to the EXIT state PJM_CHANGE_CANCEL: Notification of change to the CANCEL state PJM_CHANGE_HOLD: Notification of change to the HOLD state PJM_CHANGE_ERROR: Notification of change to the ERROR state PJM_CHANGE_SCHED: Notification of scheduling result PJM_CHANGE_REJECT: Notification of change to the REJECT state PJM_CHANGE_RUNNING_P: Notification of change to the RUNNING-P state PJM_CHANGE_RUNNING_E: Notification of change to the RUNNING-E state PJM_CHANGE_ALTER: Notification of attribute change
data_flags	int *	out	The data type reported by the job manager function is stored. For a data type, information is represented as a bit value, and the logical addition of the values below is stored. (*) Currently, only PJM_DATA_FLAGS_SIMPLE is supported. If nothing is set (0 is set), the data structure of notification contents obtained with PJM_read_data() is a structure corresponding to the notification type specified by conventional notice_kind. PJM_DATA_FLAGS_SIMPLE: Notification of simple data The data structure of notification contents obtained with PJM_read_data() is not a structure corresponding to the notification type specified by notice_kind but the simple data notification structure.
sendtime	struct timespec *	out	The time when the job manager function sends a notification is stored.

Argument	Type	Input-Output	Description
datasize	int *	out	The size of data to be read with PJM_read_data() is stored.

Table D.2 Return Values of PJM_read_head()

Result	Return Value
In the case of success	0 is returned.
In the case of an error	The value corresponding to any one of the following errors is returned. PJM_ERR_INVALID: PJM_read_head() was called without reading data. PJM_ERR_READ: A read error occurred. PJM_ERR_INTERNAL: An internal error occurred.

Appendix E Reference: APIs Relevant to Data Information

This appendix describes APIs relevant to data information.

E.1 PJM_read_data()

This function reads notification data sent from the job manager function (data corresponding to the header read with PJM_read_head()).

```
#include <FJSVtcs/pjm/pjmapi.h>
int PJM_read_data(int fd, void **data_p);
```

Table E.1 Arguments of PJM_read_data()

Argument	Type	Input-Output	Description
fd	int	in	The file descriptor returned from PJM_connect is set.
data_p	void **	out	Storage pointer of obtained data The caller secures a memory space of datasize read with PJM_read_head(), and specifies the pointer to this space. Stored data is a structure corresponding to the notification type and data type. The following are the notification types. PJM_INFO_JOB: Job information notification structure PJM_CHANGE_ACCEPT: Notification structure of change to the ACCEPT state PJM_CHANGE_QUEUED: Notification structure of change to the QUEUED state PJM_CHANGE_RUNNING_A: Notification structure of change to the RUNNING-A state PJM_CHANGE_RUNNING: Notification structure of change to the RUNNING state PJM_CHANGE_RUNOUT: Notification structure of change to the RUNOUT state PJM_CHANGE_EXIT: Notification structure of change to the EXIT state PJM_CHANGE_CANCEL: Notification structure of change to the CANCEL state PJM_CHANGE_HOLD: Notification structure of change to the HOLD state PJM_CHANGE_ERROR: Notification structure of change to the ERROR state PJM_CHANGE_SCHED: Notification structure of scheduling result PJM_CHANGE_REJECT: Notification structure of change to the REJECT state PJM_CHANGE_RUNNING_P: Notification structure of change to the RUNNING-P state PJM_CHANGE_RUNNING_E: Notification structure of change to the RUNNING-E state PJM_CHANGE_ALTER: Notification structure of attribute change The following is the data type. PJM_DATA_FLAGS_SIMPLE: Simple data notification structure

Table E.2 Return Values of PJM_read_data()

Result	Return Value
In the case of success	0 is returned.
In the case of an error	The value corresponding to any one of the following errors is returned. PJM_ERR_READ: A read error occurred. PJM_ERR_INTERNAL: An internal error occurred.

E.1.1 Job Information Notification Structure (PJM_INFO_JOB)

The function PJM_read_data() reads the notification data sent when the notification type is PJM_INFO_JOB. The following are the job information notification structures that are reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_info_job {
   uint16_t         job_model;                      /* Job model */
   uint16_t         job_flags;                      /* Job additional information */
   uint16_t         num_retry;                      /* Retry count */
   int16_t          pre_jobstatus;                  /* Previous job status */
   int16_t          jobstatus;                      /* Job status */
   int16_t          job_aprio;                      /* Job priority level within the resource unit */
   int16_t          job_uprio;                      /* Job priority level within the same user */
   int16_t          pad1;
   uint32_t         job_type;                       /* Job type */
   uint32_t         jobid;                          /* Job ID */
   uint32_t         blkno;                          /* Bulk number */
   uint32_t         stepno;                         /* Step number */
   int              subjob_num;                     /* Number of sub jobs */
   uint             exit_code;                      /* exitcode of the user script */
   int32_t          signal_no;                      /* Signal number of the user script */
   int              pjm_code;                       /* PJM code */
   uint             node_x;                         /* Allocated node shape x */
   uint             node_y;                         /* Allocated node shape y */
   uint             node_z;                         /* Allocated node shape z */
   uint             node_num;                       /* Number of allocated nodes */
   uid_t            uid;                            /* User ID */
   gid_t            exec_gid;                       /* Execution group ID */
   uint             node_req_x;                     /* Number of requested nodes in
                                                       the x direction */
   uint             node_req_y;                     /* Number of requested nodes in
                                                       the y direction */
   uint             node_req_z;                     /* Number of requested nodes in
                                                       the z direction */
   uint             node_req_num;                   /* Number of requested nodes */
   uint             node_mpi_x;                     /* --Node shape with mpi option
                                                       specification (x) */
   uint             node_mpi_y;                     /* --Node shape with mpi option
                                                       specification (y) */
   uint             node_mpi_z;                     /* --Node shape with mpi option
                                                       specification (z) */
   uint             node_mpi_num;                   /* --Number of nodes with mpi option
                                                       specification */
   int              proc;                           /* Number of processes */
   int              proc_bynode;                    /* Number of processes for 1 node */
   int              sd_num;                         /* Number of dependence relation expressions */
   uid_t            lasthold_uid;                   /* User ID held/cancelled in the last state */
   uint32_t         subjobflag;                     /* Attribute of transfer wait for preceding
                                                       step job result */
   uint             hold_count;                     /* HOLD count */
   int              run_count;                      /* RERUN count */
   uint             unavailable_nodenum;            /* Number of unavailable nodes */
   uint             sum_cpu_req_num;                /* Total number of requested CPUs */
   uint32_t         used_cpunum;                    /* Number of CPUs used */
   int              umask;                          /* File mask */
   int              mailflag;                       /* Mail send flag */
   uint             pro_exit_code;                  /* Prologue exit code */
   uint             epi_exit_code;                  /* Epilogue exit code */
   uint32_t         vn_cpu_req;                     /* Requested number of CPU cores
                                                       by virtual node */
   int              rankmap_type;                   /* Rank map type */
   int              rankmap_num;                    /* Number of placed rank maps */
   int              vn_multi;                       /* Number of placed virtual nodes */
   int              node_type;                      /* Node type */
   uint32_t         num_alloc_vnode;                /* Number of allocated virtual nodes */
   uint             sum_cpu_alloc_num;              /* Total number of allocated CPUs */
   uint             used_nodenum;                   /* Number of nodes used */
   uint             sum_cpu_prealloc_num;           /* Total number of scheduler allocation CPUs */
```

```
    time_t           create_date;                    /* Job submission time */
    time_t           start_date;                     /* Job start time */
    time_t           end_date;                       /* Job end time */
    time_t           elapse;                         /* Job execution elapse time */
    time_t           elapse_off_acc;                 /* Job execution elapse time that is not subject
                                                        to billing */
    time_t           sched_date;                     /* Job execution start time */
    time_t           que_date;                       /* Time of QUEUED transition */
    time_t           runa_date;                      /* Time of RUNNING-A transition */
    time_t           run_date;                       /* Time of RUNNING transition */
    time_t           runout_date;                    /* Time of RUNOUT transition */
    time_t           exit_date;                      /* Time of EXIT transition */
    time_t           cancel_date;                    /* Time of CANCEL transition */
    time_t           hold_date;                      /* Time of HOLD transition */
    time_t           err_date;                       /* Time of ERR transition */
    uint64_t         attribute;                      /* Job attribute */
    time_t           spec_date;                      /* Execution start time specified at
                                                        job submission */
    uint64_t         elapse_limit;                   /* Limit value of job elapse time */
    char             jobname[64];                    /* Job name */
    char             rscunit_name[PJM_RSCUNAME_MAX]; /* Resource unit name */
    char             rscgrp_name[PJM_RSCGROUP_MAX];  /* Resource group name */
    uint64_t         mem_req;                        /* Requested memory amount (by node) */
    uint64_t         node_cpu;                       /* Limit value of CPU time (by node) */
    time_t           reject_date;                    /* Reject transition time */
    char             hostname[16];                   /* Submitted host name */
    char             reason[64];                     /* REASON */
    time_t           fst_start_date;                 /* Initial job execution start time */
    uint64_t         sum_runa_time;                  /* Cumulative RUNNING-A time (seconds) */
    uint64_t         sum_run_time;                   /* Cumulative RUNNING time (seconds) */
    uint64_t         sum_hold_time;                  /* Cumulative HOLD time (seconds) */
    uint64_t         sum_wait_time;                  /* Cumulative wait time (seconds) */
    uint64_t         mem_lmt;                        /* Memory amount limit value */
    uint64_t         mem_job_alloc;                  /* Allocated memory amount */
    uint64_t         prc_cputm_lmt;                  /* CPU time limit value by process */
    uint64_t         prc_corefile_lmt;               /* Core file size limit value by process */
    uint64_t         prc_cre_proc_lmt;               /* Limit value of the number of user processes
                                                        by process */
    uint64_t         prc_data_lmt;                   /* Data segment limit value by process */
    uint64_t         prc_locked_mem_lmt;             /* Lock memory limit value by process */
    uint64_t         prc_psx_msq_que_lmt;            /* POSIX message queue limit value by process */
    uint64_t         prc_openfiles_lmt;              /* File descriptor limit value by process */
    uint64_t         prc_pndng_sgnl_lmt;             /* Limit value of the number of signals
                                                        by process */
    uint64_t         prc_prmfl_lmt;                  /* File size limit value by process */
    uint64_t         prc_stack_lmt;                  /* Stack segment limit value by process */
    uint64_t         prc_vmem_lmt;                   /* Virtual memory size limit value by process */
    uint64_t         max_used_mem;                   /* Maximum memory use amount (bytes) */
    uint64_t         usctmut;                        /* Total user CPU time and total system CPU time
                                                        (seconds) */
    time_t           snapshottime;                   /* Data collection year/month/day */
    time_t           pjdel_date;                     /* Job deletion request time */
    time_t           delete_date;                    /* Job deletion time */
    time_t           all_prec_subjob_exit_date;      /* Preceding sub job end time */
    char             fsname[64];                     /* External file system name */
    char             appname[64];                    /* Application name */
    time_t           prologue_start_date;            /* Prologue start time */
    time_t           runp_date;                      /* Time of RUNNING-P transition */
    time_t           epilogue_start_date;            /* Epilogue start time */
    time_t           rune_date;                      /* Time of RUNNING-E transition */
    time_t           prologue_end_date;              /* Prologue end time */
    time_t           epilogue_end_date;              /* Epilogue end time */
    uint64_t         pack_policy;                    /* Virtual node placement policy */
```

```
uint64_t         exec_policy;                    /* Execution mode */
uint64_t         sum_vm_job_use;                 /* Total used memory amount by virtual node
                                                    (bytes) */
uint64_t         sum_usr_cputm;                  /* Total user CPU time */
uint64_t         sum_sys_cputm;                  /* Total system CPU time */
uint64_t         mem_job_prealloc;               /* Scheduler allocation memory amount (bytes) */
off_t            curdir_ofs;                     /* Offset to job submission directory */
off_t            mail_ofs;                       /* Offset to mail address */
off_t            shell_ofs;                      /* Offset to job shell */
off_t            comment_ofs;                    /* Offset to comment */
off_t            stdout_ofs;                     /* Offset to standard output file */
off_t            stderr_ofs;                     /* Offset to standard error output file */
off_t            job_acct_ofs;                   /* Offset to job statistical information
                                                    file path */
off_t            ndlist_ofs;                     /* Offset to node ID list */
off_t            tofulist_ofs;                   /* Offset to Tofu coordinate list */
off_t            sd_ofs;                         /* Offset to dependency expression
                                                    (Pjmapi_sd_t) */
int32_t          req_mpi_static_proc;            /* Number of requested MPI static processes */
int32_t          req_mpi_proc;                   /* Number of requested MPI processes */
int32_t          alloc_mpi_static_proc;          /* Number of allocated MPI static processes */
int32_t          alloc_mpi_proc;                 /* Number of allocated MPI processed */
uint64_t         numa_policy;                    /* NUMA policy */
uint32_t         start_blkno;                    /* Bulk start number */
uint32_t         end_blkno;                      /* Bulk end number */
uint32_t         affected_nid;                   /* Node ID that affected job result */
uint32_t         prealloc_rmexit_exitcode;       /* prealloc exit end code */
uint32_t         predel_rmexit_exitcode;         /* predel exit end code */
uint32_t         postfree_rmexit_exitcode;       /* postfree exit end code */
uint64_t         prealloc_start_time;            /* prealloc exit start time */
uint64_t         prealloc_end_time;              /* prealloc exit end time */
uint64_t         predel_start_time;              /* predel exit start time */
uint64_t         predel_end_time;                /* predel exit end time */
uint64_t         postfree_start_time;            /* postfree exit start time */
uint64_t         postfree_end_time;              /* postfree exit end time */
uint8_t          prealloc_exec_kind;             /* prealloc exit execution timing */
uint8_t          predel_exec_kind;               /* predel exit execution timing */
uint8_t          postfree_exec_kind;             /* postfree exit execution timing */
uint8_t          backfill_flg;                   /* Backfill flag */
uint8_t          pad4[4];
struct timespec  last_sched_date;                /* Scheduling start time */
uint64_t         pjsub_option_flg;               /* pjsub command option */
uint64_t         pjsub_option_exflg;             /* Enhanced pjsub command option */
uint64_t         pjsub_L_arg_flg;                /* argument flag of the -L option of
                                                    the pjsub command */
uint64_t         pjsub_mpi_arg_flg;              /* argument flag of the --mpi option of
                                                    the pjsub command */
uint64_t         pjsub_step_arg_flg;             /* argument flag of the --step option of
                                                    the pjsub command */
uint64_t         pjsub_P_arg_flg;                /* argument flag of the -P option of
                                                    the pjsub command */
time_t           last_suspended_date;            /* For future extension */
time_t           last_resumed_date;              /* For future extension */
uint64_t         sum_suspended_time;             /* For future extension */
uint32_t         total_suspended_count;          /* For future extension */
uint8_t          pad5[3];
uint8_t          elapsed_time_mode;              /* Elapse time limit specification method */
uint64_t         adaptive_elapsed_time_min;      /* Elapse time limit minimum value */
uint64_t         adaptive_elapsed_time_max;      /* Elapse time limit maximum value */
uint64_t         job_env_boot_time;              /* Job execution environment boot time */
uint64_t         job_env_shutdown_time;          /* Job execution environment shutdown time */
int64_t          fj_profiler;                    /* Fujitsu profiler use count */
off_t            req_cstmrsc_ofs;                /* Offset to custom resource information
```

```
                                                       (Pjmapi_req_cstmrsc_t) */
   off_t            supplementary_info_ofs;         /* Offset to additional information */
   time_t           total_node_down_time;           /* For expansion */
   char             arch_info[16]                   /* Machine type */
   off_t            hw_info_ofs;                    /* Offset to hardware specific information */
}  Pjmapi_info_job_t;
```

```
typedef struct Pjmapi_sd {
   int              endcode_type;                   /* End code type */
   int              form_type;                      /* Condition format */
   int              form_value_num;                 /* Number of condition values */
   int              deletetype;                     /* Deletion type */
   int              to_stepno_num;                  /* Number of dependent step numbers */
   uint8_t          pad1[4];
   off_t            form_value_ofs;                 /* Offset to condition value */
   off_t            to_stepno_ofs;                  /* Offset to dependent step number */
} Pjmapi_sd_t;
```

```
typedef enum {
   PJM_CSTMRSC_VALUE_TYPE_NUMERIC = 1,
   PJM_CSTMRSC_VALUE_TYPE_STRING = 2
} Pjmapi_cstmrsc_value_type_t;
```

```
typedef struct Pjmapi_req_cstmrsc {
   off_t          next_ofs;                /* Offset where next custom resource information
                                             (Pjmapi_req_cstmrsc_t) is stored */
   char           name[PJM_MAX_CSTM_NAME_LEN]       /* Custom resource name */
   Pjmapi_cstmrsc_info_t  cstmrsc_info;             /* Custom resource type structure */
} Pjmapi_req_cstmrsc_t;
```

```
typedef struct Pjmapi_cstmrsc_info {
   uint8_t        is_pernode;  /* Whether NodeID is specified, 1 if specified */
   uint8_t        value_type;  /* Value type of custom resource (specifiable with
                                  pjmapi_cstmrsc_value_type_t) */
   uint8_t        pad[6];
   union {
        int64_t      num_value;                            /* Requested amount of custom resource */
        char         string_value[PJM_MAX_CSTM_NAME_LEN]; /* Requested type of custom resource */
   } value_rsc;
} Pjmapi_cstmrsc_info_t;
```

```
typedef struct Pjmapi_info_hwspecific_fx {
   uint64_t       tofu_user_comm_recv_byte;            /* Tofu user communication receive
                                                          data size (bytes) */
   uint64_t       tofu_user_comm_send_byte;            /* Tofu user communication send
                                                          data size (bytes) */
   uint64_t       tofu_sys_comm_rsv_byte;              /* Tofu system communication receive
                                                          data size (bytes) */
   uint64_t       tofu_sys_comm_send_byte;             /* Tofu system communication send
                                                          data size (bytes) */
   uint32_t       sum_alloc_assistcpunum;              /* Number of allocated assistant cores */
   uint32_t       sum_used_assistcpunum;               /* Number of assistant cores used */
   uint64_t       sum_usr_assistcputm;                 /* Total user CPU use time of
                                                          assistant core */
   uint64_t       sum_sys_assistcputm;                 /* Total system CPU use time of
                                                          assistant core */
   uint64_t       sum_used_assistant_core_max_mem;     /* Maximum use memory amount of
                                                          assistant core */
   uint64_t       sector_cache_using_program_count;    /* Start count of sector cache using
                                                          program */
   uint64_t       intra_node_barrier_using_program_count; /* Count of chip internal barrier using
                                                             program */
   Pjmapi_job_power_consumption_t power_consumption;   /* Power consumption-related information */
   Pjmapi_reserved_param_t reserved_param;             /* For future extension */
   Pjmapi_reserved_info_t reserved_info;               /* For future extension */
} Pjmapi_info_hwspecific_fx_t;
```

```
typedef struct Pjmapi_info_hwspecific_pcc {
   Pjmapi_job_power_consumption_pcc_t power_consumption; /* PC cluster power consumption-related
                                                            information */
} Pjmapi_info_hwspecific_pcc_t;
```

```
typedef struct Pjmapi_job_power_consumption {
   uint16_t       power_consumption_state;            /* Acquisition state of power information */
   uint8_t        utilization_info_of_power_api;      /* Power knob use information */
   uint8_t        pad[1];
   uint32_t       num_cmg;                            /* Number of CMGs */
   off_t          cmgs_ofs;                           /* Offset to power consumption structure
                                                         by CMG */
   Pjmapi_power_consumption_t ideal_cpu_peripherals;  /* Peripheral power consumption
                                                         information in CPU (estimation) */
   Pjmapi_power_consumption_t ideal_opticalmodule;    /* Optical module power consumption
                                                         information (estimation) */
   Pjmapi_power_consumption_t ideal_tofu;             /* Tofu power consumption information
                                                         (estimation) */
   Pjmapi_power_consumption_t ideal_pcie;             /* PCI-E power consumption information
                                                         (estimation) */
   Pjmapi_power_consumption_t ideal_node;             /* Node power consumption information
                                                         (estimation) */
   Pjmapi_power_consumption_t measured_node;          /* Node power consumption information
                                                         (result) */
   struct timespec measure_start_date;                /* Power measurement start time */
   struct timespec measure_end_date;                  /* Power measurement end time */
} Pjmapi_job_power_consumption_t;
```

```
typedef struct Pjmapi_job_power_consumption_pcc {
   uint16_t      power_consumption_state;             /* Acquisition state of power information */
   uint8_t       pad[2];
   uint32_t      num_pkg;                             /* Number of packages */
   off_t          pkgs_ofs;                           /* Offset to power consumption structure
                                                         by package */
   Pjmapi_power_consumption_t measured_node;          /* Node power consumption information
                                                         (result) */
   struct timespec measure_start_date;                /* Power measurement start time */
   struct timespec measure_end_date;                  /* Power measurement end time */
} Pjmapi_job_power_consumption_pcc_t;
```

```
typedef struct Pjmapi_cmg_power_consumption {
   int32_t cmgno;                                     /* CMG NUMBER */
   uint8_t pad[4];
   Pjmapi_power_consumption_t ideal_core;             /* Compute core power consumption
                                                         information by CMG (estimation) */
   Pjmapi_power_consumption_t ideal_l2cache;          /* L2 cache power consumption
                                                         information by CMG (estimation) */
   Pjmapi_power_consumption_t ideal_mem;              /* Memory power consumption
                                                         information by CMG (estimation) */
} Pjmapi_cmg_power_consumption_t;
```

```
typedef struct Pjmapi_pkg_power_consumption {
   int32_t pkgno;                                     /* Package number */
   uint8_t pad[4];
   Pjmapi_power_consumption_t cpu;                    /* CPU power consumption information
                                                         by package */
   Pjmapi_power_consumption_t mem;                    /* Memory power consumption information
```

```
                                                         by package */
   Pjmapi_power_consumption_t pp0;                    /* PP0 power consumption information
                                                         by package */
} Pjmapi_pkg_power_consumption_t;
```

```
typedef struct Pjmapi_power_consumption {
   double           avg_power;                        /* Average power consumption */
   double           max_power;                        /* Maximum power consumption */
   double           min_power;                        /* Minimum power consumption */
   double           energy;                           /* Power consumption amount */
} Pjmapi_power_consumption_t;
```

```
typedef struct Pjmapi_reserved_param {
   uint64_t         reserved1;                        /* For future extension */
   uint64_t         reserved2;                        /* For future extension */
   uint64_t         reserved3;                        /* For future extension */
   uint64_t         reserved4;                        /* For future extension */
   uint64_t         reserved5;                        /* For future extension */
   int32_t          reserved6;                        /* For future extension */
   int32_t          reserved7;                        /* For future extension */
   int32_t          reserved8;                        /* For future extension */
   int32_t          reserved9;                        /* For future extension */
   uint64_t         reserved10;                       /* For future extension */
   uint64_t         reserved11;                       /* For future extension */
   off_t            reserved12;                       /* For future extension */
   off_t            reserved13;                       /* For future extension */
} Pjmapi_reserved_param_t;
```

```
typedef struct Pjmapi_reserved_info {
   struct timespec  reserved1;                        /* For future extension */
   struct timespec  reserved2;                        /* For future extension */
   struct timespec  reserved3;                        /* For future extension */
   struct timespec  reserved4;                        /* For future extension */
   struct timespec  reserved5;                        /* For future extension */
   struct timespec  reserved6;                        /* For future extension */
   uint64_t         reserved7;                        /* For future extension */
   double           reserved8;                        /* For future extension */
   double           reserved9;                        /* For future extension */
   uint64_t         reserved10;                       /* For future extension */
   uint64_t         reserved11;                       /* For future extension */
   uint64_t         reserved12;                       /* For future extension */
   uint64_t         reserved13;                       /* For future extension */
   uint64_t         reserved14;                       /* For future extension */
   uint64_t         reserved15;                       /* For future extension */
   uint64_t         reserved16;                       /* For future extension */
   double           reserved17;                       /* For future extension */
   uint64_t         reserved18;                       /* For future extension */
   uint64_t         reserved19;                       /* For future extension */
   uint64_t         reserved20;                       /* For future extension */
   uint64_t         reserved21;                       /* For future extension */
   double           reserved22;                       /* For future extension */
   uint64_t         reserved23;                       /* For future extension */
   uint64_t         reserved24;                       /* For future extension */
   uint64_t         reserved25;                       /* For future extension */
   uint64_t         reserved26;                       /* For future extension */
} Pjmapi_reserved_info_t;
```

Table E.3 Members of Job Information Notification Structure PJM_INFO_JOB

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
jobstatus	int16_t	out	The current status of the target job is stored.
job_aprio	int16_t	out	The job priority level within the resource unit is stored.
job_uprio	int16_t	out	The job priority level within the same user is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
subjob_num	int	out	The number of the sub jobs of the target job is stored.
exit_code	uint	out	The exit code of the shell script of the target job is stored.
signal_no	int32_t	out	The signal number of the shell script of the target job is stored.
pjm_code	int	out	A code indicating the processing result of the job manager function in job execution of the target job is stored.
node_x	uint	out	The allocated shape of the target job is stored.
node_y	uint		
node_z	uint		
node_num	uint	out	The number of the allocated nodes of the target job is stored.
uid	uid_t	out	The uid of the job submitter is stored.
exec_gid	gid_t	out	The gid with which the job is executed is stored.
node_req_x	uint	out	The number of the requested nodes in the direction of x, y, or z is stored.
node_req_y	uint		
node_req_z	uint		
node_req_num	uint	out	The number of requested nodes of the job is stored.
node_mpi_x	uint	out	The number of the mpi option-specified nodes in the direction of x, y, or z is stored.
node_mpi_y	uint		
node_mpi_z	uint		

Member	Type	Input-Output	Description
node_mpi_num	uint	out	The number of the mpi option-specified nodes of the job is stored.
proc	int	out	The number of the processes of the job is stored.
proc_bynode	int	out	The number of the processes of the job for 1 node is stored.
sd_num	int	out	The number of the dependence relation expressions set for the job is stored.
lasthold_uid	uid_t	out	If the target job has ever been held, the user ID of the last user who held it is stored. If the job was canceled, the user ID of the user who canceled it is stored.
subjobflag	uint32_t	out	The attribute of waiting for transfer of the result of the preceding step job of the target job is stored.
hold_count	uint	out	The number of times when the target job was held is stored.
run_count	int	out	The number of times when the target job is re-executed is stored.
unavailable_nodenum	uint	out	The number of available nodes in the allocated scope of the target job is stored.
sum_cpu_req_num	uint	out	The number of CPUs requested by the target job is stored.
used_cpunum	uint32_t	out	The number of CPUs used by the target job is stored.
umask	int	out	The umask value of the submitter user of the target job (value converted into a decimal number) is stored.
mailflag	int	out	The flag about whether there is mail transfer of the target job is stored. The value is as follows. 1: Job start 2: Job end 4: Error occurrence 8: Statistical information output (without node information) 16: Statistical information output (with node information)
pro_exit_code	uint	out	The end code of the prologue script of the target job is stored.
epi_exit_code	uint	out	The end code of the epilogue script of the target job is stored.
vn_cpu_req	uint32_t	out	The number of allocated cores of the target job by virtual node is stored.
rankmap_type	int	out	The rank map of the target job is stored. The value is any of the following. 0: No specification 1: rank-map-bynode 2: rank-map-bychip
rankmap_num	int	out	The number of placed rank maps of the target job is stored.
node_type	int	out	The node type of the target job is stored.
num_alloc_vnode	uint32_t	out	The number of allocated virtual nodes of the target job is stored.
sum_cpu_alloc_num	uint	out	The total number of the allocated CPUs is stored.
used_nodenum	uint	out	The number of nodes used is stored.
sum_cpu_prealloc_num	uint	out	The total number of CPUs allocated by the scheduler function is stored.

Member	Type	Input-Output	Description
create_date	time_t	out	The time when the target job was submitted (execution time of the pjsub command) is stored.
start_date	time_t	out	The job start time of the target job is stored.
end_date	time_t	out	The job end time of the target job is stored.
elapse	time_t	out	The job elapse time of the target job is stored.
elapse_off_acc	time_t	out	Of the job elapse time of the target job, the time that is not subject to billing is stored.
sched_date	time_t	out	The job execution start time of the target job is stored.
que_date	time_t	out	The time of the transition of the target job to QUEUED is stored.
runa_date	time_t	out	The time of the transition of the target job to RUNNING-A is stored.
run_date	time_t	out	The time of the transition of the target job to RUNNING is stored.
runout_date	time_t	out	The time of the transition of the target job to RUNOUT is stored.
exit_date	time_t	out	The time of the transition of the target job to EXIT is stored.
cancel_date	time_t	out	The time of the transition of the target job to CANCEL is stored.
hold_date	time_t	out	The time of the transition of the target job to HOLD is stored.
err_date	time_t	out	The time of the transition of the target job to ERROR is stored.
attribute	uint64_t	out	The attribute of the job is set by the following flags. 0x000002: Specification of strict 0x004000: Specification of strict-io
spec_date	time_t	out	The specification of the date and time when the job execution to be started is stored.
elapse_limit	uint64_t	out	The limit value of the elapsed time is stored. In case of UNLIMITED, PJM_RLIM_INFINITY(~0ULL) is stored. If limit values of the elapsed time are specified as a range, the maximum time of the limit of the elapse time (seconds) is set.
jobname[64]	char[]	out	The job name of the target job is stored.
rscunit_name[PJM_RSCUNAME_MAX]	char[]	out	The resource unit name of the job is stored.
rscgrp_name[PJM_RSCGROUP_MAX]	char[]	out	The resource group name of the job is stored.
mem_req	uint64_t	out	The requested memory amount by node is stored.
node_cpu	uint64_t	out	The limit value of CPU time by node is stored.
reject_date	time_t	out	The transition time to REJECT is stored.
hostname[16]	char[]	out	The node name of the node where the target job was submitted (up to 15 characters from the beginning) is stored.
reason[64]	char[]	out	The REASON of the target job is stored.

Member	Type	Input-Output	Description
fst_start_date	time_t	out	The time when the target job transition to the RUNNING state occurred for the first time is stored.
sum_runa_time	uint64_t	out	The cumulative time of the RUNNING-A state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_run_time	uint64_t	out	The cumulative time of the RUNNING state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_hold_time	uint64_t	out	The cumulative time of the HOLD state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_wait_time	uint64_t	out	The cumulative wait time of the target job is stored.
mem_lmt	uint64_t	out	The memory amount limit value of the job in the node of the target job (bytes) is stored.
mem_job_alloc	uint64_t	out	The memory amount allocated in the node of the target job (bytes) is stored.
prc_cputm_lmt	uint64_t	out	The limit value of CPU use time of the target job by process (seconds) is stored.
prc_corefile_lmt	uint64_t	out	The limit value of the core file size of the target job by process (bytes) is stored.
prc_cre_proc_lmt	uint64_t	out	The limit value of the number of user processes of the target job by process is stored.
prc_data_lmt	uint64_t	out	The limit value of the data segment of the target job by process is stored.
prc_locked_mem_lmt	uint64_t	out	The limit value of the locked memory of the target job by process is stored.
prc_psx_msq_que_lmt	uint64_t	out	The limit value of the POSIX message queue of the target job by process is stored.
prc_openfiles_lmt	uint64_t	out	The limit value of the file descriptor of the target job by process is stored.
prc_pndng_sgnl_lmt	uint64_t	out	The limit value of the number of signal of the target job by process is stored.
prc_prmfl_lmt	uint64_t	out	The limit value of the file size of the target job by process is stored.
prc_stack_lmt	uint64_t	out	The limit value of the stack segment of the target job by process is stored.
prc_vmem_lmt	uint64_t	out	The limit value of the virtual memory size of the target job by process is stored.
max_used_mem	uint64_t	out	The maximum memory usage of the target job is stored.
usctmut	uint64_t	out	The total user CPU time and total system CPU time of the target job (seconds) are stored.
snapshottime	time_t	out	The data collection date (year/month/day) of the target job is stored.
pjdel_date	time_t	out	The job deletion request time of the target job is stored.
delete_date	time_t	out	The job deletion time of the target job is stored.
all_prec_subjob_exit_date	time_t	out	The preceding sub job end time of the target job is stored.
fsname[64]	char[]	out	The file system name of the target job is stored.

Member	Type	Input-Output	Description
appname[64]	char[]	out	The application name of the target job is stored.
prologue_start_date	time_t	out	The prologue start time in the compute node of the target job is stored.
runp_date	time_t	out	The time of the transition of the state of the PJM of the target job to RUNNING_P is stored.
epilogue_start_date	time_t	out	The epilogue start time in the compute node of the target job is stored.
rune_date	time_t	out	The time of the transition of the state of the PJM of the target job to RUNNING_E is stored.
prologue_end_date	time_t	out	The prologue end time in the compute node of the target job is stored.
epilogue_end_date	time_t	out	The epilogue end time in the compute node of the target job is stored.
pack_policy	uint64_t	out	The virtual node policy of the target job is stored. The value is as follows. 0: PACK 1: UNPACK 2: Absolutely PACK 3: Absolutely UNPACK
exec_policy	uint64_t	out	The execution mode policy of the target job is stored. The value is as follows. 0: SHARE 1: SIMPLEX
sum_vm_job_use	uint64_t	out	The total memory amount of the target job by virtual node (bytes) is stored.
sum_usr_cputm	uint64_t	out	The total user CPU use time is stored.
sum_sys_cputm	uint64_t	out	The total system CPU use time is stored.
mem_job_prealloc	uint64_t	out	The allocated memory amount of the target job is stored.
curdir_ofs	off_t	out	The offset to the character string, in which the job submission directory set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. char* curdir = (char*) PJMAPI_OFF_TO_PTR(curdir_ofs)
mail_ofs	off_t	out	The offset to the character string, in which the mail address set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. char* mail = (char*) PJMAPI_OFF_TO_PTR (mail_ofs)
shell_ofs	off_t	out	The offset to the character string, in which the shell script set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. char* shell = (char*) PJMAPI_OFF_TO_PTR(shell_ofs)
comment_ofs	off_t	out	The offset to the character string, in which the comment set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string.

Member	Type	Input-Output	Description
			char* comment = (char*) PJMAPI_OFF_TO_PTR(comment_ofs)
stdout_ofs	off_t	out	The offset to the character string, in which the path of the standard output file set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. char* stdout = (char*) PJMAPI_OFF_TO_PTR(stdout_ofs)
stderr_ofs	off_t	out	The offset to the character string, in which the path of the standard error output file set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. char* stderr = (char*) PJMAPI_OFF_TO_PTR (stderr_ofs)
job_acct_ofs	off_t	out	The offset to the character string, in which the path of the job statistical information file set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. char* job_acct= (char*) PJMAPI_OFF_TO_PTR(job_acct_ofs)
ndlist_ofs	off_t	out	The offset to the character string, in which the node ID list set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. int nolist= (int *) PJMAPI_OFF_TO_PTR(ndlist_ofs)
tofulist_ofs	off_t	out	The offset to the character string, in which the Tofu coordinate list set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. tofu_3d_t tofulist= (tofu_3d_t *) PJMAPI_OFF_TO_PTR(tofulist_ofs) (*1) The Tofu coordinates are stored in the structure of the Tofu coordinates. For details, see (*1) indicated outside of this table.
sd_ofs	off_t	out	The offset to the dependence relation expression structure, in which the dependence relation expression information set for the job is stored, is stored. (*2) The following expression can be used to obtain the pointer to the relevant structure. pjmapi_sd_t* sd = (sd_t*) PJMAPI_OFF_TO_PTR(sd_ofs)
req_mpi_static_proc	int32_t	out	The number of requested MPI static processes set for the job is stored.
req_mpi_proc	int32_t	out	The number of requested MPI processes set for the job is stored.
alloc_mpi_static_proc	int32_t	out	The number of allocated MPI static processes set for the job is stored.
alloc_mpi_proc	int32_t	out	The number of allocated MPI processes set for the job is stored.
numa_policy	uint64_t	out	The NUMA policy set for the job is stored. The value to be set is as follows. 0: PACK 1: UNPACK

Member	Type	Input-Output	Description
start_blkno	uint32_t	out	The bulk start number set for the job is stored.
end_blkno	uint32_t	out	The bulk end number set for the job is stored.
affected_nid	uint32_t	out	The node ID that affected the job result of the target job is stored.
prealloc_rmexit_exitcode	uint32_t	out	The end code of the prealloc exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
predel_rmexit_exitcode	uint32_t	out	The end code of the predel exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
postfree_rmexit_exitcode	uint32_t	out	The end code of the postfree exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
prealloc_start_time	uint64_t	out	The time of the start of the prealloc exit of the target job is stored.
prealloc_end_time	uint64_t	out	The time of the end of the prealloc exit of the target job is stored.
predel_start_time	uint64_t	out	The time of the start of the predel exit of the target job is stored.
predel_end_time	uint64_t	out	The time of the end of the predel exit of the target job is stored.
postfree_start_time	uint64_t	out	The time of the start of the postfree exit of the target job is stored.
postfree_end_time	uint64_t	out	The time of the end of the postfree exit of the target job is stored.
prealloc_exec_kind	uint8_t	out	The prealloc exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 1: Job start timing

Member	Type	Input-Output	Description
predel_exec_kind	uint8_t	out	The predel exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 3: pjdel command execution timing 4: pjhold command execution timing 5: Timing of a deletion request from the job manager function or job scheduler function 6: Timing of a compute node error 7: Timing when the CPU time is exceeded 8: Timing when the elapsed time is exceeded 9: Timing when the memory use amount exceeded
postfree_exec_kind	uint8_t	out	The postfree exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 2: Job end timing
backfill_flg	uint8_t	out	The flag of a backfilled job
last_sched_date	struct timespec	out	The scheduling start time of the target job is stored.
pjsub_option_flg	uint64_t	out	The options specified for the pjsub command are stored as flags. The stored values are as follows. 0x0000000000000001: --at 0x0000000000000002: --bulk 0x0000000000000004: --dir-prefix 0x0000000000000080: --comment 0x0000000000000200: -e 0x0000000000000800: --gid 0x0000000000001000: --gname 0x0000000000008000: --interact 0x0000000000010000: -j 0x0000000000020000: --rsc-list 0x0000000000040000: -m 0x0000000000080000: --mail-list 0x0000000000100000: --mpi 0x0000000000400000: --name 0x0000000001000000: --norestart 0x0000000004000000: -o 0x0000000020000000: -p 0x0000000040000000: --restart 0x0000000080000000: -s 0x0000000100000000: -S 0x0000000400000000: --sparam 0x0000000800000000: --spath 0x0000001000000000: --step 0x0000200000000000: --vset 0x0000400000000000: -w 0x0000800000000000: -x 0x0001000000000000: -X 0x0020000000000000: --reason 0x0040000000000000: --fs 0x0080000000000000: --appname 0x0100000000000000: -P--vn-policy 0x0200000000000000: -P--exec-policy 0x0400000000000000: -P

Member	Type	Input-Output	Description
pjsub_option_exflg	uint64_t	out	The options specified for the pjsub command are stored as flags. The stored values are as follows. 0x0000000000000001:--net-route dynamic
pjsub_L_arg_flg	uint64_t	out	The options specified for the pjsub command are stored as flags. The stored values are as follows. 0x0000000000000001: node 0x0000000000000004: elapse 0x0000000000000008: node-mem 0x0000000000000020: rscunit 0x0000000000000040: rscgrp 0x0000000000000100: proc-core 0x0000000000000200: proc-cpu 0x0000000000000400: proc-crproc 0x0000000000000800: proc-data 0x0000000000001000: proc-lockm 0x0000000000002000: proc-msgq 0x0000000000004000: proc-openfd 0x0000000000008000: proc-psig 0x0000000000010000: proc-filesz 0x0000000000020000: proc-stack 0x0000000000040000: proc-vmem 0x0000000000800000: extended resource
pjsub_mpi_arg_flg	uint64_t	out	The arguments specified for the --mpi option of the pjsub command are set as flags and stored. The stored values are as follows. 0x0000000000000001: shape 0x0000000000000002: proc 0x0000000000000004: rank-map-bynode 0x0000000000000008: rank-map-bychip 0x0000000000000010: rank-map-hostfile 0x0000000000000020: assign-online-node
pjsub_step_arg_flg	uint64_t	out	The arguments specified for the -- step option of the pjsub command are set as flags and stored. The stored values are as follows. 0x0000000000000001: jid 0x0000000000000002: sd 0x0000000000000004: sn 0x0000000000000008: send 0x0000000000000010: jnam
pjsub_P_arg_flg	uint64_t	out	The arguments specified for the -P option of the pjsub command are set as flags and stored. The stored values are as follows. 0x0000000000000001: vn-policy 0x0000000000000002: exec-policy
last_suspended_date	time_t	out	It is used for future extension.
last_resumed_date	time_t	out	It is used for future extension.
sum_suspend_time	uint64_t	out	It is used for future extension.
total_suspended_count	uint32_t	out	It is used for future extension.

Member	Type	Input-Output	Description
elapsed_time_mode	uint8_t	out	The specification method of the elapsed time of the target job The following values are stored. PJM_ELAPSED_TIME_MODE_ADAPTIVE: Specification method for when the elapsed time limit values are specified as a range PJM_ELAPSED_TIME_MODE_FIXED: Specification method for when the elapsed time limit values are not specified as a range
adaptive_elapsed_time_min	uint64_t	out	The minimum elapsed time value of a job for which the target elapsed time limit values are specified as a range is stored. If the elapsed time limit values are not specified as a range, 0 is set.
adaptive_elapsed_time_max	uint64_t	out	The maximum elapse time value of a job for which the target elapsed time limit values are specified as a range is stored. If the elapsed time limit values are not specified as a range, 0 is set.
job_env_boot_time	uint64_t	out	The time taken for the boot processing of the job execution environment is stored.
job_env_shutdown_time	uint64_t	out	The time taken for the shutdown processing of the job execution environment is stored.
fj_profiler	int64_t	out	The number of times of Fujitsu profiler use by the target job is stored.
req_cstmrsc_ofs	off_t	out	The offset to the requested custom resource information (Pjmapi_req_cstmrsc_t) is stored. The following expression can be used to obtain the pointer to the requested custom resource information. Pjmapi_req_cstmrsc_t* req_cstm_p = (Pjmapi_req_cstmrsc_t *) PJMAPI_OFF_TO_PTR(req_cstmrsc_ofs) If the requested custom resource information does not exist, 0 is stored.
supplementary_info_ofs	off_t	out	The offset to the additional information (char) is stored. The following expression can be used to obtain the pointer to the additional information. char* buff_supliinfo_p = (char *) PJMAPI_OFF_TO_PTR(supplementary_info_ofs) If the additional information does not exist, 0 is stored.
total_node_down_time	time_t	out	For extension
arch_info[16]	char[]	out	The character string indicating a machine type is stored.
hw_info_ofs	off_t	out	The offset to the hardware specific information is stored. The following expression can be used to obtain the pointer to the hardware specific information by machine type. For machine type FX: Pjmapi_info_hwspecific_fx_t *hw_fx_info_p = (Pjmapi_info_hwspecific_fx_t*) PJMAPI_OFF_TO_PTR(hw_info_ofs) If the hardware specific information does not exist, 0 is stored. For machine type PG: Pjmapi_info_hwspecific_pcc_t *hw_pcc_info_p = (Pjmapi_info_hwspecific_pcc_t*)

Member	Type	Input-Output	Description
			PJMAPI_OFF_TO_PTR(hw_info_ofs) If the hardware specific information does not exist, 0 is stored.
endcode_type	int	out	The end code type of a dependence relation expression is stored. Either one of the following values is set. PJM_SD_ENDCODE_TYPE_EC: The end status of the shell is used for dependence judgment. PJM_SD_ENDCODE_TYPE_PC: The PJM code is used for dependence judgment.
form_type	int	out	A dependence relation expression is stored. Any one of the following values is set. PJM_SD_FORM_TYPE_EQ: = form_value PJM_SD_FORM_TYPE_NE: != form_value PJM_SD_FORM_TYPE_GT: > form_value PJM_SD_FORM_TYPE_LT: < form_value PJM_SD_FORM_TYPE_GE: >= form_value PJM_SD_FORM_TYPE_LE: <=form_value
form_value_num	int	out	The value of dependence relation expression is stored.
deletetype	int	out	A deletion method of a dependence relation expression is stored. Any one of the following values is set. PJM_SD_DELTYPE_ONE: Only the relevant sub job is deleted. PJM_SD_DELTYPE_AFTER: The relevant sub job and subsequent dependent sub jobs are deleted. PJM_SD_DELTYPE_ALL: All the sub jobs are deleted.
to_steno_num	int	out	The number of step numbers on which the job is dependent is stored.
form_value_ofs	off_t	out	The offset to the value of the dependence relation expression is stored. The following expression can be used to obtain the pointer. i int form_value[] = (int*) PJMAPI_OFF_TO_PTR(form_value_ofs)
to_stepno_ofs	off_t	out	The offset to the step number on which the job is dependent is stored. The following expression can be used to obtain the pointer. int to_stepno[] = (int*) PJMAPI_OFF_TO_PTR(to_stepno_ofs)
next_ofs	off_t	out	The offset to the next custom resource information (Pjmapi_req_cstmrsc_t) is stored. If the next custom resource information does not exist, 0 is stored.
name[PJM_MAX_CSTM_NAME_LEN]	char[]	out	The custom resource name is stored.
cstmrsc_info	Pjmapi_cstmrsc_info_t	out	The requested amount or requested type of the custom resource information is stored.
is_pernode	uint8_t	out	If the custom resource is a resource by node, 1 is set. If the custom resource is not a resource by node, 0 is set.

Member	Type	Input-Output	Description
value_type	uint8_t	out	The value type of the custom resource (value that can be specified by pjmapi_cstmrsc_value_type_t) Either numeric specification (PJM_CSTMRSC_VALUE_TYPE_NUMERIC) or type specification(PJM_CSTMRSC_VALUE_TYPE_STRING) is stored.
value_rsc	union	out	The requested amount or requested type of the custom resource information is stored.
num_value	int64_t	out	The requested amount of the custom resource information is set.
string_value[PJM_MAX_CSTM_NAME_LEN]	char[]	out	The type of the custom resource information is set.
tofu_user_comm_recv_byte	uint64_t	out	The receive data size used for user level communication of the target job via the Tofu interconnect (bytes) is stored.
tofu_user_comm_send_byte	uint64_t	out	The send data size used for user level communication of the target job via the Tofu interconnect (bytes) is stored.
tofu_sys_comm_rsv_byte	uint64_t	out	The receive data size used for system communication of the target job via the Tofu interconnect (bytes) is stored.
tofu_sys_comm_send_byte	uint64_t	out	The send data size used for system communication of the target job via the Tofu interconnect (bytes) is stored.
sum_alloc_assistcpunum	uint32_t	out	The number of allocated assistant cores set for the job is stored.
sum_used_assistcpunum	uint32_t	out	The number of used assistant cores set for the job is stored.
sum_usr_assistcputm	uint64_t	out	The total CPU use time of users of the assistant cores set for the job is stored.
sum_sys_assistcputm	uint64_t	out	The total CPU use time of the system of the assistant cores set for the job is stored.
sum_used_assistant_core_max_mem	uint64_t	out	The maximum use amount of the memory used by the assistant cores of the target job (bytes) is stored.
sector_cache_using_program_count	uint64_t	out	The number of times when programs that use the sector cache of the target job are started is stored.
intra_node_barrier_using_program_count	uint64_t	out	The number of times when programs that use the chip internal barrier of the target job are started is stored.
power_consumption	Pjmapi_job_power_consumption_t	out	The power consumption-related information is stored.
reserved_param	Pjmapi_reserved_param_t	out	It is used for future extension.
reserved_info	Pjmapi_reserved_info_t	out	It is used for future extension.
power_consumption	Pjmapi_job_power_consumption_pcc_t	out	Power consumption-related information of the PC cluster is stored.
power_consumption_state	uint16_t	out	The acquisition status of power information of the target job is stored as a value of logical addition flags. The value is as follows. 0x0: The obtained information does not include nodes affected by node-sharing jobs. 0x1: The obtained information includes nodes affected by

Member	Type	Input-Output	Description
			node-sharing jobs. 0x2: There are nodes whose information failed to be obtained. 0x4: Since there are preceding jobs, information of some nodes was not obtained.
utilization_info_of_power_api	uint8_t	out	Whether the target job uses the Power API and whether the power knob is operated are stored with the corresponding bits set on or off. The value is as follows. 0b00(0): The API is not used and the knob is not operated. 0b01(1): The API is used and the knob is not operated. 0b03(3): The API is used and the knob is operated.
num_cmg	uint32_t	out	The number of CMGs of the target job is stored.
cmgs_ofs	off_t	out	The offset to the power consumption structure by CMG is stored. The following expression can be used to obtain the pointer to the power consumption structure by CMG. Pjmapi_cmg_power_consumption_t *cmgs_p =(Pjmapi_cmg_power_consumption_t *)PJMAPI_OFF_TO_PTR(cmgs_ofs)
ideal_cpu_peripherals	Pjmapi_power_consumption_t	out	Peripheral power consumption information in the CPU (estimation) is stored.
ideal_opticalmodule	Pjmapi_power_consumption_t	out	Optical module power consumption information (estimation) is stored.
ideal_tofu	Pjmapi_power_consumption_t	out	Tofu power consumption information (estimation) is stored.
ideal_pcie	Pjmapi_power_consumption_t	out	PCI-E power consumption information (estimation) is stored.
ideal_node	Pjmapi_power_consumption_t	out	Node power consumption information (estimation) is stored.
measured_node	Pjmapi_power_consumption_t	out	Node power consumption information (result) is stored.
measure_start_date	struct timespec	out	The power measurement start time is stored.
measure_end_date	struct timespec	out	The power measurement end time is stored.
num_pkg	uint32_t	out	The number of packages of the target job is stored.
pkgs_ofs	off_t	out	The offset to the power consumption structure by package is stored.
cmgno	int32_t	out	The CMG number is stored.
ideal_core	Pjmapi_power_consumption_t	out	The compute core power consumption information by CMG (estimation) is stored.
ideal_l2cache	Pjmapi_power_consumption_t	out	The L2 cache power consumption information by CMG (estimation) is stored.
ideal_mem	Pjmapi_power_consumption_t	out	The memory power consumption information by CMG (estimation) is stored.
pkgno	int32_t	out	Package number is stored.
cpu	Pjmapi_power_consumption_t	out	The CPU power consumption information by package is stored.

Member	Type	Input-Output	Description
mem	Pjmapi_power_consumption_t	out	The memory power consumption information by package is stored.
pp0	Pjmapi_power_consumption_t	out	The pp0 power consumption information by package is stored.
avg_power	double	out	The average power consumption of the target power items is stored.
max_power	double	out	The maximum power consumption of the target power items is stored.
min_power	double	out	The minimum power consumption of the target power items is stored.
energy	double	out	The power consumption amount of the target power items is stored.

(*1)
The structure of Tofu coordinates is as follows.

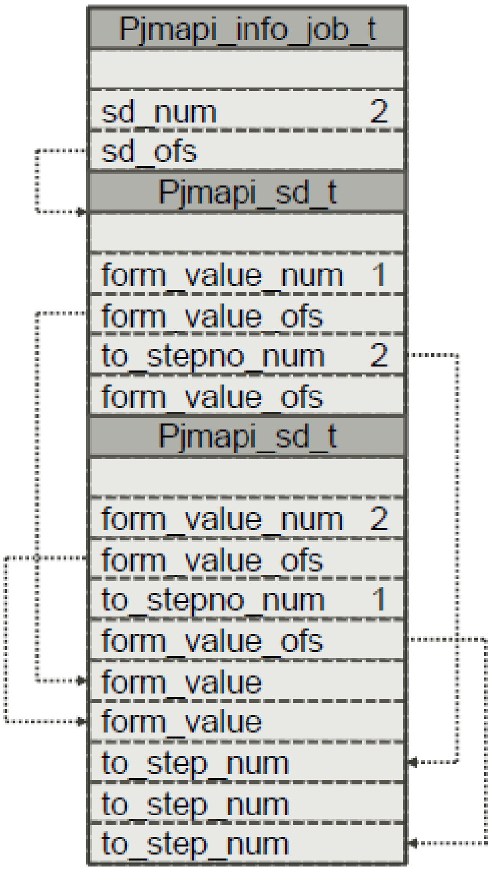
```
typedef struct tofu_3d{
            unsigned int    x;
            unsigned int    y;
            unsigned int    z;
            unsigned int    pad;
} tofu_3d_t;
```

Table E.4 Members of Tofu coordinate Structure

Member	Type	Input-Output	Description
x	unsigned int	out	A Tofu x coordinate is stored.
y	unsigned int	out	A Tofu y coordinate is stored.
z	unsigned int	out	A Tofu z coordinate is stored.

(*2)
The offset to the dependence relation expression structure, in which the dependence relation expression information set for the job is stored, is stored in the argument off_t sd_ofs. However, for a step job, the dependence relation expressions, expression values that can be specified for dependence relation expressions, and the number of dependent jobs are variable. Information of a step job is accessed using the offset. The following example has two dependence relation expressions. The first dependence relation expression has two values and one dependent job. The second dependence relation expression has one value and two dependent jobs.

Figure E.1 Structure Example of a Dependence Relation Expression



(*3) The member reserved*n* is for future expansion.

E.1.2 Notification Structure of Change to the ACCEPT State (PJM_CHANGE_ACCEPT)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_ACCEPT. The following is the notification structure of change to the ACCEPT state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_accept {
   uint16_t           job_model;      /* Job model */
   uint16_t           job_flags;      /* Job additional information */
   uint32_t           job_type;       /* Job type */
   uint32_t           jobid;          /* Job ID */
   uint32_t           blkno;          /* Bulk number */
   uint32_t           stepno;         /* Step number */
   uint32_t           pad2;
   time_t             create_date;    /* Job submission time */
} Pjmapi_change_accept_t;
```

Table E.5 Members of Notification Structure of Change to the ACCEPT State PJM_CHANGE_ACCEPT

Member	Type	Input-Output	Description
job_model	uint16_t	out	The bit indicated by any one of the following macros is set according to the job model.

Member	Type	Input-Output	Description
			PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
create_date	time_t	out	The time when the target job was submitted (execution time of the pjsub command) is stored.

E.1.3 Notification Structure of Change to the QUEUED State (PJM_CHANGE_QUEUED)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_QUEUED. The following is the notification structure of change to the QUEUED state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_queued {
   uint16_t          job_model;                  /* Job model */
   uint16_t          job_flags;                  /* Job additional information */
   uint16_t          num_retry;                  /* Retry count */
   int16_t           pre_jobstatus;              /* Previous job status */
   uint32_t          job_type;                   /* Job type */
   uint32_t          jobid;                      /* Job ID */
   uint32_t          blkno;                      /* Bulk number */
   uint32_t          stepno;                     /* Step number */
   int               subjob_num;                 /* Number of sub jobs */
   uint              pro_exit_code;              /* Prologue exit code */
   time_t            que_date;                   /* Time of QUEUED transition */
   uint64_t          sum_runa_time;              /* Cumulative RUNNING-A time (seconds) */
   uint64_t          sum_hold_time;              /* Cumulative HOLD time (seconds) */
   time_t            prologue_end_date;          /* Prologue end time */
   time_t            all_prec_subjob_exit_date;  /* Preceding sub job end time */
   uint32_t          affected_nid;               /* Node ID that affected job result */
   uint32_t          prealloc_rmexit_exitcode;   /* prealloc exit end code */
   uint32_t          predel_rmexit_exitcode;     /* predel exit end code */
   uint32_t          postfree_rmexit_exitcode;   /* postfree exit end code */
   uint64_t          prealloc_start_time;        /* prealloc exit start time */
   uint64_t          prealloc_end_time;          /* prealloc exit end time */
   uint64_t          predel_start_time;          /* predel exit start time */
   uint64_t          predel_end_time;            /* predel exit end time */
   uint64_t          postfree_start_time;        /* postfree exit start time */
   uint64_t          postfree_end_time;          /* postfree exit end time */
   uint8_t           prealloc_exec_kind;         /* prealloc exit execution timing */
   uint8_t           predel_exec_kind;           /* predel exit execution timing */
   uint8_t           postfree_exec_kind;         /* postfree exit execution timing */
   uint8_t           pad1[5];
} Pjmapi_change_queued_t;
```

Table E.6 Members of Notification Structure of Change to the QUEUED State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
subjob_num	int	out	The number of the sub jobs of the target job is stored.
pro_exit_code	uint	out	The end code of the prologue script of the target job is stored.
que_date	time_t	out	The time of the transition of the target job to QUEUED is stored.
sum_runa_time	uint64_t	out	The cumulative time of the RUNNING-A state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_hold_time	uint64_t	out	The cumulative time of the HOLD state of the target job (seconds, rounded up to the nearest whole digit) is stored.
prologue_end_date	time_t	out	The prologue end time in the compute node of the target job is stored.
all_prec_subjob_exit_date	time_t	out	The preceding sub job end time of the target job is stored.
affected_nid	uint32_t	out	The node ID that affected the job result of the target job is stored.
prealloc_rmexit_exitcode	uint32_t	out	The end code of the prealloc exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
predel_rmexit_exitcode	uint32_t	out	The end code of the predel exit of the target job is stored. The value to be set is as follows.

Member	Type	Input-Output	Description
			0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
postfree_rmexit_exitcode	uint32_t	out	The end code of the postfree exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
prealloc_start_time	uint64_t	out	The time of the start of the prealloc exit of the target job is stored.
prealloc_end_time	uint64_t	out	The time of the end of the prealloc exit of the target job is stored.
predel_start_time	uint64_t	out	The time of the start of the predel exit of the target job is stored.
predel_end_time	uint64_t	out	The time of the end of the predel exit of the target job is stored.
postfree_start_time	uint64_t	out	The time of the start of the postfree exit of the target job is stored.
postfree_end_time	uint64_t	out	The time of the end of the postfree exit of the target job is stored.
prealloc_exec_kind	uint8_t	out	The prealloc exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 1: Job start timing
predel_exec_kind	uint8_t	out	The predel exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 3: pjdel command execution timing 4: pjhold command execution timing 5: Timing of a deletion request from the job manager function or job scheduler function 6: Timing of a compute node error 7: Timing when the CPU time is exceeded 8: Timing when the elapsed time is exceeded 9: Timing when the memory use amount exceeded
postfree_exec_kind	uint8_t	out	The postfree exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 2: Job end timing

E.1.4 Notification Structure of Change to the RUNNING-A State (PJM_CHANGE_RUNNING_A)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_RUNNING_A. The following is the notification structure of change to the RUNNING-A state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_running_a {
   uint16_t         job_model;                /* Job model */
   uint16_t         job_flags;                /* Job additional information */
   uint16_t         num_retry;                /* Retry count */
   int16_t          pre_jobstatus;            /* Previous job status */
   uint32_t         job_type;                 /* Job type */
   uint32_t         jobid;                    /* Job ID */
   uint32_t         blkno;                    /* Bulk number */
   uint32_t         stepno;                   /* Step number */
   time_t           runa_date;                /* Time of RUNNING-A transition */
   time_t           all_prec_subjob_exit_date;  /* Preceding sub job end time */
   uint64_t         sum_wait_time;            /* Cumulative wait time */
   uint             node_num;                 /* Number of allocated nodes */
   uint             node_x;                   /* Allocated node shape x */
   uint             node_y;                   /* Allocated node shape y */
   uint             node_z;                   /* Allocated node shape z */
   uint32_t         vn_cpu_req;               /* Requested number of CPU cores
                                                 by virtual node */
   uint32_t         num_alloc_vnode;          /* Number of allocated virtual nodes */
   uint             sum_cpu_prealloc_num;     /* Total number of scheduler allocation CPUs */
   uint             pad2;
   time_t           sched_date;               /* Job execution start time */
   uint64_t         mem_job_prealloc;         /* Scheduler allocation memory amount */
   struct timespec  last_sched_date;          /* Scheduling start time */
   uint8_t          backfill_flg;             /* Backfill flag */
} Pjmapi_change_running_a_t;
```

Table E.7 Members of Notification Structure of Change to the RUNNING-A State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.

Member	Type	Input-Output	Description
runa_date	time_t	out	The time of the transition of the target job to RUNNING-A is stored.
all_prec_subjob_exit_date	time_t	out	The preceding sub job end time of the target job is stored.
sum_wait_time	uint64_t	out	The cumulative wait time of the target job is stored.
node_num	uint	out	The number of the allocated nodes of the target job is stored.
node_x	uint	out	The allocated shape of the target job is stored.
node_y	uint		
node_z	uint		
vn_cpu_req	uint32_t	out	The number of allocated cores of the target job by virtual node is stored.
num_alloc_vnode	uint32_t	out	The number of allocated virtual nodes of the target job is stored.
sum_cpu_prealloc_num	uint	out	The total number of CPUs allocated by the scheduler function is stored.
sched_date	time_t	out	The job execution start time of the target job is stored.
mem_job_prealloc	uint64_t	out	The allocated memory amount of the target job is stored.
last_sched_date	struct timespec	out	The scheduling start time of the target job is stored.
backfill_flg	uint8_t	out	The flag of a backfilled job

E.1.5 Notification Structure of Change to the RUNNING State (PJM_CHANGE_RUNNING)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_RUNNING. The following is the notification structure of change to the RUNNING state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_running {
   uint16_t          job_model;          /* Job model */
   uint16_t          job_flags;          /* Job additional information */
   uint16_t          num_retry;          /* Retry count */
   int16_t           pre_jobstatus;      /* Previous job status */
   uint32_t          job_type;           /* Job type */
   uint32_t          jobid;              /* Job ID */
   uint32_t          blkno;              /* Bulk number */
   uint32_t          stepno;             /* Step number */
   int               run_count;          /* RERUN count */
   uint              pro_exit_code;      /* Prologue exit code */
   time_t            start_date;         /* Job start time */
   time_t            run_date;           /* Time of RUNNING transition */
   time_t            fst_start_date;     /* Initial job execution start time */
   uint64_t          sum_runa_time;      /* Cumulative RUNNING-A time (seconds) */
   time_t            prologue_end_date;  /* Prologue end time */
} Pjmapi_change_running_t;
```

Table E.8 Members of Notification Structure of Change to the RUNNING State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The bit indicated by any one of the following macros is set according to the job model.

Member	Type	Input-Output	Description
			PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTERACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
run_count	int	out	The number of times when the target job is re-executed is stored.
pro_exit_code	uint	out	The end code of the prologue script of the target job is stored.
start_date	time_t	out	The job start time of the target job is stored.
run_date	time_t	out	The time of the transition of the target job to RUNNING is stored.
fst_start_date	time_t	out	The time when the target job transition to the RUNNING state occurred for the first time is stored.
sum_runa_time	uint64_t	out	The cumulative time of the RUNNING-A state of the target job (seconds, rounded up to the nearest whole digit) is stored.
prologue_end_date	time_t	out	The prologue end time in the compute node of the target job is stored.

E.1.6 Notification Structure of Change to the RUNOUT State (PJM_CHANGE_RUNOUT)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_RUNOUT. The following is the notification structure of change to the RUNOUT state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_runout {
   uint16_t       job_model;                       /* Job model */
   uint16_t       job_flags;                       /* Job additional information */
   uint16_t       num_retry;                       /* Retry count */
   int16_t        pre_jobstatus;                   /* Previous job status */
   uint32_t       job_type;                        /* Job type */
   uint32_t       jobid;                           /* Job ID */
   uint32_t       blkno;                           /* Bulk number */
   uint32_t       stepno;                          /* Step number */
   uint           exit_code;                       /* exitcode of the user script */
   uint           used_nodenum;                    /* Number of nodes used */
   int32_t        signal_no;                       /* Signal number of the user script */
   int            pjm_code;                        /* PJM code */
   uint           unavailable_nodenum;             /* Number of unavailable nodes */
   uint           sum_cpu_alloc_num;               /* Total number of allocated CPUs */
   uint32_t       used_cpunum;                     /* Number of CPUs used */
   uint           epi_exit_code;                   /* Epilogue exit code */
   uint64_t       sum_usr_cputm;                   /* Total user CPU time */
   uint64_t       sum_sys_cputm;                   /* Total system CPU time */
   time_t         end_date;                        /* Job end time */
   time_t         elapse;                          /* Job execution elapse time */
   time_t         elapse_off_acc;                  /* Job execution elapse time that is not subject
                                                      to billing
   char           last_rscunit [PJM_RSCUNAME_MAX]; /* Submit resource unit name */
   char           last_rscgrp [PJM_RSCGROUP_MAX];  /* Submit resource group name */
   time_t         runout_date;                     /* Time of RUNOUT transition */
   uint64_t       sum_run_time;                    /* Cumulative RUNNING time */
   uint64_t       mem_job_alloc;                   /* Allocated memory amount */
   uint64_t       max_used_mem;                    /* Maximum memory use amount (byte)*/
   uint64_t       usctmut;                         /* Total user CPU time and total system CPU time
                                                      (seconds) */
   time_t         snapshottime;                    /* Data collection year/month/day */
   time_t         epilogue_end_date;               /* Epilogue end time */
   uint64_t       sum_vm_job_use;                  /* Total used memory amount by virtual node */
   uint32_t       affected_nid;                    /* Node ID that affected job result */
   uint32_t       prealloc_rmexit_exitcode;        /* prealloc exit end code */
   uint32_t       predel_rmexit_exitcode;          /* predel exit end code */
   uint32_t       postfree_rmexit_exitcode;        /* postfree exit end code */
   uint64_t       prealloc_start_time;             /* prealloc exit start time */
   uint64_t       prealloc_end_time;               /* prealloc exit end time */
   uint64_t       predel_start_time;               /* predel exit start time */
   uint64_t       predel_end_time;                 /* predel exit end time */
   uint64_t       postfree_start_time;             /* postfree exit start time */
   uint64_t       postfree_end_time;               /* postfree exit end time */
   uint8_t        prealloc_exec_kind;              /* prealloc exit execution timing */
   uint8_t        predel_exec_kind;                /* predel exit execution timing */
   uint8_t        postfree_exec_kind;              /* postfree exit execution timing */
   uint8_t        pad[5];
   uint64_t       job_env_boot_time;               /* Job execution environment boot time */
   uint64_t       job_env_shutdown_time;           /* Job execution environment shutdown time */
   int64_t        fj_profiler;                     /* Fujitsu profiler use count */
   time_t         total_node_down_time;            /* For expansion */
   char           arch_info[16];                   /* Machine type */
   off_t          req_cstmrsc_ofs;                 /* Offset to custom resource information
                                                      (Pjmapi_req_cstmrsc_t) */
   off_t          hw_info_ofs;                     /* Offset to hardware specific information */
} Pjmapi_change_runout_t;
```

```
typedef enum {
   PJM_CSTMRSC_VALUE_TYPE_NUMERIC = 1,
   PJM_CSTMRSC_VALUE_TYPE_STRING = 2
} Pjmapi_cstmrsc_value_type_t;
```

```
typedef struct Pjmapi_req_cstmrsc {
   off_t             next_ofs;        /* Offset where next custom resource information
                                        (Pjmapi_req_cstmrsc_t) is stored */
   char              name[PJM_MAX_CSTM_NAME_LEN];   /* Custom resource name */
   Pjmapi_cstmrsc_info_t  cstmrsc_info;             /* Custom resource type structure */
} Pjmapi_req_cstmrsc_t;
```

```
typedef struct Pjmapi_cstmrsc_info {
   uint8_t          is_pernode;       /* Whether NodeID is specified, 1 if specified */
   uint8_t          value_type;       /* Value type of custom resource (specifiable with
                                         pjmapi_cstmrsc_value_type_t) */
   uint8_t          pad[6];
   union {
        int64_t        num_value;                           /* Requested amount of custom resource */
        char           string_value[PJM_MAX_CSTM_NAME_LEN]; /* Requested type of custom resource */
   } value_rsc;
} Pjmapi_cstmrsc_info_t;
```

```
typedef struct Pjmapi_info_hwspecific_fx {
   uint64_t         tofu_user_comm_recv_byte;             /* Tofu user communication receive
                                                             data size (bytes) */
   uint64_t         tofu_user_comm_send_byte;             /* Tofu user communication send
                                                             data size (bytes) */
   uint64_t         tofu_sys_comm_rsv_byte;               /* Tofu system communication receive
                                                             data size (bytes) */
   uint64_t         tofu_sys_comm_send_byte;              /* Tofu system communication send
                                                             data size (bytes) */
   uint32_t         sum_alloc_assistcpunum;               /* Number of allocated assistant cores */
   uint32_t         sum_used_assistcpunum;                /* Number of assistant cores used */
   uint64_t         sum_usr_assistcputm;                  /* Total user CPU use time of
                                                             assistant core */
   uint64_t         sum_sys_assistcputm;                  /* Total system CPU use time of
                                                             assistant core */
   uint64_t         sum_used_assistant_core_max_mem       /* Maximum use memory amount of
                                                             assistant core */
   uint64_t         sector_cache_using_program_count;     /* Start count of sector cache using
                                                             program */
   uint64_t         intra_node_barrier_using_program_count; /* Count of chip internal barrier using
                                                               program */
   Pjmapi_job_power_consumption_t power_consumption;      /* Power consumption-related
                                                             information */
   Pjmapi_reserved_param_t reserved_param;                /* For future extension */
   Pjmapi_reserved_info_t reserved_info;                  /* For future extension */
} Pjmapi_info_hwspecific_fx_t;
```

```
typedef struct Pjmapi_info_hwspecific_pcc {
   Pjmapi_job_power_consumption_pcc_t power_consumption; /* PC cluster power consumption-related
                                                            information */
} Pjmapi_info_hwspecific_pcc_t;
```

```
typedef struct Pjmapi_job_power_consumption {
   uint16_t         power_consumption_state;             /* Acquisition state of power information */
   uint8_t          utilization_info_of_power_api;       /* Power knob use information */
   uint8_t          pad[1];
   uint32_t         num_cmg;                             /* Number of CMGs */
   off_t            cmgs_ofs;                            /* Offset to power consumption structure
                                                            by CMG */
   Pjmapi_power_consumption_t ideal_cpu_peripherals;     /* Peripheral power consumption
                                                            information in CPU (estimation) */
   Pjmapi_power_consumption_t ideal_opticalmodule;       /* Optical module power consumption
                                                            information (estimation) */
   Pjmapi_power_consumption_t ideal_tofu;                /* Tofu power consumption information
                                                            (estimation) */
   Pjmapi_power_consumption_t ideal_pcie;                /* PCI-E power consumption information
                                                            (estimation) */
   Pjmapi_power_consumption_t ideal_node;                /* Node power consumption information
                                                            (estimation) */
   Pjmapi_power_consumption_t measured_node;             /* Node power consumption information
                                                            (result) */
   struct timespec measure_start_date;                   /* Power measurement start time */
   struct timespec measure_end_date;                     /* Power measurement end time */
} Pjmapi_job_power_consumption_t;
```

```
typedef struct Pjmapi_job_power_consumption_pcc {
   uint16_t         power_consumption_state;             /* Acquisition state of power information */
   uint8_t          pad[2];
   uint32_t         num_pkg;                             /* Number of packages */
   off_t pkgs_ofs;                                       /* Offset to power consumption structure by
                                                            package */
   Pjmapi_power_consumption_t measured_node;             /* Node power consumption information
                                                            (result) */
   struct timespec measure_start_date;                   /* Power measurement start time */
   struct timespec measure_end_date;                     /* Power measurement end time */
} Pjmapi_job_power_consumption_pcc_t;
```

```
typedef struct Pjmapi_cmg_power_consumption {
   int32_t cmgno;                                        /* CMG NUMBER */
   uint8_t pad[4];
   Pjmapi_power_consumption_t ideal_core;                /* Compute core power consumption
                                                            information by CMG (estimation) */
   Pjmapi_power_consumption_t ideal_l2cache;             /* L2 cache power consumption
                                                            information by CMG (estimation) */
   Pjmapi_power_consumption_t ideal_mem;                 /* Memory power consumption
                                                            information by CMG (estimation) */
} Pjmapi_cmg_power_consumption_t;
```

```
typedef struct Pjmapi_pkg_power_consumption {
   int32_t pkgno;                                       /* Package number */
   uint8_t pad[4];
   Pjmapi_power_consumption_t cpu;                      /* CPU power consumption information by
                                                           package */
   Pjmapi_power_consumption_t mem;                      /* Memory power consumption information by
                                                           package */
   Pjmapi_power_consumption_t pp0;                      /* PP0 power consumption information by
                                                           package */
} Pjmapi_pkg_power_consumption_t;
```

```
typedef struct Pjmapi_power_consumption {
   double           avg_power;                          /* Average power consumption */
   double           max_power;                          /* Maximum power consumption */
   double           min_power;                          /* Minimum power consumption */
   double           energy;                             /* Power consumption amount */
} Pjmapi_power_consumption_t;
```

```
typedef struct Pjmapi_reserved_param {
   uint64_t         reserved1;                          /* For future extension */
   uint64_t         reserved2;                          /* For future extension */
   uint64_t         reserved3;                          /* For future extension */
   uint64_t         reserved4;                          /* For future extension */
   uint64_t         reserved5;                          /* For future extension */
   int32_t          reserved6;                          /* For future extension */
   int32_t          reserved7;                          /* For future extension */
   int32_t          reserved8;                          /* For future extension */
   int32_t          reserved9;                          /* For future extension */
   uint64_t         reserved10;                         /* For future extension */
   uint64_t         reserved11;                         /* For future extension */
   off_t            reserved12;                         /* For future extension */
   off_t            reserved13;                         /* For future extension */
} Pjmapi_reserved_param_t;
```

```
typedef struct Pjmapi_reserved_info {
   struct timespec  reserved1;                          /* For future extension */
   struct timespec  reserved2;                          /* For future extension */
   struct timespec  reserved3;                          /* For future extension */
   struct timespec  reserved4;                          /* For future extension */
   struct timespec  reserved5;                          /* For future extension */
   struct timespec  reserved6;                          /* For future extension */
   uint64_t         reserved7;                          /* For future extension */
   double           reserved8;                          /* For future extension */
   double           reserved9;                          /* For future extension */
   uint64_t         reserved10;                         /* For future extension */
   uint64_t         reserved11;                         /* For future extension */
   uint64_t         reserved12;                         /* For future extension */
   uint64_t         reserved13;                         /* For future extension */
   uint64_t         reserved14;                         /* For future extension */
   uint64_t         reserved15;                         /* For future extension */
   uint64_t         reserved16;                         /* For future extension */
   double           reserved17;                         /* For future extension */
   uint64_t         reserved18;                         /* For future extension */
   uint64_t         reserved19;                         /* For future extension */
   uint64_t         reserved20;                         /* For future extension */
   uint64_t         reserved21;                         /* For future extension */
   double           reserved22;                         /* For future extension */
   uint64_t         reserved23;                         /* For future extension */
   uint64_t         reserved24;                         /* For future extension */
   uint64_t         reserved25;                         /* For future extension */
   uint64_t         reserved26;                         /* For future extension */
} Pjmapi_reserved_info_t;
```

Table E.9 Members of Notification Structure of Change to the RUNOUT State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
exit_code	uint	out	The exit code of the shell script of the target job is stored.
used_nodenum	uint	out	The number of nodes used is stored.
signal_no	int32_t	out	The signal number of the shell script of the target job is stored.
pjm_code	int	out	A code indicating the processing result of the job manager function in job execution of the target job is stored.
unavailable_nodenum	uint	out	The number of available nodes in the allocated scope of the target job is stored.
sum_cpu_alloc_num	uint	out	The total number of the allocated CPUs is stored.
used_cpunum	uint32_t	out	The number of CPUs used by the target job is stored.

Member	Type	Input-Output	Description
epi_exit_code	uint	out	The end code of the epilogue script of the target job is stored.
sum_usr_cputm	uint64_t	out	The total user CPU use time is stored.
sum_sys_cputm	uint64_t	out	The total system CPU use time is stored.
end_date	time_t	out	The job end time of the target job is stored.
elapse	time_t	out	The job elapse time of the target job is stored.
elapse_off_acc	time_t	out	Of the job elapse time of the target job, the time that is not subject to billing is stored.
last_rscunit[PJM_RSCUNAME_MAX]	char[]	out	The submit resource unit name of the target job is stored.
last_rscgrp[PJM_RSCGROUP_MAX]	char[]	out	The submit resource group name of the target job is stored.
runout_date	time_t	out	The time of the transition of the target job to RUNOUT is stored.
sum_run_time	uint64_t	out	The cumulative time of the RUNNING state of the target job (seconds, rounded up to the nearest whole digit) is stored.
mem_job_alloc	uint64_t	out	The memory amount allocated in the node of the target job (bytes) is stored.
max_used_mem	uint64_t	out	The maximum memory usage of the target job is stored.
usctmut	uint64_t	out	The total user CPU time and total system CPU time of the target job (seconds) are stored.
snapshottime	time_t	out	The data collection date (year/month/day) of the target job is stored.
epilogue_end_date	time_t	out	The prologue end time in the compute node of the target job is stored.
sum_vm_job_use	uint64_t	out	The total memory amount of the target job by virtual node (bytes) is stored.
affected_nid	uint32_t	out	The node ID that affected the job result of the target job is stored.
prealloc_rmexit_exitcode	uint32_t	out	The end code of the prealloc exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
predel_rmexit_exitcode	uint32_t	out	The end code of the predel exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit

Member	Type	Input-Output	Description
			script 255: Error other than the above
postfree_rmexit_exitcode	uint32_t	out	The end code of the postfree exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
prealloc_start_time	uint64_t	out	The time of the start of the prealloc exit of the target job is stored.
prealloc_end_time	uint64_t	out	The time of the end of the prealloc exit of the target job is stored.
predel_start_time	uint64_t	out	The time of the start of the predel exit of the target job is stored.
predel_end_time	uint64_t	out	The time of the end of the predel exit of the target job is stored.
postfree_start_time	uint64_t	out	The time of the start of the postfree exit of the target job is stored.
postfree_end_time	uint64_t	out	The time of the end of the postfree exit of the target job is stored.
prealloc_exec_kind	uint8_t	out	The prealloc exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 1: Job start timing
predel_exec_kind	uint8_t	out	The predel exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 3: pjdel command execution timing 4: pjhold command execution timing 5: Timing of a deletion request from the job manager function or job scheduler function 6: Timing of a compute node error 7: Timing when the CPU time is exceeded 8: Timing when the elapsed time is exceeded 9: Timing when the memory use amount exceeded
postfree_exec_kind	uint8_t	out	The postfree exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 2: Job end timing
job_env_boot_time	uint64_t	out	The time taken for the boot processing of the job execution environment is stored.
job_env_shutdown_time	uint64_t	out	The time taken for the shutdown processing of the job execution environment is stored.
fj_profiler	int64_t	out	The number of times of Fujitsu profiler use by the target job is stored.
total_node_down_time	time_t	out	For extension
arch_info[16]	char[]	out	The character string indicating a machine type is stored.

Member	Type	Input-Output	Description
req_cstmrsc_ofs	off_t	out	The offset to the requested custom resource information (Pjmapi_req_cstmrsc_t) is stored. The following expression can be used to obtain the pointer to the requested custom resource information. Pjmapi_req_cstmrsc_t* req_cstm_p = (Pjmapi_req_cstmrsc_t *) PJMAPI_OFF_TO_PTR(req_cstmrsc_ofs) If the requested custom resource information does not exist, 0 is stored.
hw_info_ofs	off_t	out	The offset to the hardware specific information is stored. The following expression can be used to obtain the pointer to the hardware specific information by machine type. For machine type FX: Pjmapi_info_hwspecific_fx_t *hw_fx_info_p = (Pjmapi_info_hwspecific_fx_t*) PJMAPI_OFF_TO_PTR(hw_info_ofs) If the hardware specific information does not exist, 0 is stored. For machine type PG: Pjmapi_info_hwspecific_pcc_t *hw_pcc_info_p = (Pjmapi_info_hwspecific_pcc_t*) PJMAPI_OFF_TO_PTR(hw_info_ofs) If the hardware specific information does not exist, 0 is stored.
next_ofs	off_t	out	The offset to the next custom resource information (Pjmapi_req_cstmrsc_t) is stored. If the next custom resource information does not exist, 0 is stored.
name[PJM_MAX_CSTM_ NAME_LEN]	char[]	out	The custom resource name is stored.
cstmrsc_info	Pjmapi_cstmrsc_ info_t	out	The requested amount or requested type of the custom resource information is stored.
is_pernode	uint8_t	out	If the custom resource is a resource by node, 1 is set. If the custom resource is not a resource by node, 0 is set.
value_type	uint8_t	out	The value type of the custom resource (value that can be specified by pjmapi_cstmrsc_value_type_t) Either numeric specification (PJM_CSTMRSC_VALUE_TYPE_NUMERIC) or type specification(PJM_CSTMRSC_VALUE_TYPE_STRING) is stored.
value_rsc	union	out	The requested amount or requested type of the custom resource information is stored.
num_value	int64_t	out	The requested amount of the custom resource information is set.
string_value[PJM_MAX_CSTM _NAME_LEN]	char[]	out	The requested amount of the custom resource information is set.
tofu_user_comm_recv_byte	uint64_t	out	The receive data size used for user level communication of the target job via the Tofu interconnect (bytes) is stored.
tofu_user_comm_send_byte	uint64_t	out	The send data size used for user level communication of the target job via the Tofu interconnect (bytes) is stored.
tofu_sys_comm_rsv_byte	uint64_t	out	The receive data size used for system communication of the target job via the Tofu interconnect (bytes) is stored.

Member	Type	Input-Output	Description
tofu_sys_comm_send_byte	uint64_t	out	The send data size used for system communication of the target job via the Tofu interconnect (bytes) is stored.
sum_alloc_assistcpunum	uint32_t	out	The number of allocated assistant cores set for the job is stored.
sum_used_assistcpunum	uint32_t	out	The number of used assistant cores set for the job is stored.
sum_usr_assistcputm	uint64_t	out	The total CPU use time of the system of the assistant cores set for the job is stored.
sum_sys_assistcputm	uint64_t	out	The total CPU use time of the system of the assistant cores set for the job is stored.
sum_used_assistant_core_max_mem	uint64_t	out	The maximum use amount of the memory used by the assistant cores of the target job (bytes) is stored.
sector_cache_using_program_count	uint64_t	out	The number of times when programs that use the sector cache of the target job are started is stored.
intra_node_barrier_using_program_count	uint64_t	out	The number of times when programs that use the chip internal barrier of the target job are started is stored.
power_consumption	Pjmapi_job_power_consumption_t	out	The power consumption-related information is stored.
reserved_param	Pjmapi_reserved_param_t	out	It is used for future extension.
reserved_info	Pjmapi_reserved_info_t	out	It is used for future extension.
power_consumption	Pjmapi_job_power_consumption_pcc_t	out	Power consumption-related information of the PC cluster is stored.
power_consumption_state	uint16_t	out	The acquisition status of power information of the target job is stored as a value of logical addition flags. The value is as follows. 0x0: The obtained information does not include nodes affected by node-sharing jobs. 0x1: The obtained information includes nodes affected by node-sharing jobs. 0x2: There are nodes whose information failed to be obtained. 0x4: Since there are preceding jobs, information of some nodes was not obtained.
utilization_info_of_power_api	uint8_t	out	Whether the target job uses the Power API and whether the power knob is operated are stored with the corresponding bits set on or off. The value is as follows. 0b00(0): The API is not used and the knob is not operated. 0b01(1): The API is used and the knob is not operated. 0b03(3): The API is used and the knob is operated.
num_cmg	uint32_t	out	The number of CMGs of the target job is stored.
cmgs_ofs	off_t	out	The offset to the power consumption structure by CMG is stored. The following expression can be used to obtain the pointer to the power consumption structure by CMG. Pjmapi_cmg_power_consumption_t *cmgs_p =(Pjmapi_cmg_power_consumption_t *)PJMAPI_OFF_TO_PTR(cmgs_ofs)

Member	Type	Input-Output	Description
ideal_cpu_peripherals	Pjmapi_power_consumption_t	out	Peripheral power consumption information in the CPU (estimation) is stored.
ideal_opticalmodule	Pjmapi_power_consumption_t	out	Optical module power consumption information (estimation) is stored.
ideal_tofu	Pjmapi_power_consumption_t	out	Tofu power consumption information (estimation) is stored.
ideal_pcie	Pjmapi_power_consumption_t	out	PCI-E power consumption information (estimation) is stored.
ideal_node	Pjmapi_power_consumption_t	out	Node power consumption information (estimation) is stored.
measured_node	Pjmapi_power_consumption_t	out	Node power consumption information (result) is stored.
measure_start_date	struct timespec	out	The power measurement start time is stored.
measure_end_date	struct timespec	out	The power measurement end time is stored.
num_pkg	uint32_t	out	The number of packages of the target job is stored.
pkgs_ofs	off_t	out	The offset to the power consumption structure by package is stored.
cmgno	int32_t	out	The CMG number is stored.
ideal_core	Pjmapi_power_consumption_t	out	The compute core power consumption information by CMG (estimation) is stored.
ideal_l2cache	Pjmapi_power_consumption_t	out	The L2 cache power consumption information by CMG (estimation) is stored.
ideal_mem	Pjmapi_power_consumption_t	out	The memory power consumption information by CMG (estimation) is stored.
pkgno	int32_t	out	Package number is stored.
cpu	Pjmapi_power_consumption_t	out	The CPU power consumption information by package is stored.
mem	Pjmapi_power_consumption_t	out	The memory power consumption information by package is stored.
pp0	Pjmapi_power_consumption_t	out	The pp0 power consumption information by package is stored.
avg_power	double	out	The average power consumption of the target power items is stored.
max_power	double	out	The maximum power consumption of the target power items is stored.
min_power	double	out	The minimum power consumption of the target power items is stored.
energy	double	out	The power consumption amount of the target power items is stored.

(*) The member reserved*n* is for future expansion.

E.1.7 Notification Structure of Change to the EXIT State (PJM_CHANGE_EXIT)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_EXIT. The following is the notification structure of change to the EXIT state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_exit {
   uint16_t            job_model;                 /* Job model */
   uint16_t            job_flags;                 /* Job additional information */
   uint16_t            num_retry;                 /* Retry count */
   int16_t             pre_jobstatus;             /* Previous job status */
   uint32_t            job_type;                  /* Job type */
   uint32_t            jobid;                     /* Job ID */
   uint32_t            blkno;                     /* Bulk number */
   uint32_t            stepno;                    /* Step number */
   int                 pjm_code;                  /* PJM code */
   uid_t               lasthold_uid;              /* User ID held/cancelled in the last state */
   int                 mailflag;                  /* Mail send flag */
   time_t              exit_date;                 /* Time of EXIT transition */
   time_t              elapse;                    /* Job execution elapse time */
   time_t              elapse_off_acc;            /* Job execution elapse time that is not subject
                                                     to billing */
   char                reason[64];                /* REASON */
   uint64_t            sum_runa_time;             /* Cumulative RUNNING-A time (seconds) */
   uint64_t            sum_hold_time;             /* Cumulative HOLD time (seconds) */
   uint64_t            sum_wait_time;             /* Cumulative wait time (seconds) */
   time_t              snapshottime;              /* Data collection year/month/day */
   uint32_t            affected_nid;              /* Node ID that affected job result */
   uint32_t            prealloc_rmexit_exitcode;  /* prealloc exit end code */
   uint32_t            predel_rmexit_exitcode;    /* predel exit end code */
   uint32_t            postfree_rmexit_exitcode;  /* postfree exit end code */
   uint64_t            prealloc_start_time;       /* prealloc exit start time */
   uint64_t            prealloc_end_time;         /* prealloc exit end tim */
   uint64_t            predel_start_time;         /* predel exit start time */
   uint64_t            predel_end_time;           /* predel exit end time */
   uint64_t            postfree_start_time;       /* postfree exit start time */
   uint64_t            postfree_end_time;         /* postfree exit end time */
   uint8_t             prealloc_exec_kind;        /* prealloc exit execution timing */
   uint8_t             predel_exec_kind;          /* predel exit execution timing */
   uint8_t             postfree_exec_kind;        /* postfree exit execution timing */
   uint8_t             pad1[5];
} Pjmapi_change_exit_t;
```

Table E.10 Members of Notification Structure of Change to the EXIT State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.

Member	Type	Input-Output	Description
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
pjm_code	int	out	A code indicating the processing result of the job manager function in job execution of the target job is stored.
lasthold_uid	uid_t	out	If the target job has ever been held, the user ID of the last user who held it is stored. If the job was canceled, the user ID of the user who canceled it is stored.
mailflag	int	out	The flag about whether there is mail transfer of the target job is stored. The value is as follows. 1: Job start 2: Job end 4: Error occurrence 8: Statistical information output (without node information) 16: Statistical information output (with node information)
exit_date	time_t	out	The time of the transition of the target job to EXIT is stored.
elapse	time_t	out	The job elapse time of the target job is stored.
elapse_off_acc	time_t	out	Of the job elapse time of the target job, the time that is not subject to billing is stored.
reason[64]	char[]	out	The REASON of the target job is stored.
sum_runa_time	uint64_t	out	The cumulative time of the RUNNING-A state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_hold_time	uint64_t	out	The cumulative time of the HOLD state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_wait_time	uint64_t	out	The cumulative wait time of the target job is stored.
snapshottime	time_t	out	The data collection date (year/month/day) of the target job is stored.
affected_nid	uint32_t	out	The node ID that affected the job result of the target job is stored.
prealloc_rmexit_exitcode	uint32_t	out	The end code of the prealloc exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
predel_rmexit_exitcode	uint32_t	out	The end code of the predel exit of the target job is stored. The value to be set is as follows. 0: Normal end

Member	Type	Input-Output	Description
			1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
postfree_rmexit_exitcode	uint32_t	out	The end code of the postfree exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
prealloc_start_time	uint64_t	out	The time of the start of the prealloc exit of the target job is stored.
prealloc_end_time	uint64_t	out	The time of the end of the prealloc exit of the target job is stored.
predel_start_time	uint64_t	out	The time of the start of the predel exit of the target job is stored.
predel_end_time	uint64_t	out	The time of the end of the predel exit of the target job is stored.
postfree_start_time	uint64_t	out	The time of the start of the postfree exit of the target job is stored.
postfree_end_time	uint64_t	out	The time of the end of the postfree exit of the target job is stored.
prealloc_exec_kind	uint8_t	out	The prealloc exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 1: Job start timing
predel_exec_kind	uint8_t	out	The predel exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 3: pjdel command execution timing 4: pjhold command execution timing 5: Timing of a deletion request from the job manager function or job scheduler function 6: Timing of a compute node error 7: Timing when the CPU time is exceeded 8: Timing when the elapsed time is exceeded 9: Timing when the memory use amount exceeded
postfree_exec_kind	uint8_t	out	The postfree exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 2: Job end timing

E.1.8 Notification Structure of Change to the CANCEL State (PJM_CHANGE_CANCEL)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_CANCEL. The following is the notification structure of change to the CANCEL state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_cancel {
   uint16_t           job_model;                 /* Job model */
   uint16_t           job_flags;                 /* Job additional information */
   uint16_t           num_retry;                 /* Retry count */
   int16_t            pre_jobstatus;             /* Previous job status */
   uint32_t           job_type;                  /* Job type */
   uint32_t           jobid;                     /* Job ID */
   uint32_t           blkno;                     /* Bulk number */
   uint32_t           stepno;                    /* Step number */
   int                pjm_code;                  /* PJM code */
   uid_t              lasthold_uid;              /* User ID held/cancelled in the last state */
   int                mailflag;                  /* Mail send flag */
   uint               pro_exit_code;             /* Prologue exit code */
   time_t             cancel_date;               /* Time of CANCEL transition */
   time_t             elapse;                    /* Job execution elapse time */
   time_t             elapse_off_acc;            /* Job execution elapse time that is not subject
                                                    to billing */
   char               reason[64];                /* REASON */
   uint64_t           sum_runa_time;             /* Cumulative RUNNING-A time (seconds) */
   uint64_t           sum_hold_time;             /* Cumulative HOLD time (seconds) */
   uint64_t           sum_wait_time;             /* Cumulative wait time (seconds) */
   time_t             pjdel_date;                /* Job deletion request time *./
   time_t             delete_date;               /* Job deletion time */
   time_t             snapshottime;              /* Data collection year/month/day */
   time_t             prologue_end_date;         /* Prologue end time */
   time_t             all_prec_subjob_exit_date; /* Preceding sub job end time */
   uint32_t           affected_nid;              /* Node ID that affected job result */
   uint32_t           prealloc_rmexit_exitcode;  /* prealloc exit end code */
   uint32_t           predel_rmexit_exitcode;    /* predel exit end code */
   uint32_t           postfree_rmexit_exitcode;  /* postfree exit end code */
   uint64_t           prealloc_start_time;       /* prealloc exit start time */
   uint64_t           prealloc_end_time;         /* prealloc exit end tim */
   uint64_t           predel_start_time;         /* predel exit start time */
   uint64_t           predel_end_time;           /* predel exit end time */
   uint64_t           postfree_start_time;       /* postfree exit start time */
   uint64_t           postfree_end_time;         /* postfree exit end time */
   uint8_t            prealloc_exec_kind;        /* prealloc exit execution timing */
   uint8_t            predel_exec_kind;          /* predel exit execution timing */
   uint8_t            postfree_exec_kind;        /* postfree exit execution timing */
   uint8_t            pad1[5];
} Pjmapi_change_cancel_t;
```

Table E.11 Members of Notification Structure of Change to the CANCEL State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows.

Member	Type	Input-Output	Description
			PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
pjm_code	int	out	A code indicating the processing result of the job manager function in job execution of the target job is stored.
lasthold_uid	uid_t	out	If the target job has ever been held, the user ID of the last user who held it is stored. If the job was canceled, the user ID of the user who canceled it is stored.
mailflag	int	out	The flag about whether there is mail transfer of the target job is stored. The value is as follows. 1: Job start 2: Job end 4: Error occurrence 8: Statistical information output (without node information) 16: Statistical information output (with node information)
pro_exit_code	uint	out	The end code of the prologue script of the target job is stored.
cancel_date	time_t	out	The time of the transition of the target job to CANCEL is stored.
all_prec_subjob_exit_date	time_t	out	The preceding sub job end time of the target job is stored.
elapse	time_t	out	The job elapse time of the target job is stored.
elapse_off_acc	time_t	out	Of the job elapse time of the target job, the time that is not subject to billing is stored.
reason[64]	char[]	out	The REASON of the target job is stored.
sum_runa_time	uint64_t	out	The cumulative time of the RUNNING-A state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_hold_time	uint64_t	out	The cumulative time of the HOLD state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_wait_time	uint64_t	out	The cumulative wait time of the target job is stored.
pjdel_date	time_t	out	The job deletion request time of the target job is stored.
delete_date	time_t	out	The job deletion time of the target job is stored.
snapshottime	time_t	out	The data collection date (year/month/day) of the target job is stored.
prologue_end_date	time_t	out	The prologue end time in the compute node of the target job is stored.
affected_nid	uint32_t	out	The node ID that affected the job result of the target job is stored.

Member	Type	Input-Output	Description
prealloc_rmexit_exitcode	uint32_t	out	The end code of the prealloc exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
predel_rmexit_exitcode	uint32_t	out	The end code of the predel exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
postfree_rmexit_exitcode	uint32_t	out	The end code of the postfree exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
prealloc_start_time	uint64_t	out	The time of the start of the prealloc exit of the target job is stored.
prealloc_end_time	uint64_t	out	The time of the end of the prealloc exit of the target job is stored.
predel_start_time	uint64_t	out	The time of the start of the predel exit of the target job is stored.
predel_end_time	uint64_t	out	The time of the end of the predel exit of the target job is stored.
postfree_start_time	uint64_t	out	The time of the start of the postfree exit of the target job is stored.
postfree_end_time	uint64_t	out	The time of the end of the postfree exit of the target job is stored.
prealloc_exec_kind	uint8_t	out	The prealloc exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 1: Job start timing
predel_exec_kind	uint8_t	out	The predel exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 3: pjdel command execution timing 4: pjhold command execution timing 5: Timing of a deletion request from the job manager function or job scheduler function

Member	Type	Input-Output	Description
			6: Timing of a compute node error 7: Timing when the CPU time is exceeded 8: Timing when the elapsed time is exceeded 9: Timing when the memory use amount exceeded
postfree_exec_kind	uint8_t	out	The postfree exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 2: Job end timing

E.1.9 Notification Structure of Change to the HOLD State (PJM_CHANGE_HOLD)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_HOLD. The following is the notification structure of change to the HOLD state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_hold {
   uint16_t            job_model;                 /* Job model */
   uint16_t            job_flags;                 /* Job additional information */
   uint16_t            num_retry;                 /* Retry count */
   int16_t             pre_jobstatus;             /* Previous job status */
   uint32_t            job_type;                  /* Job type */
   uint32_t            jobid;                     /* Job ID */
   uint32_t            blkno;                     /* Bulk number */
   uint32_t            stepno;                    /* Step number */
   uid_t               lasthold_uid;              /* User ID held/cancelled in the last state */
   uint                hold_count;                /* HOLD count */
   int                 mailflag;                  /* Mail send flag */
   uint                pro_exit_code;             /* Prologue exit code */
   time_t              hold_date;                 /* Time of HOLD transition */
   uint64_t            sum_runa_time;             /* Cumulative RUNNING-A time (seconds) */
   uint64_t            sum_wait_time;             /* Cumulative wait time (seconds) */
   time_t              prologue_end_date;         /* Prologue end time */
   time_t              all_prec_subjob_exit_date; /* Preceding sub job end time */
   char                reason[64];                /* REASON */
   uint32_t            affected_nid;              /* Node ID that affected job result */
   uint32_t            prealloc_rmexit_exitcode;  /* prealloc exit end code */
   uint32_t            predel_rmexit_exitcode;    /* predel exit end code */
   uint32_t            postfree_rmexit_exitcode;  /* postfree exit end code */
   uint64_t            prealloc_start_time;       /* prealloc exit start time */
   uint64_t            prealloc_end_time;         /* prealloc exit end tim */
   uint64_t            predel_start_time;         /* predel exit start time */
   uint64_t            predel_end_time;           /* predel exit end time */
   uint64_t            postfree_start_time;       /* postfree exit start time */
   uint64_t            postfree_end_time;         /* postfree exit end time */
   uint8_t             prealloc_exec_kind;        /* prealloc exit execution timing */
   uint8_t             predel_exec_kind;          /* predel exit execution timing */
   uint8_t             postfree_exec_kind;        /* postfree exit execution timing */
   uint8_t             pad1[5];
} Pjmapi_change_hold_t;
```

Table E.12 Members of Notification Structure of Change to the HOLD State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model.

Member	Type	Input-Output	Description
			PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
lasthold_uid	uid_t	out	If the target job has ever been held, the user ID of the last user who held it is stored. If the job was canceled, the user ID of the user who canceled it is stored.
hold_count	uint	out	The number of times when the target job was held is stored.
mailflag	int	out	The flag about whether there is mail transfer of the target job is stored. The value is as follows. 1: Job start 2: Job end 4: Error occurrence 8: Statistical information output (without node information) 16: Statistical information output (with node information)
pro_exit_code	uint	out	The end code of the prologue script of the target job is stored.
hold_date	time_t	out	The time of the transition of the target job to HOLD is stored.
sum_runa_time	uint64_t	out	The cumulative time of the RUNNING-A state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_wait_time	uint64_t	out	The cumulative wait time of the target job is stored.
prologue_end_date	time_t	out	The prologue end time in the compute node of the target job is stored.
all_prec_subjob_exit_date	time_t	out	The preceding sub job end time of the target job is stored.
reason[64]	char[]	out	The REASON of the target job is stored.
affected_nid	uint32_t	out	The node ID that affected the job result of the target job is stored.
prealloc_rmexit_exitcode	uint32_t	out	The end code of the prealloc exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job

Member	Type	Input-Output	Description
			102: Failure in execution of the resource management exit script 255: Error other than the above
predel_rmexit_exitcode	uint32_t	out	The end code of the predel exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
postfree_rmexit_exitcode	uint32_t	out	The end code of the postfree exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
prealloc_start_time	uint64_t	out	The time of the start of the prealloc exit of the target job is stored.
prealloc_end_time	uint64_t	out	The time of the end of the prealloc exit of the target job is stored.
predel_start_time	uint64_t	out	The time of the start of the predel exit of the target job is stored.
predel_end_time	uint64_t	out	The time of the end of the predel exit of the target job is stored.
postfree_start_time	uint64_t	out	The time of the start of the postfree exit of the target job is stored.
postfree_end_time	uint64_t	out	The time of the end of the postfree exit of the target job is stored.
prealloc_exec_kind	uint8_t	out	The prealloc exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 1: Job start timing
predel_exec_kind	uint8_t	out	The predel exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 3: pjdel command execution timing 4: pjhold command execution timing 5: Timing of a deletion request from the job manager function or job scheduler function 6: Timing of a compute node error 7: Timing when the CPU time is exceeded 8: Timing when the elapsed time is exceeded 9: Timing when the memory use amount exceeded
postfree_exec_kind	uint8_t	out	The postfree exit execution timing of the target job is stored. The value to be set is as follows.

Member	Type	Input-Output	Description
			0: Not executed 2: Job end timing

E.1.10 Notification Structure of Change to the ERROR State (PJM_CHANGE_ERROR)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_ERROR. The following is the notification structure of change to the ERROR state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_error {
   uint16_t           job_model;                 /* Job model */
   uint16_t           job_flags;                 /* Job additional information */
   uint16_t           num_retry;                 /* Retry count */
   int16_t            pre_jobstatus;             /* Previous job status */
   uint32_t           job_type;                  /* Job type */
   uint32_t           jobid;                     /* Job ID */
   uint32_t           blkno;                     /* Bulk number */
   uint32_t           stepno;                    /* Step number */
   int                pjm_code;                  /* PJM code */
   uid_t              lasthold_uid;              /* User ID held/cancelled in the last state */
   int                mailflag;                  /* Mail send flag */
   uint               pro_exit_code;             /* Prologue exit code */
   time_t             err_date;                  /* Time of ERR transition */
   time_t             all_prec_subjob_exit_date; /* Preceding sub job end time */
   char               reason[64];                /* REASON */
   uint64_t           sum_runa_time;             /* Cumulative RUNNING-A time (seconds) */
   uint64_t           sum_hold_time;             /* Cumulative HOLD time (seconds) */
   uint64_t           sum_wait_time;             /* Cumulative wait time (seconds) */
   time_t             snapshottime;              /* Data collection year/month/day */
   time_t             prologue_end_date;         /* Prologue end time */
   uint32_t           affected_nid;              /* Node ID that affected job result */
   uint32_t           prealloc_rmexit_exitcode;  /* prealloc exit end code */
   uint32_t           predel_rmexit_exitcode;    /* predel exit end code */
   uint32_t           postfree_rmexit_exitcode;  /* postfree exit end code */
   uint64_t           prealloc_start_time;       /* prealloc exit start time */
   uint64_t           prealloc_end_time;         /* prealloc exit end tim */
   uint64_t           predel_start_time;         /* predel exit start time */
   uint64_t           predel_end_time;           /* predel exit end time */
   uint64_t           postfree_start_time;       /* postfree exit start time */
   uint64_t           postfree_end_time;         /* postfree exit end time */
   uint8_t            prealloc_exec_kind;        /* prealloc exit execution timing */
   uint8_t            predel_exec_kind;          /* predel exit execution timing */
   uint8_t            postfree_exec_kind;        /* postfree exit execution timing */
   uint8_t            pad1[5];
} Pjmapi_change_error_t;
```

Table E.13 Members of Notification Structure of Change to the ERROR State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job

Member	Type	Input-Output	Description
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
pjm_code	int	out	A code indicating the processing result of the job manager function in job execution of the target job is stored.
lasthold_uid	uid_t	out	If the target job has ever been held, the user ID of the last user who held it is stored. If the job was canceled, the user ID of the user who canceled it is stored.
mailflag	int	out	The flag about whether there is mail transfer of the target job is stored. The value is as follows. 1: Job start 2: Job end 4: Error occurrence 8: Statistical information output (without node information) 16: Statistical information output (with node information)
pro_exit_code	uint	out	The end code of the prologue script of the target job is stored.
err_date	time_t	out	The time of the transition of the target job to ERROR is stored.
all_prec_subjob_exit_date	time_t	out	The preceding sub job end time of the target job is stored.
reason[64]	char	out	The REASON of the target job is stored.
sum_runa_time	uint64_t	out	The cumulative time of the RUNNING-A state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_hold_time	uint64_t	out	The cumulative time of the HOLD state of the target job (seconds, rounded up to the nearest whole digit) is stored.
sum_wait_time	uint64_t	out	The cumulative wait time of the target job is stored.
snapshottime	time_t	out	The data collection date (year/month/day) of the target job is stored.
prologue_end_date	time_t	out	The prologue end time in the compute node of the target job is stored.
affected_nid	uint32_t	out	The node ID that affected the job result of the target job is stored.
prealloc_rmexit_exitcode	uint32_t	out	The end code of the prealloc exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job

Member	Type	Input-Output	Description
			3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
predel_rmexit_exitcode	uint32_t	out	The end code of the predel exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
postfree_rmexit_exitcode	uint32_t	out	The end code of the postfree exit of the target job is stored. The value to be set is as follows. 0: Normal end 1: Specification of setting the job in an error state 2: Specification of re-execution of the job 3: Specification of setting the job in the HOLD state 4: Specification of deleting the job 102: Failure in execution of the resource management exit script 255: Error other than the above
prealloc_start_time	uint64_t	out	The time of the start of the prealloc exit of the target job is stored.
prealloc_end_time	uint64_t	out	The time of the end of the prealloc exit of the target job is stored.
predel_start_time	uint64_t	out	The time of the start of the predel exit of the target job is stored.
predel_end_time	uint64_t	out	The time of the end of the predel exit of the target job is stored.
postfree_start_time	uint64_t	out	The time of the start of the postfree exit of the target job is stored.
postfree_end_time	uint64_t	out	The time of the end of the postfree exit of the target job is stored.
prealloc_exec_kind	uint8_t	out	The prealloc exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 1: Job start timing
predel_exec_kind	uint8_t	out	The predel exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 3: pjdel command execution timing 4: pjhold command execution timing 5: Timing of a deletion request from the job manager function or job scheduler function 6: Timing of a compute node error 7: Timing when the CPU time is exceeded 8: Timing when the elapsed time is exceeded 9: Timing when the memory use amount exceeded

Member	Type	Input-Output	Description
postfree_exec_kind	uint8_t	out	The postfree exit execution timing of the target job is stored. The value to be set is as follows. 0: Not executed 2: Job end timing

E.1.11 Notification Structure of Change to the REJECT State (PJM_CHANGE_REJECT)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_REJECT. The following is the notification structure of change to the REJECT state of the job reported by the argumaent data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_reject {
   uint16_t            job_model;                /* Job model */
   uint16_t            job_flags;                /* Job additional information */
   uint32_t            job_type;                 /* Job type */
   uint32_t            jobid;                    /* Job ID */
   uint32_t            blkno;                    /* Bulk number */
   uint32_t            stepno;                   /* Step number */
   int                 pjm_code;                 /* PJM code */
   time_t              reject_date;              /* Reject transition time */
} Pjmapi_change_reject_t;
```

Table E.14 Members of Notification Structure of Change to the REJECT State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
pjm_code	int	out	A code indicating the processing result of the job manager function in job execution of the target job is stored.
reject_date	time_t	out	The transition time to REJECT is stored.

E.1.12 Notification Structure of Change to the RUNNING-P State (PJM_CHANGE_RUNNING_P)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_RUNNING_P. The following is the notification structure of change to the RUNNING-P state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_running_p {
   uint16_t           job_model;               /* Job model */
   uint16_t           job_flags;               /* Job additional information */
   uint16_t           num_retry;               /* Retry count */
   int16_t            pre_jobstatus;           /* Previous job status */
   uint32_t           job_type;                /* Job type */
   uint32_t           jobid;                   /* Job ID */
   uint32_t           blkno;                   /* Bulk number */
   uint32_t           stepno;                  /* Step number */
   time_t             prologue_start_date;     /* Prologue start time */
   time_t             runp_date;               /* Time of RUNNING-P transition */
} Pjmapi_change_running_p_t;
```

Table E.15 Members of Notification Structure of Change to the RUNNING-P State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
prologue_start_date	time_t	out	The prologue start time in the compute node of the target job is stored.
runp_date	time_t	out	The time of the transition of the state of the PJM of the target job to RUNNING_P is stored.

E.1.13 Notification Structure of Change to the RUNNING-E State (PJM_CHANGE_RUNNING_E)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_RUNNING_E. The following is the notification structure of change to the RUNNING-E state of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_running_e {
   uint16_t           job_model;              /* Job model */
   uint16_t           job_flags;              /* Job additional information */
   uint16_t           num_retry;              /* Retry count */
   int16_t            pre_jobstatus;          /* Previous job status */
   uint32_t           job_type;               /* Job type */
   uint32_t           jobid;                  /* Job ID */
   uint32_t           blkno;                  /* Bulk number */
   uint32_t           stepno;                 /* Step number */
   time_t             epilogue_start_date;    /* Epilogue start time */
   time_t             rune_date;              /* Time of RUNNING-E transition */
} Pjmapi_change_running_e_t;
```

Table E.16 Members of Notification Structure of Change to the RUNNING-E State

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
epilogue_start_date	time_t	out	The epilogue start time in the compute node of the target job is stored.
rune_date	time_t	out	The time of the transition of the state of the PJM of the target job to RUNNING_E is stored.

E.1.14 Notification Structure of Scheduling Result (PJM_CHANGE_SCHED)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_SCHED. The following is the notification structure of the scheduling result of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_sched {
   uint16_t            job_model;                 /* Job model */
   uint16_t            job_flags;                 /* Job additional information */
   uint16_t            num_retry;                 /* Retry count */
   int16_t             pre_jobstatus;             /* Previous job status */
   uint32_t            job_type;                  /* Job type */
   uint32_t            jobid;                     /* Job ID */
   uint32_t            blkno;                     /* Bulk number */
   uint32_t            stepno;                    /* Step number */
   uint                node_num;                  /* Number of allocated nodes */
   uint                node_x;                    /* Allocated node shape x */
   uint                node_y;                    /* Allocated node shape y */
   uint                node_z;                    /* Allocated node shape z */
   uint32_t            vn_cpu_req;                /* Requested number of CPU cores
                                                     by virtual node */
   uint32_t            num_alloc_vnode;           /* Number of allocated virtual nodes */
   uint                sum_cpu_prealloc_num;      /* Total number of scheduler allocation CPUs */
   uint                pad2;
   time_t              sched_date;                /* Job execution start time */
   uint64_t            mem_job_prealloc;          /* Scheduler allocation memory amount (bytes) */
   off_t               ndlist_ofs;                /* Offset to node ID list */
   off_t               tofulist_ofs;              /* Offset to Tofu coordinate list */
   struct timespec     last_sched_date;           /* Scheduling start time */
   uint8_t             backfill_flg;              /* Backfill flag */
   uint8_t             pad1[7];
} Pjmapi_change_sched_t;
```

Table E.17 Members of Notification Structure of Scheduling Result

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
num_retry	uint16_t	out	The retry count of the target job is stored.
pre_jobstatus	int16_t	out	The previous status of the target job before the current status is stored.
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
node_num	uint	out	The number of the allocated nodes of the target job is stored.
node_x	uint	out	The allocated shape of the target job is stored.
node_y	uint		
node_z	uint		

Member	Type	Input-Output	Description
vn_cpu_req	uint32_t	out	The number of allocated cores of the target job by virtual node is stored.
num_alloc_vnode	uint32_t	out	The number of allocated virtual nodes of the target job is stored.
sum_cpu_prealloc_num	uint	out	The total number of CPUs allocated by the scheduler function is stored.
sched_date	time_t	out	The job execution start time of the target job is stored.
mem_job_prealloc	uint64_t	out	The allocated memory amount of the target job is stored.
ndlist_ofs	off_t	out	The offset to the character string, in which the node ID list set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. int nolist= (int *) PJMAPI_OFF_TO_PTR(ndlist_ofs)
tofulist_ofs	off_t	out	The offset to the character string, in which the Tofu coordinate list set for the job is stored, is stored. The following expression can be used to obtain the pointer to the character string. tofu_3d_t tofulist= (tofu_3d_t *) PJMAPI_OFF_TO_PTR(tofulist_ofs) (*1) The Tofu coordinates are stored in the structure of the Tofu coordinates. For details, see "Table E.4 Members of Tofu coordinate Structure." in " E.1.1 Job Information Notification Structure (PJM_INFO_JOB)."
last_sched_date	struct timespec	out	The scheduling start time of the target job is stored.
backfill_flg	uint8_t	out	The flag of a backfilled job

E.1.15 Notification Structure of Attribute Change (PJM_CHANGE_ALTER)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_ALTER. The following is the notification structure of attribute change of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_alter {
   uint16_t           job_model;                      /* Job model */
   uint16_t           job_flags;                      /* Job additional information */
   uint16_t           pad[2];
   uint32_t           job_type;                       /* Job type */
   uint32_t           jobid;                          /* Job ID */
   uint32_t           blkno;                          /* Bulk number */
   uint32_t           stepno;                         /* Step number */
   char               rscunit_name[PJM_RSCUNAME_MAX]; /* Resource unit name */
   char               rscgrp_name[PJM_RSCGROUP_MAX];  /* Resource group name */
   uint64_t           elapse_limit;                   /* Limit value of job elapse time */
   int16_t            job_aprio;                      /* Job priority level within
                                                         the resource unit */
   int16_t            job_uprio;                      /* Job priority level within the same user */
   off_t              req_cstmrsc_ofs;                /* Offset to custom resource information
                                                         (Pjmapi_req_cstmrsc_t) */
} Pjmapi_change_alter_t;
```

Table E.18 Members of Notification Structure of Attribute Change

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
job_type	uint32_t	out	The job type of the target job is stored. The job type is as follows. PJM_JOBTYPE_BATCH: Batch job PJM_JOBTYPE_INTARACT: Interactive job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
rscunit_name [PJM_RSCUNAME_MAX]	char	out	The resource unit name of the job is stored.
rscgrp_name [PJM_RSCGROUP_MAX]	char	out	The resource group name of the job is stored.
elapse_limit	uint64_t	out	The limit value of the elapsed time is stored. In case of UNLIMITED, PJM_RLIM_INFINITY(~0ULL) is stored. If limit values of the elapsed time are specified as a range, the maximum time of the limit of the elapse time (seconds) is set.
job_aprio	int16_t	out	The job priority level within the resource unit is stored.
job_uprio	int16_t	out	The job priority level within the same user is stored.
req_cstmrsc_ofs	off_t	out	The offset to the requested custom resource information (Pjmapi_req_cstmrsc_t) is stored. The following expression can be used to obtain the pointer to the requested custom resource information. Pjmapi_req_cstmrsc_t* req_cstm_p = (Pjmapi_req_cstmrsc_t *) PJMAPI_OFF_TO_PTR(req_cstmrsc_ofs) If the requested custom resource information does not exist, 0 is stored.

E.1.16 Simple Data Notification Structure (PJM_CHANGE_SIMPLE_DATA)

The function PJM_read_data() reads notification data that is received when the notification type is PJM_CHANGE_SIMPLE_DATA. The following is the notification structure of simple data of the job reported by the argument data_p of the function PJM_read_data().

```
typedef struct Pjmapi_change_simple_data {
   uint16_t           job_model;              /* Job model */
   uint16_t           job_flags;              /* Job additional information */
   uint32_t           jobid;                  /* Job ID */
   uint32_t           blkno;                  /* Bulk number */
   uint32_t           stepno;                 /* Step number */
   time_t              accept_date;              /* Job submission time */
} Pjmapi_change_simple_data_t;
```

Table E.19 Members of Simple Data Notification Structure

Member	Type	Input-Output	Description
job_model	uint16_t	out	The job model of the target job is stored. The bit indicated by any one of the following macros is set according to the job model. PJM_JOBMODEL_NORMAL: Normal job PJM_JOBMODEL_BULK: Bulk job PJM_JOBMODEL_STEP: Step job
job_flags	uint16_t	out	Job additional information of the target job is stored. The job additional information is as follows. PJM_JOBFLAGS_BULK_SUBJOB: Sub job of a bulk job
jobid	uint32_t	out	The job ID of the target job is stored.
blkno	uint32_t	out	The bulk number of the target job is stored.
stepno	uint32_t	out	The step number of the target job is stored.
accept_date	time_t	out	The submission time of the target job is stored.

The data structure for simple data notification Pjmapi_change_simple_data_t is common to all the notification events.

Each parameter of Pjmapi_change_simple_data_t is the minimum needed information for obtaining detailed information from the notification information of this simple data by using the pmdumpjobinfo command. accept_date (job submission time) is used to identify the job when jobid exceeds UINT32_MAX and makes a circuit.